Benutzerhandbuch für Echtzeit-Streaming

Amazon IVS

Amazon IVS: Benutzerhandbuch für Echtzeit-Streaming

Table of Contents

Was ist Amazon-IVS-Echtzeit-Streaming?

Amazon Interactive Video Service (IVS)-Echtzeit-Streaming bietet Ihnen alles, was Sie benötigen, um Ihren Anwendungen Audio und Video in Echtzeit hinzuzufügen.

Stärken:

- Latenz in Echtzeit – Entwickeln Sie Anwendungen für latenzsensitive Anwendungsfälle und helfen Sie Ihren Zuschauern, mit IVS-Echtzeit-Streaming in Verbindung zu bleiben und zu interagieren. Stellen Sie Live-Streams mit einer Latenz bereit, die vom Host bis zum Zuschauer unter 300 Millisekunden liegen kann.
- Hohe Parallelität – Nutzen Sie das Potenzial groß angelegter Interaktionen mit IVS-Echtzeit-Streaming. Sie können mehr als 25 000 Zuschauer aufnehmen und bis zu 12 Moderatoren die virtuelle Bühne betreten lassen. (Standardlimits und Anweisungen zur Beantragung einer Erhöhung finden Sie unter Service Quotas für Echtzeit-Streaming und Streaming mit niedriger Latenz.)
- Für Mobilgeräte optimiert – IVS-Echtzeit-Streaming ist für mobile Anwendungsfälle optimiert und eignet sich für eine Vielzahl von Geräten und Netzwerkfunktionen. Durch die Integration der Amazon IVS Broadcast SDKs für Android und iOS können Ihre Benutzer als Hosts oder Zuschauer interagieren und hochwertige Live-Streams auf ihren Mobilgeräten genießen.

Anwendungsfälle:

- Gastspots – Entwickeln Sie Anwendungen, mit denen Gastgeber Gäste „auf der Stage“ bewerben können, sodass Zuschauer zu Hosts für Interaktionen in Echtzeit werden.
- Versus-Modus (VS) – Produzieren Sie Erlebnisse mit nebeneinanderliegenden Wettbewerben und lassen Sie die Zuschauer zuschauen, wenn die Hosts in Echtzeit gegeneinander antreten.
- Audioräume – Laden Sie Zuhörer ein, als Gäste an der Konversation teilzunehmen, und fördern Sie ein intensiveres Engagement in Ihren Audioräumen.
- Live-Videoauktionen – Verwandeln Sie Auktionen in interaktive Videoevents und sorgen Sie mit Latenz in Echtzeit für Spannung und Integrität.

Neben der Produktdokumentation hier finden Sie https://ivs.rocks/, eine spezielle Website zum Durchsuchen veröffentlichter Inhalte (Demos, Codebeispiele, Blog-Posts), Kostenschätzungen und Erleben von Amazon IVS durch Live-Demos.

Globale Lösung, regionale Kontrolle

Streaming und Anzeigen sind global

Sie können Amazon IVS verwenden, um für Zuschauer weltweit zu streamen:

- Wenn Sie streamen, nimmt Amazon IVS automatisch Videos an einem Standort in Ihrer Nähe auf.
- Zuschauer können Ihre Livestreams weltweit ansehen.

Eine andere Möglichkeit, dies zu sagen, ist, dass die „Datenebene“ global ist. Die Datenebene bezieht sich auf Streaming/Aufnahme und Betrachtung.

Kontrolle ist Regional

Während die Amazon IVS-Datenebene global ist, ist die „Steuerungsebene“ regional. Die Steuerebene bezieht sich auf die Amazon-IVS-Konsole, API und Ressourcen (Stages).

Man könnte auch sagen, dass Amazon IVS ein „regionaler AWS-Service“ ist. Das heißt, Amazon IVS-Ressourcen sind in jeder Region unabhängig von ähnlichen Ressourcen in anderen Regionen. Beispielsweise ist ein Stage, den Sie in einer Region erstellen, unabhängig von Stages, die Sie in anderen Regionen erstellen.

Wenn Sie Ressourcen verwenden (z. B. einen Kanal erstellen), müssen Sie die Region angeben, in der er erstellt wird. Wenn Sie anschließend Ressourcen verwalten, müssen Sie dies von demselben Bereich aus tun, in dem sie erstellt wurden.

Bei Verwendung der ...	Sie geben die Region an, indem Sie...
Amazon IVS-Konsole	Verwendung von Auswählen einer Region oben rechts in der Navigatio nsleiste.
Amazon IVS-API	Verwenden des entsprechenden Service-Endpunktes. Sehen Sie die API-Referenz zu Amazon-IVS-Echtzeit-Streaming. (Wenn Sie über ein SDK auf die API zugreifen, richten Sie den `region`-Parameter des SDKs ein. Siehe Tools zum Entwickeln in AWS.)

Bei Verwendung der ...	Sie geben die Region an, indem Sie...
AWS-CLI	Entweder: • Anhängen von `--region <aws-region>` an Ihren CLI-Befehl. • Platzieren Sie die Region in Ihre lokale AWS-Konfigurationsdatei.

Denken Sie daran, dass Sie unabhängig von der Region, in der ein Stage erstellt wurde, von überall zu Amazon IVS streamen können und Zuschauer es von überall aus ansehen können.

Erste Schritte mit IVS-Echtzeit-Streaming

Dieses Dokument führt Sie durch die Schritte zur Integration von Amazon-IVS-Echtzeit-Streaming in Ihre Anwendung.

Themen

Einführung in IVS-Streaming in Echtzeit

In diesem Abschnitt werden die Voraussetzungen für die Verwendung von Streaming in Echtzeit aufgeführt und wichtige Begriffe vorgestellt.

Voraussetzungen

Bevor Sie Echtzeit-Streaming zum ersten Mal verwenden können, müssen Sie die folgenden Aufgaben erledigen. Anleitungen finden Sie unter Erste Schritte mit IVS-Streaming mit niedriger Latenz.

- Erstellen eines AWS-Kontos
- Richten Sie Root-Benutzer und Administratoren ein.

Andere Referenzen:

- Referenz zum IVS-Web-Broadcast-SDK
- Referenz zum IVS-Android-Broadcast-SDK
- Referenz zum IVS-iOS-Broadcast-SDK
- Referenz zur API von IVS-Echtzeit-Streaming

Terminologie für Echtzeit-Streaming

Begriff	Beschreibung
Stage	Ein virtueller Raum, in dem die Teilnehmer Videos in Echtzeit austauschen können.
Host	Ein Teilnehmer, der ein lokales Video auf die Stage sendet.
Zuschauer	Ein Teilnehmer, der ein Video der Hosts erhält.
Teilnehmer	Ein Benutzer, der als Host oder Zuschauer mit der Stage verbunden ist.
Teilnehmer-Token	Ein Token, das einen Teilnehmer authentifiziert, wenn er einer Stage beitritt.
Broadcast-SDK	Eine Clientbibliothek, die es den Teilnehmern ermöglich t, Videos zu senden und zu empfangen.

Übersicht über die Schritte

1. IAM-Berechtigungen einrichten – Erstellen Sie eine AWS-Richtlinie für Identity and Access Management (IAM), die Benutzern grundlegende Berechtigungen gewährt, und weisen Sie diese Richtlinie Benutzern zu.
2. Schaffen Sie eine Stage – Schaffen Sie eine virtuelle Umgebung, in der die Teilnehmer Videos in Echtzeit austauschen können.
3. Verteilen Sie Teilnehmer-Token – Senden Sie Tokens an die Teilnehmer, damit sie Ihrer Stage beitreten können.

4. Integrieren Sie das IVS-Broadcast-SDK – Fügen Sie das Broadcast-SDK zu Ihrer Anwendung hinzu, damit die Teilnehmer Videos senden und empfangen können: the section called “Web”, the section called “Android” und the section called “iOS”.
5. Video veröffentlichen und abonnieren – Senden Sie Ihr Video an die Stage und erhalten Sie Videos von anderen Hosts: IVS-Konsole, the section called “Web”, the section called “Android” uand the section called “iOS”.

Schritt 1: IAM-Berechtigungen einrichten

Als Nächstes müssen Sie eine AWS Identity and Access Management (IAM)-Richtlinie erstellen, die Benutzern einen grundlegenden Satz an Berechtigungen gewährt (z. B. zum Erstellen einer Amazon IVS-Stage und zum Erstellen von Teilnehmer-Tokens) und diese Richtlinie den Benutzern zuweisen. Die Berechtigungen können Sie beim Erstellen eines neuen Benutzers zuweisen oder einem vorhandenen Benutzer hinzufügen. Im Folgenden sind beide Verfahren angegeben.

Weitere Informationen (z. B. Informationen zu IAM-Benutzern und -Richtlinien, zum Anhängen einer Richtlinie an einen Benutzer und zum Beschränken der Möglichkeiten von Benutzern mit Amazon IVS) finden Sie unter:

- Erstellen eines IAM-Benutzers im IAM-Benutzerhandbuch
- Die Informationen in Amazon-IVS-Sicherheit zu IAM und „Verwaltete Richtlinien für IVS“.
- Die IAM-Informationen in Amazon-IVS-Sicherheit

Sie können entweder eine vorhandene von AWS verwaltete Richtlinie für Amazon IVS verwenden oder eine neue Richtlinie erstellen, die die Berechtigungen anpasst, die Sie einer Reihe von Benutzern, Gruppen oder Rollen gewähren möchten. Beide Vorgehensweisen werden nachfolgend beschrieben.

Verwenden einer vorhandenen Richtlinie für IVS-Berechtigungen

In den meisten Fällen möchten Sie für Amazon IVS eine von AWS verwaltete Richtlinie verwenden. Diese werden ausführlich im Abschnitt Verwaltete Richtlinien für IVS von IVS-Sicherheit beschrieben.

- Mit der von AWS verwalteten Richtlinie `IVSReadOnlyAccess` gewähren Sie den Anwendungsentwicklern Zugriff auf alle Get- und List-API-Vorgänge von IVS (sowohl für Streaming mit niedriger Latenz als auch für Echtzeit-Streaming).

- Mit der von AWS verwalteten Richtlinie `IVSFullAccess` gewähren Sie den Anwendungsentwicklern Zugriff auf alle API-Vorgänge von IVS (sowohl für Streaming mit niedriger Latenz als auch für Echtzeit-Streaming).

Optional: Eine benutzerdefinierte Richtlinie für Amazon-IVS-Berechtigungen erstellen

Dazu gehen Sie wie folgt vor:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Policies und dann Create policy. Das Fenster Berechtigungen angeben wird geöffnet.
3. Wählen Sie im Fenster Berechtigungen angeben die Registerkarte JSON. Kopieren Sie die folgende IVS-Richtlinie und fügen Sie sie in den Textbereich Richtlinien-Editor ein. (Die Richtlinie umfasst nicht alle Aktionen von Amazon IVS. Je nach Bedarf können Sie Zugriffsberechtigungen für Vorgänge hinzufügen/löschen (Zulassen/Verweigern). Einzelheiten zu den IVS-Vorgängen finden Sie in der Referenz zur API von IVS-Echtzeit-Streaming.)

   JSON

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

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

4. Wählen Sie im Fenster Berechtigungen angeben die Option Weiter aus (scrollen Sie zum unteren Ende des Fensters, um diese Option anzuzeigen). Das Fenster Überprüfen und erstellen wird geöffnet.
5. Geben Sie im Fenster Überprüfen und erstellen einen Richtliniennamen ein und fügen Sie optional eine Beschreibung hinzu. Notieren Sie sich den Namen der Richtlinie, da Sie ihn beim Erstellen von Benutzern (weiter unten) benötigen. Wählen Sie Create policy (Richtlinie erstellen) aus (am unteren Ende des Fensters).
6. Sie gelangen zurück zum IAM-Konsolenfenster, in dem per Banner bestätigt werden sollte, dass die neue Richtlinie erstellt wurde.

Erstellen Sie einen neuen Benutzer und fügen Sie Berechtigungen hinzu

IAM-Benutzer-Zugriffsschlüssel

IAM-Zugriffsschlüssel bestehen aus einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel. Sie dienen zum Signieren Ihrer programmgesteuerten Anforderungen an AWS. Wenn Sie noch

keine Zugriffsschlüssel besitzen, können Sie diese über die AWS-Managementkonsole erstellen. Verwenden Sie als bewährte Methode keine Zugriffsschlüssel für Root-Benutzer.

Einen geheimen Zugriffsschlüssel können Sie nur beim Erstellen von Zugriffsschlüsseln anzeigen oder herunterladen. Später kann er nicht mehr wiederhergestellt werden. Sie können jedoch jederzeit neue Zugriffsschlüssel erstellen. Dazu benötigen Sie die Berechtigungen zum Ausführen der erforderlichen IAM-Aktionen.

Bewahren Sie Zugriffsschlüssel stets sicher auf. Geben Sie sie niemals an Dritte weiter (selbst wenn eine Anfrage von Amazon zu stammen scheint). Weitere Informationen finden Sie unter Verwalten der Zugriffsschlüssel für IAM-Benutzer im IAM-Benutzerhandbuch.

Verfahren

Dazu gehen Sie wie folgt vor:

1. Wählen Sie im Navigationsbereich die Option Benutzer und dann Benutzer erstellen. Das Fenster Benutzerdetails angeben wird geöffnet.
2. Gehen Sie im Fenster Benutzerdetails angeben wie folgt vor:
   a. Geben Sie unter Benutzerdetails den neuen Benutzernamen ein, der erstellt werden soll.
   b. Aktivieren Sie Benutzerzugriff auf AWS-Managementkonsole bereitstellen.
   c. Wählen Sie unter Console password (Konsolenpasswort) Autogenerated password (Automatisch generiertes Passwort).
   d. Aktivieren Sie Benutzer muss bei der nächsten Anmeldung ein neues Passwort erstellen.
   e. Wählen Sie Weiter aus. Das Fenster Berechtigungen festlegen wird geöffnet.
3. Wählen Sie unter Berechtigungen festlegen die Option Richtlinien direkt zuweisen aus. Das Fenster Berechtigungsrichtlinien wird geöffnet.
4. Geben Sie im Suchfeld einen IVS-Richtliniennamen ein (entweder eine von AWS verwaltete Richtlinie oder Ihre zuvor erstellte benutzerdefinierte Richtlinie). Wenn sie gefunden wurde, aktivieren Sie das Kästchen, um die Richtlinie auszuwählen.
5. Wählen Sie Weiter (unten im Fenster). Das Fenster Überprüfen und erstellen wird geöffnet.
6. Vergewissern Sie sich im Fenster Überprüfen und erstellen, ob alle Benutzerdetails korrekt sind, und wählen Sie dann Benutzer erstellen aus (unten im Fenster).
7. Das Fenster Passwort abrufen wird geöffnet, das Ihre Details zur Anmeldung an der Konsole enthält. Bewahren Sie diese Informationen sicher auf, damit Sie in Zukunft darauf zurückgreifen können. Klicken Sie abschließend auf Zurück zur Benutzerliste.

Hinzufügen von Berechtigungen zu einem vorhandenen Benutzer

Dazu gehen Sie wie folgt vor:

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Benutzer aus und wählen Sie dann einen vorhandenen Benutzernamen aus, der aktualisiert werden soll. (Wählen Sie den Namen aus, indem Sie darauf klicken. Aktivieren Sie nicht das Auswahlfeld.)
3. Wählen Sie auf der Seite Zusammenfassung in der Registerkarte Berechtigungen die Option Berechtigungen hinzufügen aus. Das Fenster Berechtigungen hinzufügen wird geöffnet.
4. Wählen Sie die Option Attach existing policies directly (Vorhandene Richtlinien direkt anfügen) aus. Das Fenster Berechtigungsrichtlinien wird geöffnet.
5. Geben Sie im Suchfeld einen IVS-Richtliniennamen ein (entweder eine von AWS verwaltete Richtlinie oder Ihre zuvor erstellte benutzerdefinierte Richtlinie). Wenn die Richtlinie gefunden wurde, aktivieren Sie das Kästchen, um die Richtlinie auszuwählen.
6. Wählen Sie Weiter (unten im Fenster). Das Fenster Überprüfung wird geöffnet.
7. Wählen Sie im Fenster Überprüfung die Option Berechtigungen hinzufügen aus (unten im Fenster).
8. Vergewissern Sie sich auf der Seite Summary (Zusammenfassung), dass die IVS-Richtlinie hinzugefügt wurde.

Schritt 2: Erstellen einer Stage mit optionaler Teilnehmeraufzeichnung

Bei einer Stage handelt es sich um eine virtuellen Umgebung, in der die Teilnehmer Audio und Video in Echtzeit austauschen können. Sie ist die grundlegende Ressource der Echtzeit-Streaming-API. Sie können eine Stage entweder über die Konsole oder den Vorgang CreateStage erstellen.

Wir empfehlen, dass Sie nach Möglichkeit für jede logische Sitzung eine neue Stage erstellen und diese löschen, wenn Sie fertig sind, anstatt alte Stages für eine mögliche Wiederverwendung beizubehalten. Wenn veraltete Ressourcen (alte Stages, die nicht wiederverwendet werden sollen) nicht bereinigt werden, erreichen Sie wahrscheinlich schneller das Limit der maximalen Anzahl an Stages.

Stage – mit oder ohne Aufzeichnung einzelner Teilnehmer – können Sie über die Amazon-IVS-Konsole oder die AWS-CLI erstellen. Die Erstellung und Aufzeichnung von Stage werden nachfolgend erörtert.

Aufzeichnung einzelner Teilnehmer

Sie haben die Möglichkeit, die Aufzeichnung einzelner Teilnehmer für eine Stage zu aktivieren. Wenn das Feature zur Aufzeichnung einzelner Teilnehmer in S3 aktiviert ist, werden alle Broadcasts einzelner Teilnehmer an die Stage aufgezeichnet und in einem Ihnen gehörenden Amazon-S3-Speicher-Bucket gespeichert. Anschließend ist die Aufnahme für die On-Demand-Wiedergabe verfügbar.

Das Einrichten hiervon ist eine erweiterte Option. Beim Erstellen einer Stage ist die Aufzeichnung standardmäßig deaktiviert.

Ehe Sie eine Stage für die Aufzeichnung einrichten können, müssen Sie eine Speicherkonfiguration erstellen. Dabei handelt es sich um eine Ressource, die einen Amazon-S3-Speicherort angibt, an dem die aufgezeichneten Streams für die Stage gespeichert werden. Speicherkonfiguration können Sie über die Konsole oder CLI erstellen und verwalten. Beide Verfahren sind unten aufgeführt. Nachdem Sie die Speicherkonfiguration erstellt haben, ordnen Sie sie einer Stage zu, entweder beim Erstellen der Stage (wie unten beschrieben) oder später durch Aktualisierung einer vorhandenen Stage. (Siehe CreateStage und UpdateStage in der API.) Sie können mehrere Stages derselben Speicherkonfiguration zuordnen. Speicherkonfigurationen, die keiner Stage mehr zugeordnet sind, können gelöscht werden.

Beachten Sie folgende Einschränkungen:

- Sie müssen Eigentümer des S3-Buckets sein. Das heißt, der S3-Bucket, in dem die Aufzeichnungen gespeichert werden, muss dem Konto gehören, über das eine aufzuzeichnende Stage eingerichtet wird.
- Die Stage, die Speicherkonfiguration und der S3-Speicher-Bucket müssen sich in derselben AWS-Region befinden. Wenn Sie Stages in anderen Regionen erstellen und diese aufzeichnen möchten, müssen Sie in diesen Regionen ebenfalls Speicherkonfigurationen und S3-Buckets einrichten.

Die Aufnahme in Ihrem S3-Bucket erfordert eine Autorisierung mit Ihren AWS-Anmeldeinformationen. Um IVS den erforderlichen Zugriff zu gewähren, wird beim Erstellen der Aufzeichnungskonfiguration automatisch eine serviceverknüpfte Rolle von AWS IAM erstellt. Diese Rolle erteilt IVS nur die Schreibberechtigung für den jeweiligen Bucket.

Beachten Sie, dass Netzwerkprobleme zwischen dem Streaming-Standort und AWS oder in AWS zu Datenverlusten bei der Aufzeichnung von Streams führen können. In diesen Fällen priorisiert Amazon IVS den Livestream gegenüber der Aufzeichnung. Für Redundanz nehmen Sie lokal über Ihr Streaming-Tool auf.

Weitere Informationen (darunter zum Einrichten der Nachbearbeitung oder der VOD-Wiedergabe für aufgezeichnete Dateien) finden Sie unter Aufzeichnung einzelner Teilnehmer.

Aufnahme deaktivieren

So deaktivieren Sie die Amazon-S3-Aufzeichnung für eine vorhandene Stage:

- Konsole – Deaktivieren Sie auf der Detailseite der entsprechenden Stage im Streams-Abschnitt Einzelne Teilnehmer aufzeichnen unter Automatisch in S3 aufzeichnen die Option Automatische Aufzeichnung aktivieren. Wählen Sie dann Änderungen speichern. Dadurch wird die Zuordnung der Speicherkonfiguration zur Stage entfernt. Streams der betreffenden Stage werden dann nicht mehr aufgezeichnet.
- CLI – Führen Sie den Befehl `update-stage` aus und übergeben Sie den ARN für die Aufzeichnungskonfiguration als leere Zeichenfolge:

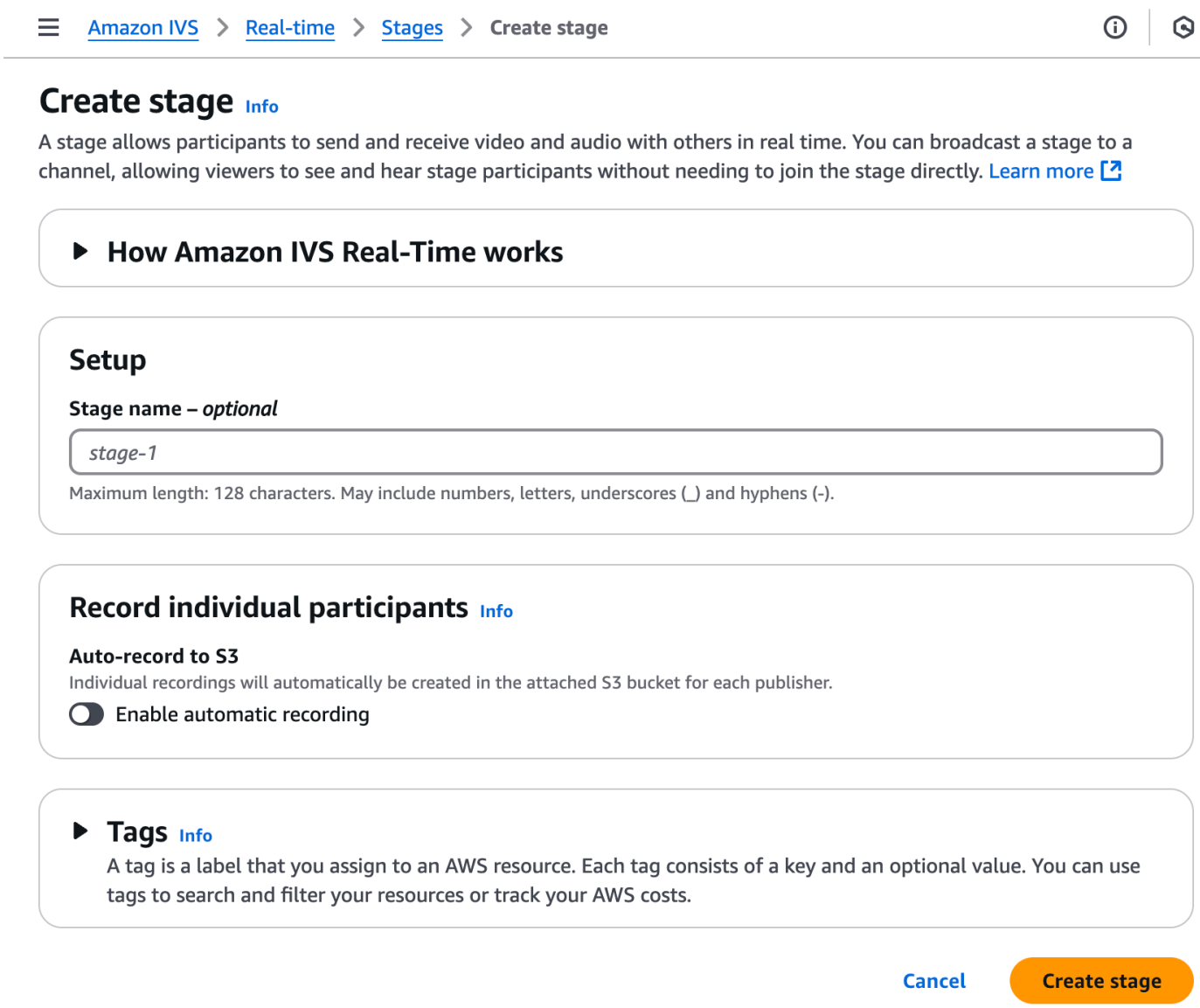
  ```
  aws ivs-realtime update-stage --arn arn:aws:ivs:us-west-2:123456789012:stage/
  abcdABCDefgh --auto-participant-recording-configuration storageConfigurationArn=""
  ```

  Hierdurch wird ein Stageobjekt mit einer leeren Zeichenfolge für `storageConfigurationArn` zurückgegeben, was anzeigt, dass die Aufzeichnung deaktiviert ist.

Anleitung zum Erstellen einer IVS-Stage über die Konsole

1. Öffnen Sie die Amazon-IVS-Konsole.

   (Sie können auf die Amazon-IVS-Konsole auch über die AWS-Managementkonsole zugreifen.)
2. Wählen Sie im linken Navigationsbereich Stage und dann Stage erstellen aus. Das Fenster Stage erstellen wird angezeigt.

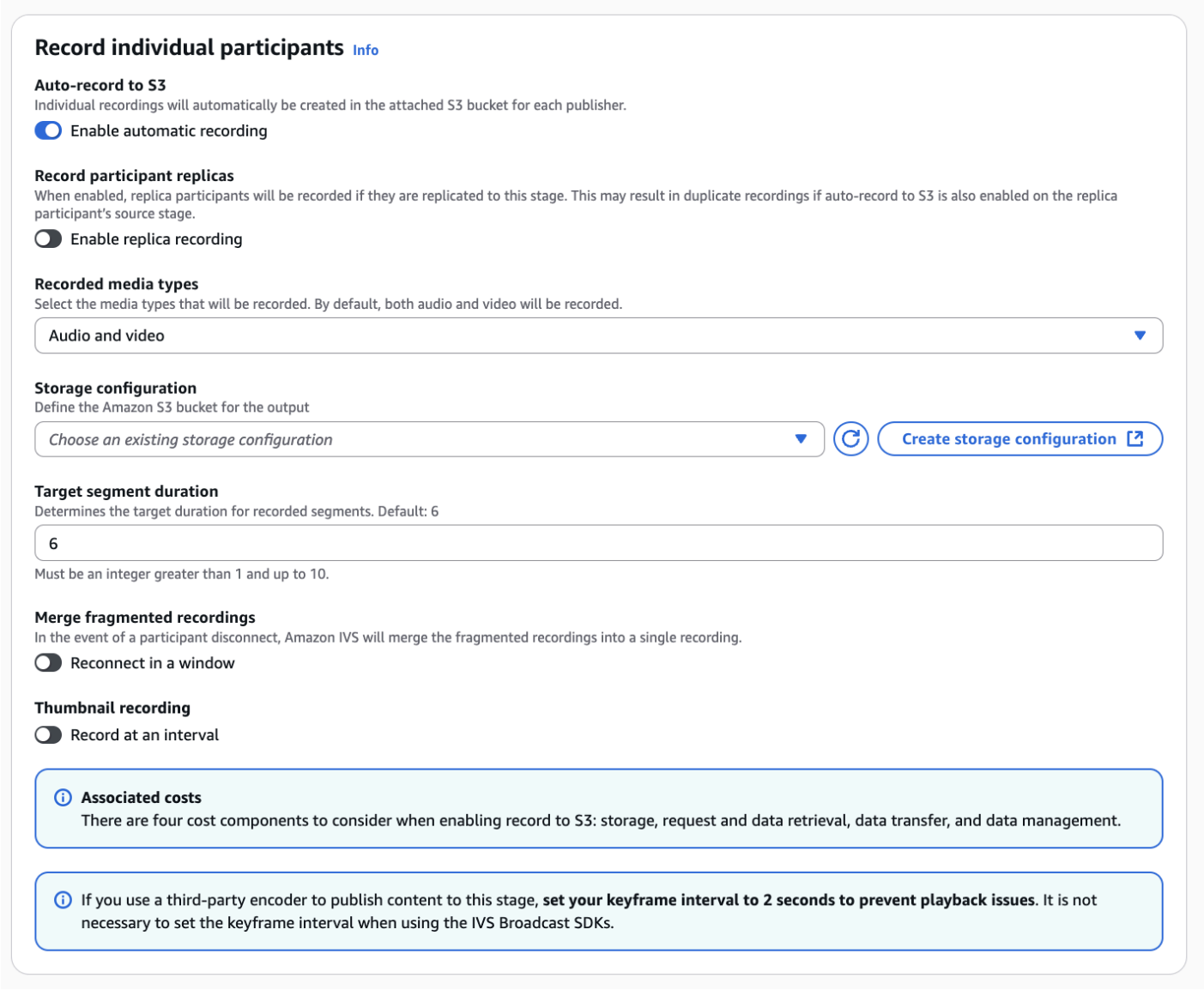


3. Geben Sie optional einen Stagenamen ein.
4. Wenn Sie die Aufzeichnung einzelner Teilnehmer aktivieren möchten, führen Sie die nachfolgenden Schritte unter Einrichten der automatischen Aufzeichnung einzelner Teilnehmer in Amazon S3 (optional) durch.
5. Wählen Sie Stage erstellen aus, um die Stage zu erstellen. Die Seite mit den Stagedetails für die neue Stage wird angezeigt.

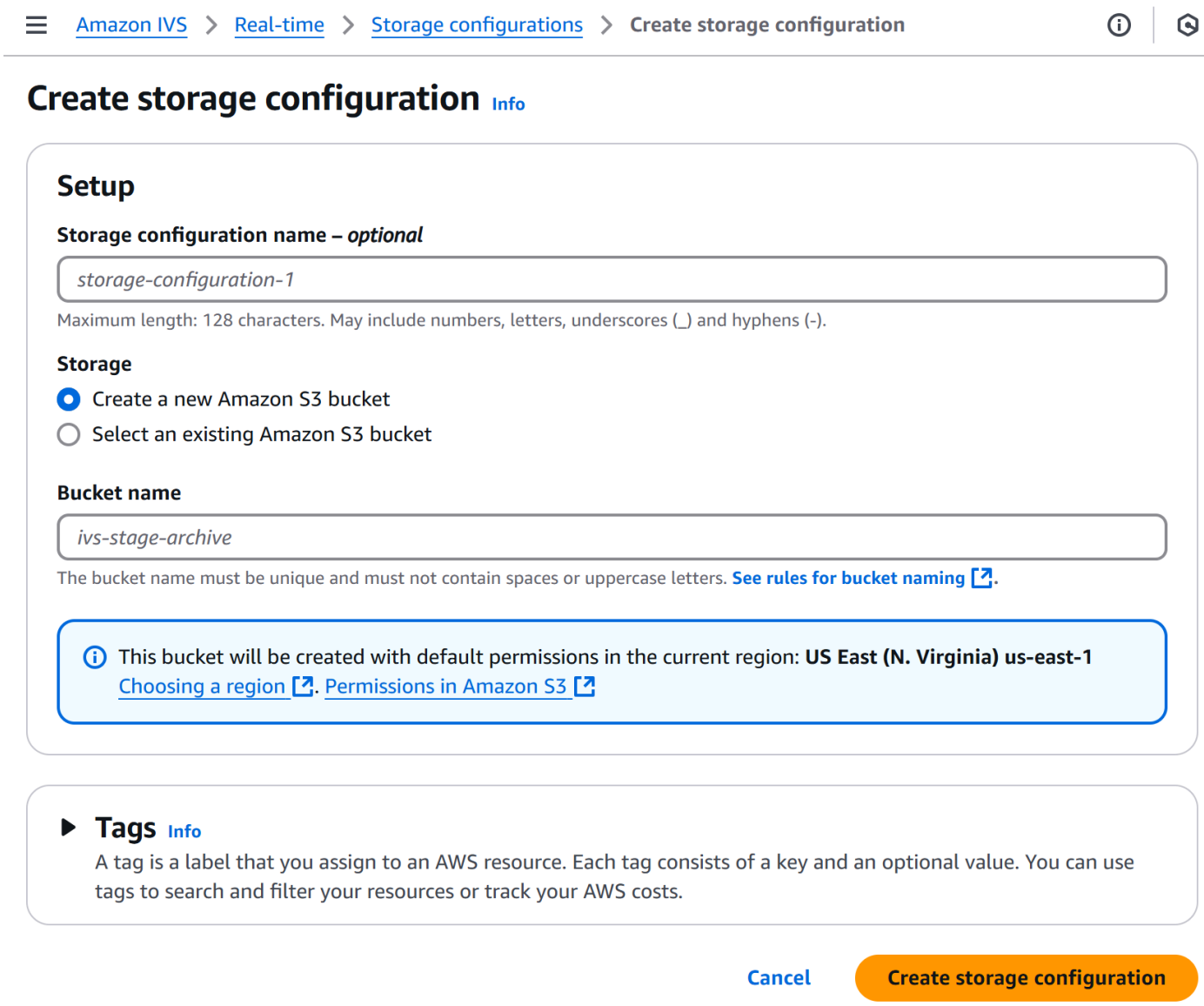
Einrichten der automatischen Aufzeichnung einzelner Teilnehmer in Amazon S3 (optional)

Gehen Sie wie folgt vor, um die Aufzeichnung beim Erstellen einer neuen Stage zu aktivieren:

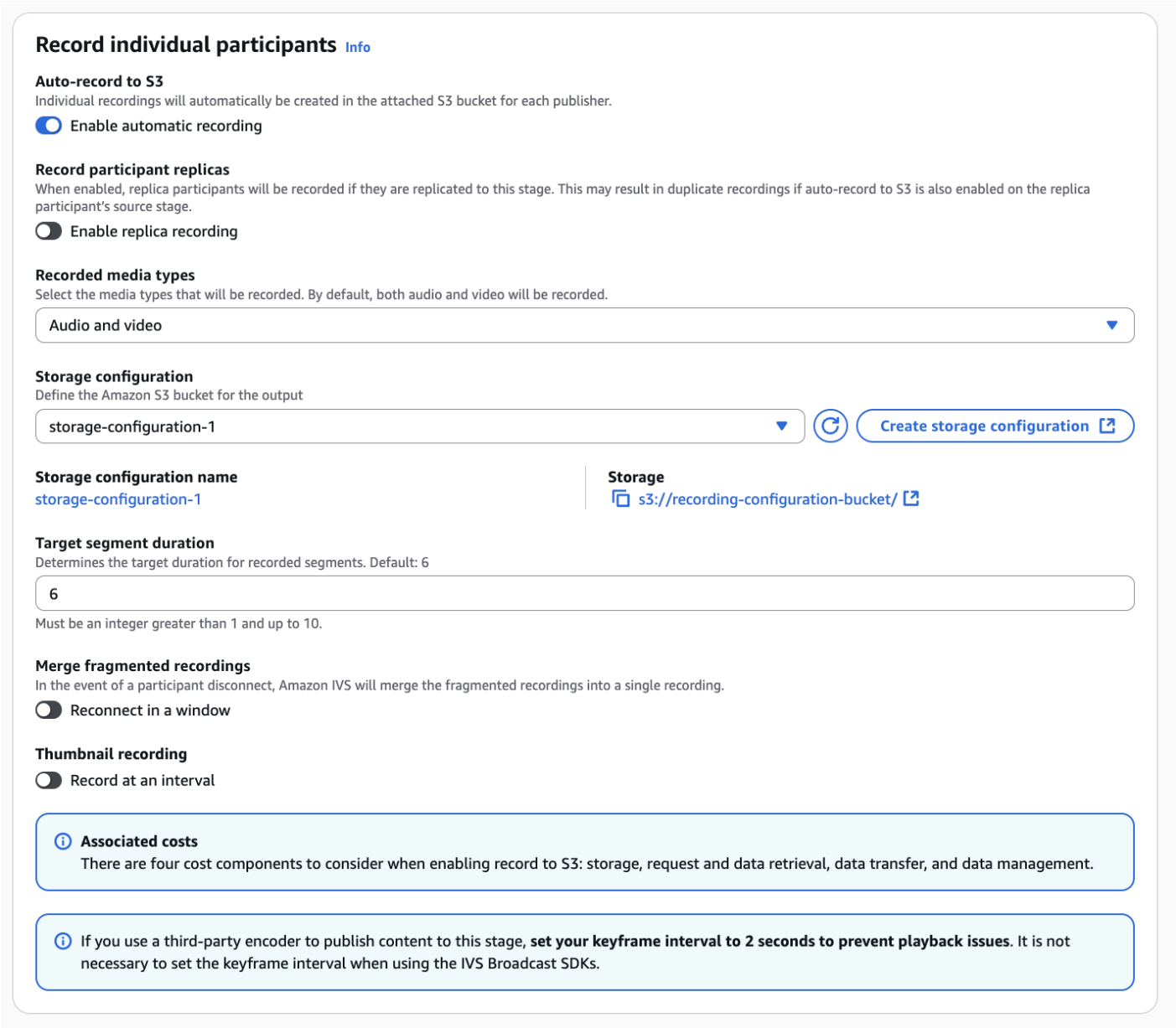
1. Aktivieren Sie auf der Seite Stage erstellen unter Einzelne Teilnehmer aufzeichnen die Option Automatische Aufzeichnung aktivieren. Daraufhin werden zusätzliche Felder angezeigt, in denen Sie die aufgezeichneten Medientypen auswählen, eine vorhandene Speicherkonfiguration auswählen oder eine neue Konfiguration erstellen können. Außerdem können Sie dort auswählen, ob in einem bestimmten Intervall Miniaturansichten aufgezeichnet werden sollen.



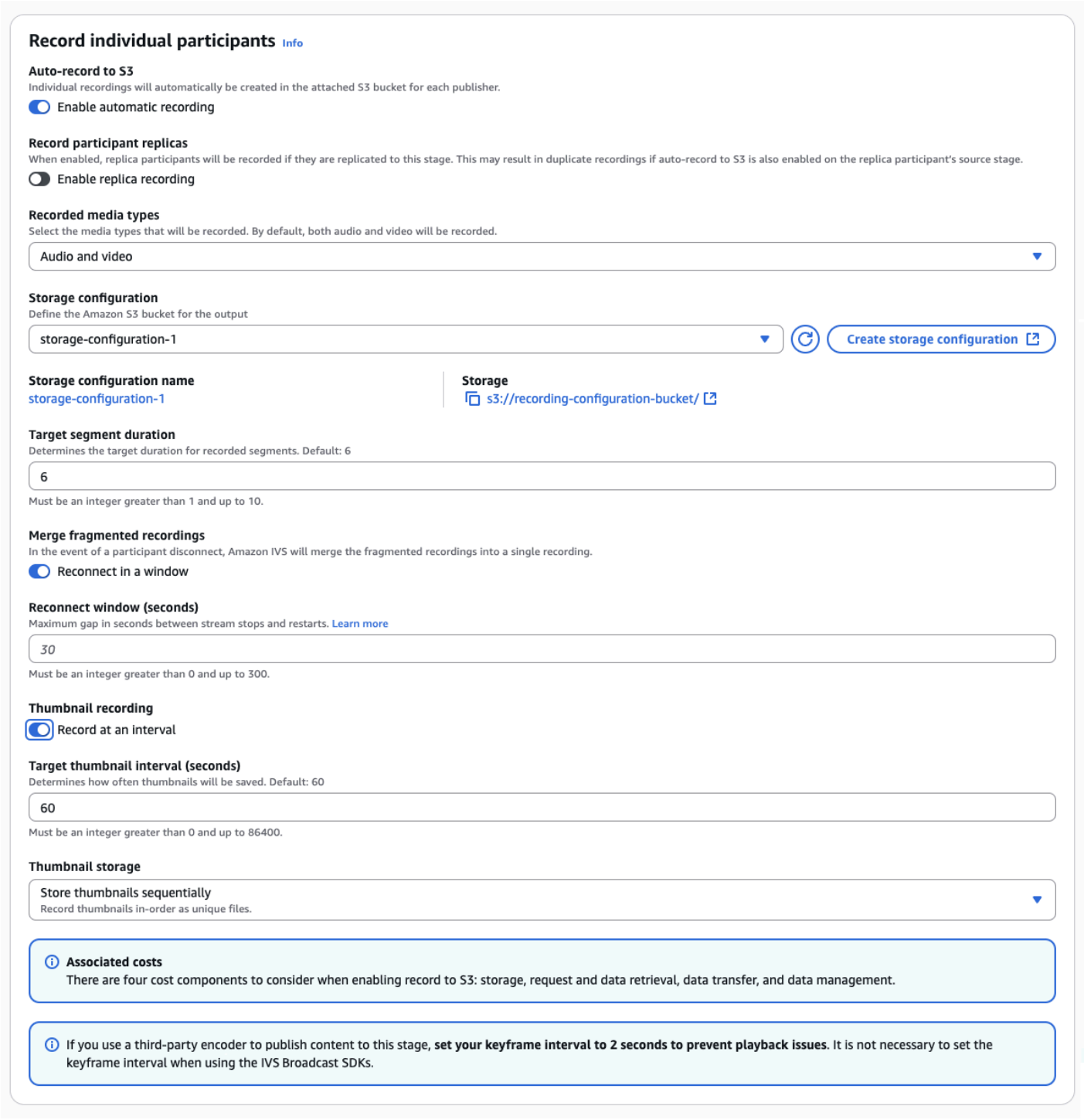
2. Wählen Sie aus, welche Medientypen aufgezeichnet werden sollen.
3. Wählen Sie Speicherkonfiguration erstellen. Ein neues Fenster wird mit Optionen geöffnet, um einen Amazon S3 Bucket zu erstellen und ihn an die neue Aufzeichnungskonfiguration anzuhängen.



4. Füllen Sie die Felder aus:
   a. Geben Sie optional einen Speicherkonfigurationsnamen ein.
   b. Geben Sie einen Bucket name (Bucket-Namen) ein.
5. Wählen Sie Speicherkonfiguration erstellen, um eine neue Speicherkonfiguration mit einem eindeutigen ARN zu erstellen. In der Regel dauert die Erstellung der Aufnahmekonfiguration einige Sekunden, aber es kann bis zu 20 Sekunden dauern. Wenn die Speicherkonfiguration erstellt ist, kehren Sie zum Fenster Stage erstellen zurück. Dort werden im Bereich Einzelne Teilnehmer aufzeichnen die neue Speicherkonfiguration und der von Ihnen erstellte S3-Bucket (Speicher) angezeigt.



6. Optional können Sie weitere Optionen aktivieren, die nicht dem Standard entsprechen, z. B. die Aufzeichnung von Teilnehmerreplikaten, die Zusammenführung der Aufzeichnung einzelner Teilnehmer und die Aufzeichnung von Miniaturansichten.



Anleitung zum Erstellen einer IVS-Stage über die CLI

Informationen zum Installieren der AWS-CLI finden Sie unter Installieren oder Aktualisieren auf die neueste Version der AWS-CLI.

Ab sofort können Sie Ressourcen über die CLI erstellen und verwalten. Dazu wenden Sie je nachdem, ob Sie eine Stage mit oder ohne aktivierte Aufzeichnung einzelner Teilnehmer erstellen möchten, eines der beiden folgenden Verfahren an.

Erstellen einer Stage ohne Aufzeichnung einzelner Teilnehmer

Die Stage-API befindet sich unter dem Namespace ivs-realtime. Beispiel zum Erstellen einer Stage:

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

Die Antwort ist:

```
{
   "stage": {
      "arn": "arn:aws:ivs:us-west-2:376666121854:stage/VSWjvX5XOkU3",
      "name": "test-stage"
   }
}
```

Erstellen einer Stage mit Aufzeichnung einzelner Teilnehmer

So erstellen Sie eine Stage mit aktivierter Aufzeichnung einzelner Teilnehmer:

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

Übergeben Sie optional den Parameter `thumbnailConfiguration`, um den Speicher, den Modus und das Intervall (in Sekunden) für die Aufzeichnung von Miniaturansichten manuell einzustellen:

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

Übergeben Sie optional den Parameter `recordingReconnectWindowSeconds`, um das Zusammenführen fragmentierter Aufzeichnungen einzelner Teilnehmer zu aktivieren:

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

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

Die Antwort ist:

```
{
   "stage": {
      "arn": "arn:aws:ivs:us-west-2:123456789012:stage/VSWjvX5XOkU3",
      "autoParticipantRecordingConfiguration": {
         "hlsConfiguration": {
             "targetSegmentDurationSeconds": 6
         },
         "mediaTypes": [
            "AUDIO_VIDEO"
         ],
         "recordingReconnectWindowSeconds": 60,
         "recordParticipantReplicas": true,
         "storageConfigurationArn": "arn:aws:ivs:us-west-2:123456789012:storage-
configuration/LKZ6QR7r55c2",
         "thumbnailConfiguration": {
            "recordingMode": "INTERVAL",
            "storage": [
               "SEQUENTIAL",
               "LATEST"
            ],
            "targetIntervalSeconds": 10
         }
      },
      "endpoints": {
         "events": "<events-endpoint>",
         "rtmp": "<rtmp-endpoint>",
         "rtmps": "<rtmps-endpoint>",
         "whip": "<whip-endpoint>"
      },
      "name": "test-stage-participant-recording"
   }
}
```

Schritt 3: Teilnehmertoken verteilen

Jetzt, da Sie eine Stage haben, müssen Sie Token erstellen und an die Teilnehmer verteilen, damit sie der Stage beitreten und mit dem Senden und Empfangen von Videos beginnen können. Für die Generierung von Token gibt es zwei Ansätze:

- Erstellen von Token mit einem Schlüsselpaar
- Erstellen von Token mit der IVS-Echtzeit-Streaming-API.

Beide Vorgehensweisen werden nachfolgend beschrieben.

Erstellen von Token mit einem Schlüsselpaar

Sie können Token in Ihrer Serveranwendung erstellen und sie an die Teilnehmer verteilen, damit diese an einer Stage teilnehmen können. Sie müssen ein öffentliches/privates ECDSA-Schlüsselpaar generieren, um die JWTs zu signieren und den öffentlichen Schlüssel in IVS zu importieren. Dann kann IVS die Token zum Zeitpunkt des Beitritts zur Stage verifizieren.

Bei IVS gibt es kein Ablaufdatum für Schlüssel. Wenn Ihr privater Schlüssel gefährdet ist, müssen Sie den alten öffentlichen Schlüssel löschen.

Erstellen eines neuen Schlüsselpaares

Es gibt verschiedene Methoden, um ein Schlüsselpaar zu erstellen. Im Folgenden geben wir zwei Beispiele.

Führen Sie die folgenden Schritte aus, um in der Konsole ein neues Schlüsselpaar zu erstellen:

1. Öffnen Sie die Amazon-IVS-Konsole. Wählen Sie die Region Ihrer Stage aus, wenn Sie sich nicht bereits darin befinden.
2. Wählen Sie im linken Navigationsmenü Echtzeit-Streaming > Öffentliche Schlüssel aus.
3. Wählen Sie Öffentlichen Schlüssel erstellen. Das Dialogfenster Öffentlichen Schlüssel erstellen wird angezeigt.
4. Folgen Sie den Eingabeaufforderungen und wählen Sie Erstellen.
5. Amazon IVS generiert ein neues Schlüsselpaar. Der öffentliche Schlüssel wird als öffentliche Schlüsselressource importiert und der private Schlüssel wird sofort zum Download zur Verfügung gestellt. Der öffentliche Schlüssel kann bei Bedarf auch später heruntergeladen werden.

Amazon IVS generiert den Schlüssel auf der Clientseite und speichert den privaten Schlüssel nicht. Speichern Sie den Schlüssel unbedingt. Sie können ihn später nicht abrufen.

Um ein neues P384-EC-Schlüsselpaar mit OpenSSL zu erstellen, gehen Sie wie folgt vor (möglicherweise müssen Sie dazu zunächst OpenSSL installieren). Mit diesem Verfahren können Sie sowohl auf die privaten als auch auf die öffentlichen Schlüssel zugreifen. Den öffentlichen Schlüssel benötigen Sie nur, wenn Sie die Verifizierung Ihrer Token testen möchten.

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

Importieren Sie nun den neuen öffentlichen Schlüssel. Nutzen Sie dazu die untenstehende Anleitung.

Importieren des öffentlichen Schlüssels

Sobald Sie über ein Schlüsselpaar verfügen, können Sie den öffentlichen Schlüssel in IVS importieren. Der private Schlüssel wird von unserem System nicht benötigt; er wird von Ihnen zum Signieren von Token verwendet.

So importieren Sie einen vorhandenen öffentlichen Schlüssel mit der Konsole:

1. Öffnen Sie die Amazon-IVS-Konsole. Wählen Sie die Region Ihrer Stage aus, wenn Sie sich nicht bereits darin befinden.
2. Wählen Sie im linken Navigationsmenü Echtzeit-Streaming > Öffentliche Schlüssel aus.
3. Wählen Sie Importieren aus. Das Dialogfenster Öffentlichen Schlüssel importieren wird angezeigt.
4. Folgen Sie den Eingabeaufforderungen und wählen Sie Importieren.
5. Amazon IVS importiert Ihren öffentlichen Schlüssel und generiert eine öffentliche Schlüsselressource.

So importieren Sie einen vorhandenen öffentlichen Schlüssel mit der CLI:

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

Sie können `--region <aws-region>` auslassen, wenn sich die Region in Ihrer lokalen AWS Konfigurationsdatei befindet.

Hier ist ein Beispiel für eine Antwort:

```
{
    "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
\nS6vRV9Wx37mjwi02hObKuCJqpj7x0lpz0bHm5v1JBvdZYAd/r2LR5aChK+/GM2Wj
\nl8MG9NJIVFaw1u3bvjEjzTASSfS1BDX1\n-----END PUBLIC KEY-----\n",
        "tags": {}
    }
}
```

API-Anforderungen

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

Generieren und Signieren des Tokens

Weitere Informationen zum Arbeiten mit JWTs und den unterstützten Bibliotheken zum Signieren von Token finden Sie unter jwt.io. In der jwt.io-Oberfläche müssen Sie Ihren privaten Schlüssel eingeben, um Token zu signieren. Der öffentliche Schlüssel wird nur benötigt, wenn Sie Token verifizieren möchten.

Alle JWTs haben drei Felder: Header, Nutzlast und Signatur.

Die JSON-Schemas für den Header und die Nutzdaten des JWT werden unten beschrieben. Alternativ können Sie ein Beispiel-JSON von der IVS-Konsole kopieren. Abrufen des Headers und der Nutzdaten-JSON von der IVS-Konsole:

1. Öffnen Sie die Amazon-IVS-Konsole. Wählen Sie die Region Ihrer Stage aus, wenn Sie sich nicht bereits darin befinden.
2. Wählen Sie im linken Navigationsbereich Echtzeit-Streaming > Stage aus.
3. Wählen Sie die Stage aus, die Sie verwenden möchten. Wählen Sie Details anzeigen aus.

4. Wählen Sie im Abschnitt Teilnehmer-Token das Drop-down-Menü neben Token erstellen aus.
5. Wählen Sie Token-Header und Nutzdaten erstellen aus.
6. Füllen Sie das Formular aus und kopieren Sie den JWT-Header und die Nutzdaten, die unten im Popup angezeigt werden.

Token-Schema: Header

Der Header gibt Folgendes an:

- `alg` ist der Signaturalgorithmus. Dies ist ES384, ein ECDSA-Signaturalgorithmus, der den SHA-384-Hash-Algorithmus verwendet.
- `typ` ist der Tokentyp, JWT.
- `kid` ist der ARN des öffentlichen Schlüssels, der zum Signieren des Tokens verwendet wird. Es muss derselbe ARN sein, der von der API-Anforderung GetPublicKey zurückgegeben wurde.

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

Token-Schema: Nutzdaten

Die Nutzdaten enthalten spezifische Daten für IVS. Alle Felder außer `user_id` sind Pflichtfelder.

- `RegisteredClaims` in der JWT-Spezifikation sind reservierte Ansprüche, die angegeben werden müssen, damit das Stage-Token gültig ist:
  - `exp` (Ablaufzeit) ist ein Unix-UTC-Zeitstempel für den Zeitpunkt, an dem das Token abläuft. (Ein Unix-Zeitstempel ist ein numerischer Wert, der die Anzahl der Sekunden von 1970-01-01T00:00:00Z UTC bis zum angegebenen UTC-Datum/Uhrzeit angibt, wobei Schaltsekunden ignoriert werden.) Das Token wird validiert, wenn der Teilnehmer einer Stage beitritt. IVS stellt Token standardmäßig mit einer Gültigkeitsdauer von 12 Stunden bereit, die wir empfehlen. Diese Frist kann auf maximal 14 Tage ab dem Zeitpunkt der Ausstellung (iat) verlängert werden. Dieser Wert muss eine Ganzzahl sein.
  - `iat` (Ausstellungszeit) ist ein Unix-UTC-Zeitstempel für den Zeitpunkt, an dem das JWT ausgestellt wurde. (Informationen zu Unix-Zeitstempeln finden Sie in der Anmerkung für `exp`.) Dieser Wert muss eine Ganzzahl sein.

  - `jti` (JWT-ID) ist die Teilnehmer-ID, die zur Nachverfolgung verwendet wird und auf den Teilnehmer verweist, dem das Token gewährt wird. Jedes Token muss eine eindeutige Teilnehmer-ID haben. Sie muss eine Zeichenfolge sein, die Groß- und Kleinschreibung berücksichtigt, bis zu 64 Zeichen lang ist und nur alphanumerische Zeichen, Bindestrich (-) und Unterstrich (_) enthält. Andere Sonderzeichen sind nicht zulässig.
- `user_id` ist ein optionaler, vom Kunden zugewiesener Name, der die Identifizierung des Tokens erleichtert. Dieser kann verwendet werden, um einen Teilnehmer mit einem Benutzer in den eigenen Systemen des Kunden zu verknüpfen. Dieser Wert muss mit dem Feld `userId` in der API-Anforderung CreateParticipantToken übereinstimmen. Er kann jeder UTF-8-codierte Text sein und ist eine Zeichenfolge mit bis zu 128 Zeichen. Dieses Feld ist für alle Stageteilnehmer sichtbar und darf daher nicht für personenbezogene, vertrauliche oder sensible Informationen verwendet werden.
- `resource` ist der ARN der Stage, z. B. `arn:aws:ivs:us-east-1:123456789012:stage/oRmLNwuCeMlQ`.
- `topic` ist die ID der Stage, die aus dem Stage-ARN extrahiert werden kann. Wenn der Stage-ARN beispielsweise `arn:aws:ivs:us-east-1:123456789012:stage/oRmLNwuCeMlQ` lautet, ist die Stage-ID `oRmLNwuCeMlQ`.
- `events_url` muss der Endpunkt für Ereignisse sein, der vom CreateStage- oder GetStage-Vorgang zurückgegeben wird. Es wird empfohlen, diesen Wert bei der Stageerstellung zwischenzuspeichern. Er kann bis zu 14 Tage lang zwischengespeichert werden. Ein Beispielwert ist `wss://global.events.live-video.net`.
- `whip_url` muss der WHIP-Endpunkt sein, der vom CreateStage- oder GetStage-Vorgang zurückgegeben wird. Es wird empfohlen, diesen Wert bei der Stageerstellung zwischenzuspeichern. Er kann bis zu 14 Tage lang zwischengespeichert werden. Ein Beispielwert ist `https://453fdfd2ad24df.global-bm.whip.live-video.net`.
- `capabilities` gibt die Fähigkeiten des Tokens an. Gültige Werte sind `allow_publish` und `allow_subscribe`. Für Token, die nur für Abonnements bestimmt sind, legen Sie `allow_subscribe` auf `true` fest.
- `attributes` ist ein optionales Feld, in dem Sie von der Anwendung bereitgestellte Attribute angeben können, die in das Token codiert und an eine Stage angehängt werden sollen. Zuordnungsschlüssel und -werte können UTF-8-codierten Text enthalten. Die maximale Länge dieses Felds beträgt insgesamt 1 KB. Dieses Feld ist für alle Stageteilnehmer sichtbar und darf daher nicht für personenbezogene, vertrauliche oder sensible Informationen verwendet werden.
- `version` muss sein `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"
}
```

Token-Schema: Signatur

Zum Erstellen der Signatur verwenden Sie den privaten Schlüssel im Header (ES384) mit dem angegebenen Algorithmus, um den codierten Header, die codierte Nutzlast und den privaten Schlüssel zu signieren.

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

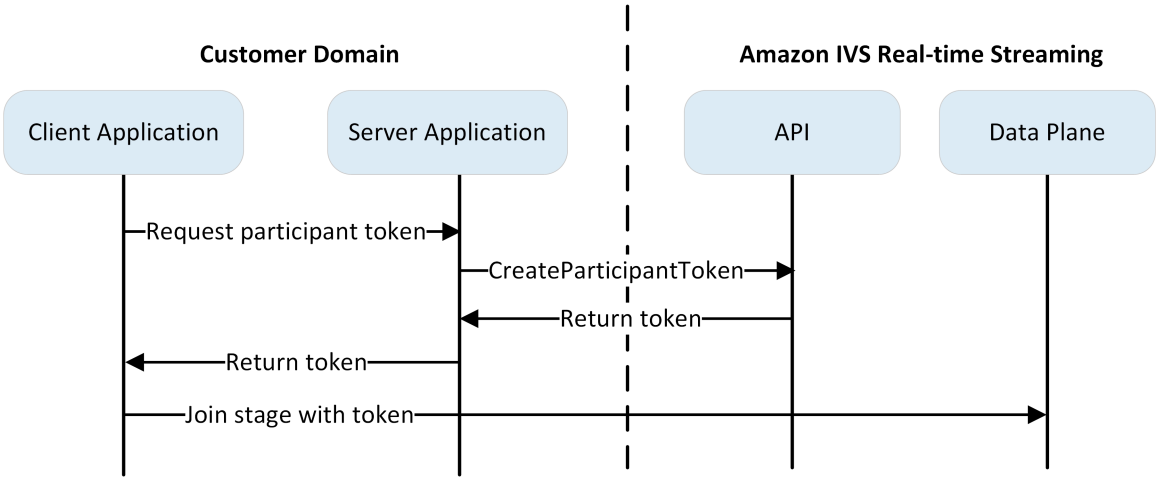
Anleitungen

1. Generieren Sie die Signatur des Tokens mit einem ES384-Signaturalgorithmus und einem privaten Schlüssel, der dem öffentlichen Schlüssel zugeordnet ist, der an IVS übergeben wird.
2. Montieren des Token.

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

```
base64UrlEncode(signature)
```

Erstellen von Token mit der IVS-Echtzeit-Streaming-API



Wie oben gezeigt, fragt eine Client-Anwendung Ihre serverseitige Anwendung nach einem Token, und die serverseitige Anwendung ruft CreateParticipantToken mithilfe eines AWS-SDK oder signierter Sigv4-Anfragen auf. Da AWS-Anmeldeinformationen zum Aufrufen der API verwendet werden, sollte das Token in einer sicheren serverseitigen Anwendung generiert werden, nicht in der clientseitigen Anwendung.

Beim Erstellen eines Teilnehmer-Tokens können Sie optional Attribute und/oder Fähigkeiten angeben:

- Sie können von der Anwendung bereitgestellte Attribute angeben, die in das Token codiert und an eine Stage angehängt werden sollen. Zuordnungsschlüssel und -werte können UTF-8-codierten Text enthalten. Die maximale Länge dieses Felds beträgt insgesamt 1 KB. Dieses Feld ist für alle Stageteilnehmer sichtbar und darf daher nicht für personenbezogene, vertrauliche oder sensible Informationen verwendet werden.
- Sie können Fähigkeiten angeben, die durch das Token aktiviert werden. Die Standardeinstellung ist `PUBLISH` und `SUBSCRIBE`, was es dem Teilnehmer ermöglicht, Audio und Video zu senden und zu empfangen, aber Sie können Tokens mit einer Teilmenge von Funktionen ausgeben. Sie könnten zum Beispiel einen Token ausgeben, der nur die Fähigkeit `SUBSCRIBE` für Moderatoren enthält. In diesem Fall könnten die Moderatoren die Teilnehmer sehen, die ein Video senden, aber kein eigenes Video senden.

Einzelheiten finden Sie unter CreateParticipantToken.

Sie können Teilnehmer-Token über die Konsole oder CLI zu Test- und Entwicklungszwecken erstellen. Höchstwahrscheinlich möchten Sie sie jedoch mit dem AWS-SDK in Ihrer Produktionsumgebung erstellen.

Sie benötigen eine Möglichkeit, um Token von Ihrem Server an alle Clients zu verteilen (z. B. über eine API-Anforderung). Diese Funktionalität wird von uns nicht bereitgestellt. Für diese Anleitung können Sie die Token einfach kopieren und in den folgenden Schritten in den Client-Code einfügen.

Wichtig: Behandeln Sie Token als nicht transparent, d. h. entwickeln Sie keine Funktionen, die auf Tokeninhalten basieren. Das Format von Token könnte sich in Zukunft ändern.

Anleitung für die Konsole

1. Navigieren Sie zu der Stage, die Sie im vorherigen Schritt erstellt haben.
2. Wählen Sie Token erstellen aus. Das Fenster Token erstellen wird angezeigt.
3. Geben Sie eine Benutzer-ID ein, die dem Token zugeordnet werden soll. Dies kann jeder UTF-8-kodierte Text sein.
4. Wählen Sie Erstellen aus.
5. Kopieren Sie das Token. Wichtig: Achten Sie darauf, das Token zu speichern. IVS speichert es nicht und Sie können es später nicht abrufen.

CLI-Anweisungen

Das Erstellen eines Chat-Tokens mit der AWS CLI ist eine erweiterte Option und erfordert, dass Sie zuerst die CLI auf Ihrem Computer herunterladen und konfigurieren. Informationen zu den ersten Schritten finden Sie im Benutzerhandbuch für die AWS-Befehlszeilenschnittstelle. Beachten Sie, dass die Generierung von Token mit der AWS-CLI für Testzwecke gut geeignet ist. Für den produktiven Einsatz empfehlen wir jedoch, Token auf der Serverseite mit dem AWS-SDK zu generieren (siehe Anweisungen unten).

1. Führen Sie den `create-participant-token`-Befehl mit dem Stage-ARN aus. Fügen Sie eine der folgenden Funktionen ein: `"PUBLISH"`, `"SUBSCRIBE"`.

```
aws ivs-realtime create-participant-token --stage-arn arn:aws:ivs:us-
west-2:376666121854:stage/VSWjvX5XOkU3 --capabilities '["PUBLISH", "SUBSCRIBE"]'
```

2. Dies gibt ein Teilnehmer-Token zurück:

```
{
    "participantToken": {
        "capabilities": [
            "PUBLISH",
            "SUBSCRIBE"
        ],
        "expirationTime": "2023-06-03T07:04:31+00:00",
        "participantId": "tU06DT5jCJeb",
        "token":
 "eyJhbGciOiJLTVMiLCJ0eXAiOiJKV1QifQ.eyJleHAiOjE2NjE1NDE0MjAsImp0aSI6ImpGcFdtdmVFTm9sUyIsInJl
TaKjllW9Qac6c5xBrdAk"
    }
}
```

3. Speichern Sie dieses Token. Sie benötigen dies, um der Stage beizutreten und Videos zu senden und zu empfangen.

AWS-SDK-Anweisungen

Sie können das AWS-SDK für die Erstellung von Tokens verwenden. Im Folgenden finden Sie Anweisungen für das AWS-SDK, das JavaScript verwendet.

Wichtig: Dieser Code muss serverseitig ausgeführt und seine Ausgabe an den Client übergeben werden.

Voraussetzung: Um das folgende Codebeispiel verwenden zu können, müssen Sie das Paket aws-sdk/client-ivs-realtime installieren. Weitere Informationen dazu finden Sie unter Erste Schritte mit AWS-SDK für JavaScript.

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

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

Schritt 4: Das IVS Broadcast SDK integrieren

IVS bietet ein Broadcast-SDK für Web, Android und iOS, das Sie in Ihre Anwendung integrieren können. Das Broadcast-SDK wird sowohl zum Senden als auch zum Empfangen von Videos verwendet. Wenn Sie RTMP-Erfassung für Ihre Stage konfiguriert haben, können Sie jeden Encoder verwenden, der an einen RTMP-Endpunkt senden kann (z. B. OBS oder ffmpeg).

In diesem Abschnitt schreiben wir eine einfache Anwendung, mit der zwei oder mehr Teilnehmer in Echtzeit interagieren können. Die folgenden Schritte führen Sie durch die Erstellung einer Anwendung namens BasicRealTime. Der vollständige Anwendungscode befindet sich auf CodePen und GitHub:

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

Web

Dateien einrichten

Richten Sie zunächst Ihre Dateien ein, indem Sie einen Ordner und eine erste HTML- und JS-Datei erstellen:

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

Sie können das Broadcast-SDK mit einem Script-Tag oder npm installieren. Unser Beispiel verwendet der Einfachheit halber das Script-Tag, kann aber leicht geändert werden, wenn Sie npm später verwenden möchten.

Verwenden eines Skript-Tags

Das Web Broadcast SDK wird als JavaScript-Bibliothek verteilt und kann unter https://web-broadcast.live-video.net/1.33.0/amazon-ivs-web-broadcast.js abgerufen werden.

Wenn sie per `<script>`-Tag geladen wird, stellt die Bibliothek eine globale Variable im Fensterbereich namens `IVSBroadcastClient` bereit.

Verwenden von npm

So installieren Sie das npm-Paket:

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

Sie können jetzt auf das IVSBroadcastClient-Objekt zugreifen:

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

Android

Erstellen Sie das Android-Projekt

1. Erstellen Sie ein Neues Projekt mit Android Studio.
2. Wählen Sie Aktivität „Leere Ansichten“.

   Hinweis: In einigen älteren Versionen von Android Studio heißt die ansichtsbasierte Aktivität Leere Aktivität. Wenn Ihr Android-Studio-Fenster Leere Aktivität und nicht Leere Ansichten-Aktivität anzeigt, wählen Sie Leere Aktivität. Andernfalls wählen Sie nicht Leere Aktivität, da wir View-APIs verwenden werden (nicht Jetpack Compose).
3. Geben Sie Ihrem Projekt einen Namen, wählen Sie dann Fertig.

Installieren Sie das Broadcast-SDK

Wenn Sie der Android-Entwicklungsumgebung die Amazon-IVS-Android-Broadcast-Bibliothek hinzufügen möchten, fügen Sie die Bibliothek der `build.gradle` – wie hier gezeigt – (für die neueste Version des Amazon IVS Broadcast SDK) zu Ihren Modulen hinzu. In neueren Projekten ist `mavenCentral` das Repository ist möglicherweise bereits in Ihrer `settings.gradle`-Datei. Wenn das der Fall ist, können Sie den `repositories`-Block weglassen. Für unser Beispiel müssen wir auch die Datenbindung im Block `android` aktivieren.

```
android {
    dataBinding.enabled true
}

repositories {
```

```
    mavenCentral()
}

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

Um das SDK manuell zu installieren, laden Sie alternativ die neueste Version von diesem Speicherort herunter:

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

iOS

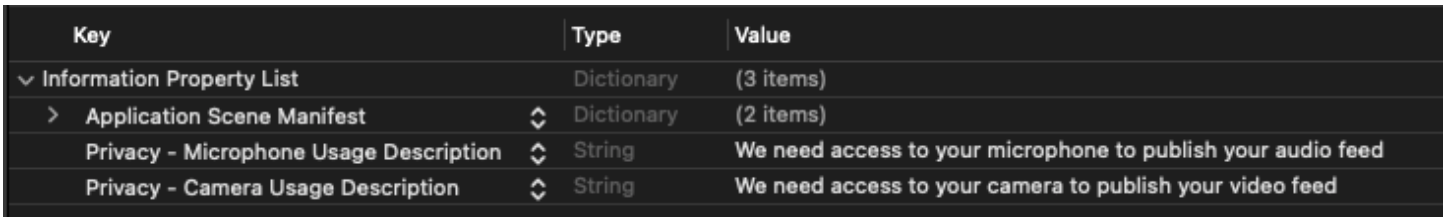
Erstellen des iOS-Projekts

1. Erstellen eines neuen Xcode-Projekts.
2. Für Plattform wählen Sie iOS.
3. Für Anwendung wählen Sie App.
4. Geben Sie den Namen des Produkts Ihrer Anwendung ein und wählen Sie Weiter.
5. Wählen (navigieren Sie zu) einem Verzeichnis, in dem das Projekt gespeichert werden soll, und wählen Sie dann Erstellen.

Als Nächstes müssen Sie das SDK einbringen. Anweisungen finden Sie unter Installieren der Bibliothek im Handbuch zum iOS-Broadcast-SDK.

So konfigurieren Sie Berechtigungen

Sie müssen die `Info.plist` Ihres Projekts aktualisieren, um zwei neue Einträge hinzuzufügen für `NSCameraUsageDescription` und `NSMicrophoneUsageDescription`. Geben Sie für die Werte benutzerfreundliche Erklärungen an, warum Ihre Anwendung nach Kamera- und Mikrofonzugriff fragt.



Schritt 5: Video veröffentlichen und abonnieren

Sie können Folgendes nutzen, um in IVS zu veröffentlichen/abonnieren (in Echtzeit):

- Die nativen IVS-Broadcast-SDKs, die WebRTC und RTMPS unterstützen. Wir empfehlen dies insbesondere für Produktionsszenarien. Nachfolgend finden Sie die Details für Web, Android und iOS.
- Die Amazon IVS-Konsole – Diese eignet sich zum Testen von Streams. Weitere Informationen finden Sie unter weiter unten in diesem Dokument.
- Andere Software- und Hardware-Encoder für das Streaming – Sie können alle Streaming-Encoder verwenden, die die Protokolle RTMP, RTMPS oder WHIP unterstützen. Weitere Informationen finden Sie unter Stream-Erfassung.

IVS-Konsole

1. Öffnen Sie die Amazon-IVS-Konsole.

   (Sie können auf die Amazon IVS Konsole auch über die AWS-Managementkonsole zugreifen.)
2. Wählen Sie im Navigationsbereich die Option Stage aus. (Wenn der Navigationsbereich ausgeblendet ist, erweitern Sie ihn über das Hamburger-Symbol.)
3. Wählen Sie die Stage aus, die Sie abonnieren oder veröffentlichen möchten, um ihre Detailseite aufzurufen.
4. Abonnieren: Wenn die Stage einen oder mehrere Publisher hat, können Sie sie abonnieren, indem Sie auf der Registerkarte Abonnieren auf die Schaltfläche Abonnieren klicken. Die Registerkarte wird unter dem Abschnitt Allgemeine Konfiguration angezeigt.
5. Zum Veröffentlichen:
   a. Wählen Sie die Registerkarte Veröffentlichen aus.
   b. Sie werden aufgefordert, der IVS-Konsole Zugriff auf Ihre Kamera und Ihr Mikrofon zu gewähren. Erlauben Sie diese Berechtigungen.
   c. Wählen Sie unten auf der Registerkarte Veröffentlichen mit den Dropdown-Feldern die Eingabegeräte für das Mikrofon und die Kamera aus.
   d. Um mit der Veröffentlichung zu beginnen, wählen Sie Veröffentlichung starten aus.
   e. Um Ihre veröffentlichten Inhalte anzuzeigen, kehren Sie zur Registerkarte Abonnieren zurück.

f. Um die Veröffentlichung zu beenden, klicken Sie auf der Registerkarte Veröffentlichen unten auf die Schaltfläche Veröffentlichung beenden.

Hinweis: Das Abonnieren und Veröffentlichen verbraucht Ressourcen, und für die Zeit, in der Sie mit der Stage verbunden sind, wird ein Stundensatz berechnet. Weitere Informationen finden Sie auf der Seite mit den IVS-Preisen unter Echtzeit-Streaming.

Mit dem IVS Web Broadcast SDK veröffentlichen und abonnieren

Dieser Abschnitt führt Sie durch die Schritte zur Veröffentlichung und zum Abonnieren einer Stage mithilfe Ihrer Web-App.

HTML-Boilerplate erstellen

Lassen Sie uns zunächst das HTML-Boilerplate erstellen und die Bibliothek als Script-Tag importieren:

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

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

  <!-- Import the SDK -->
  <script src="https://web-broadcast.live-video.net/1.33.0/amazon-ivs-web-
broadcast.js"></script>
</head>

<body>

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

</body>
</html>
```

Token-Eingabe akzeptieren und Schaltflächen zum Beitritten/Verlassen hinzufügen

Hier füllen wir den Hauptteil mit unseren Eingabekontrollen aus. Diese nehmen das Token als Eingabe und richten Schaltflächen für Beitreten und Verlassen ein. Normalerweise fordern Anwendungen das Token von der API Ihrer Anwendung an, aber in diesem Beispiel kopieren Sie das Token und fügen es in die Token-Eingabe ein.

```
<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 />
```

Mediencontainer-Elemente hinzufügen

Diese Elemente werden die Medien für unsere lokalen und externen Teilnehmer bereitstellen. Wir fügen ein Script-Tag hinzu, um die in `app.js` definierte Logik unserer Anwendung zu laden.

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

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

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

Damit ist die HTML-Seite fertig. Sie sollten sie sehen, wenn Sie `index.html` in einem Browser laden:

Erstellen von app.js

Gehen wir zur Definition des Inhalts unserer `app.js`-Datei. Importieren Sie zunächst alle erforderlichen Eigenschaften aus der globalen Version des SDK:

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

Anwendungsvariablen erstellen

Erstellen Sie Variablen, um Verweise auf unsere HTML-Elemente für die Schaltflächen Beitreten und Verlassen zu speichern und den Status für die Anwendung zu speichern:

```
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;
```

JoinStage 1 erstellen: Definieren Sie die Funktion und validieren Sie die Eingabe

Die Funktion `joinStage` nimmt das Eingabe-Token, stellt eine Verbindung zur Stage her und beginnt mit der Veröffentlichung von Video- und Audiodaten, die von `getUserMedia` empfangen werden.

Zu Beginn definieren wir die Funktion und validieren den Status und die Token-Eingabe. Wir werden diese Funktion in den nächsten Abschnitten näher erläutern.

```
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
};
```

JoinStage 2 erstellen: Medien zum Veröffentlichen abrufen

Hier sind die Medien, die auf der Stage veröffentlicht werden:

```
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]);
```

JoinStage 3 erstellen: Definieren Sie die Stagestrategie und erstellen Sie die Stage

Diese Stagestrategie ist das Herzstück der Entscheidungslogik, anhand derer das SDK entscheidet, was veröffentlicht und welche Teilnehmer abonniert werden sollen. Weitere Informationen zum Zweck der Funktion finden Sie unter Strategie.

Diese Strategie ist einfach. Nachdem Sie die Stage betreten haben, veröffentlichen Sie die soeben abgerufenen Streams und abonnieren die Audio- und Videodaten aller Remote-Teilnehmer:

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

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

JoinStage 4 erstellen: Stageereignisse verarbeiten und Medien rendern

Stages geben viele Ereignisse ab. Wir müssen auf die `STAGE_PARTICIPANT_STREAMS_ADDED` und `STAGE_PARTICIPANT_LEFT` hören, um Medien auf und von der Seite zu rendern und zu entfernen. Eine umfassendere Reihe von Ereignissen finden Sie unter Ereignisse.

Beachten Sie, dass wir hier vier Hilfsfunktionen erstellen, die uns bei der Verwaltung der erforderlichen DOM-Elemente unterstützen:`setupParticipant`, `teardownParticipant`, `createVideoEl` und `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;
}
```

JoinStage 5 erstellen: Treten Sie der Stage bei

Vervollständigen wir unsere Funktion `joinStage` , indem Sie endlich die Stage betreten!

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

LeaveStage erstellen

Definieren Sie die `leaveStage`-Funktion, die die Verlassen-Schaltfläche aufrufen wird.

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

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

Input-Event-Handler initialisieren

Wir fügen eine letzte Funktion zu unserer `app.js`-Datei hinzu. Diese Funktion wird sofort aufgerufen, wenn die Seite geladen wird, und richtet Event-Handler für den Beitritt und das Verlassen der Stage ein.

```
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
```

Führen Sie die Anwendung aus und geben Sie ein Token an

Jetzt können Sie die Webseite lokal oder mit anderen teilen, die Seite öffnen, ein Teilnehmer-Token eingeben und der Stage beitreten.

Die nächsten Themen

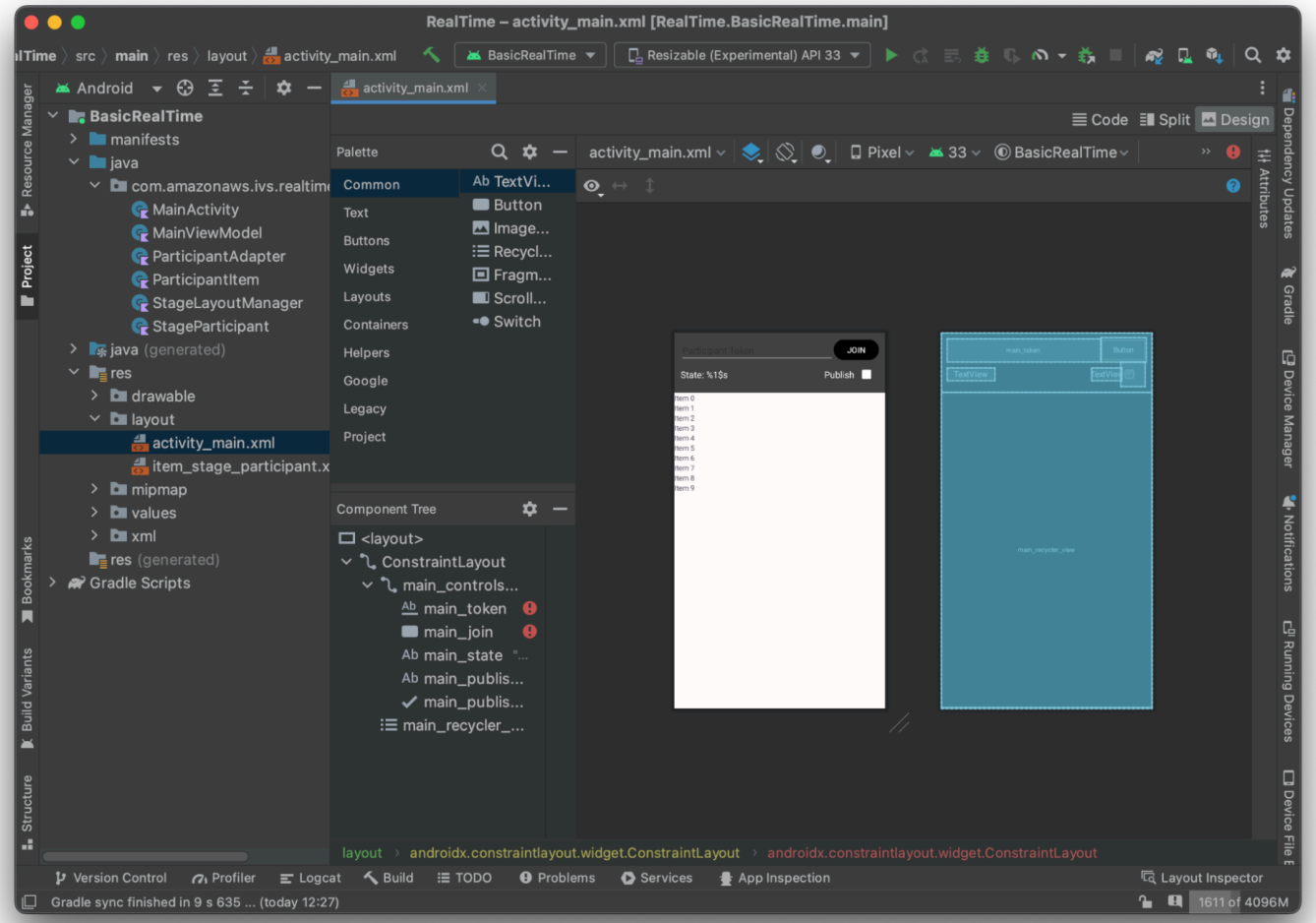
Ausführlichere Beispiele für npm, React und mehr finden Sie in IVS-Broadcast-SDK: Web-Leitfaden (Anleitung zu Echtzeit-Streaming).

Mit dem IVS Android Broadcast SDK veröffentlichen und abonnieren

Dieser Abschnitt führt Sie durch die Schritte zur Veröffentlichung und zum Abonnieren einer Stage mithilfe Ihrer Android-App.

Ansichten erstellen

Wir beginnen mit der Erstellung eines einfachen Layouts für unsere Anwendung mithilfe der automatisch erstellten `activity_main.xml`-Datei. Das Layout enthält einen `EditText`, um ein Token hinzuzufügen, einen Beitreten-`Button`, eine `TextView`, um den Status der Stage anzuzeigen, und eine `CheckBox`, um die Veröffentlichung umzuschalten.



Hier ist das XML hinter der Ansicht:

```
<?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>
```

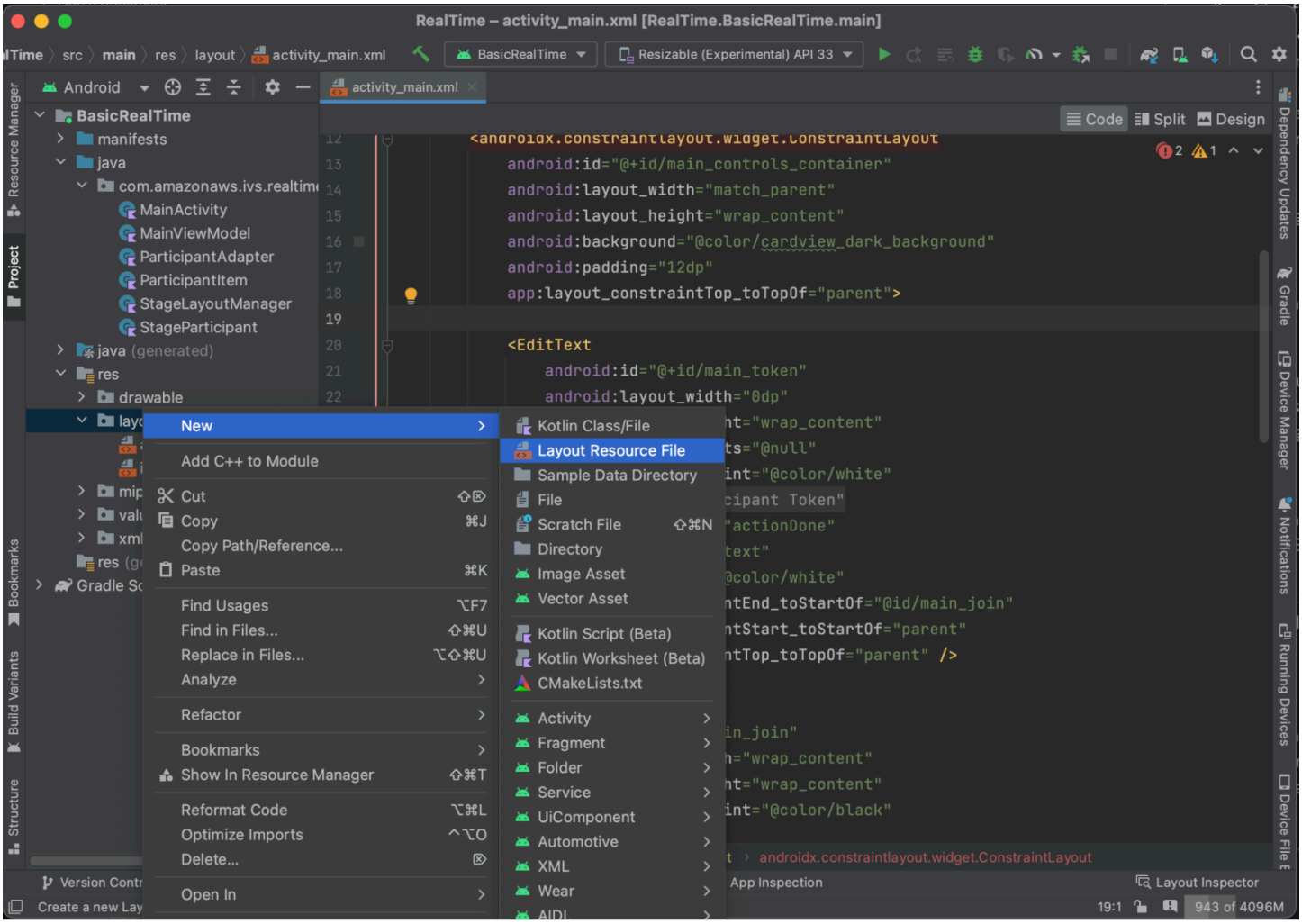
Wir haben hier auf ein paar String-IDs verwiesen, also erstellen wir unsere gesamte `strings.xml`-Datei jetzt:

```
<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>
```

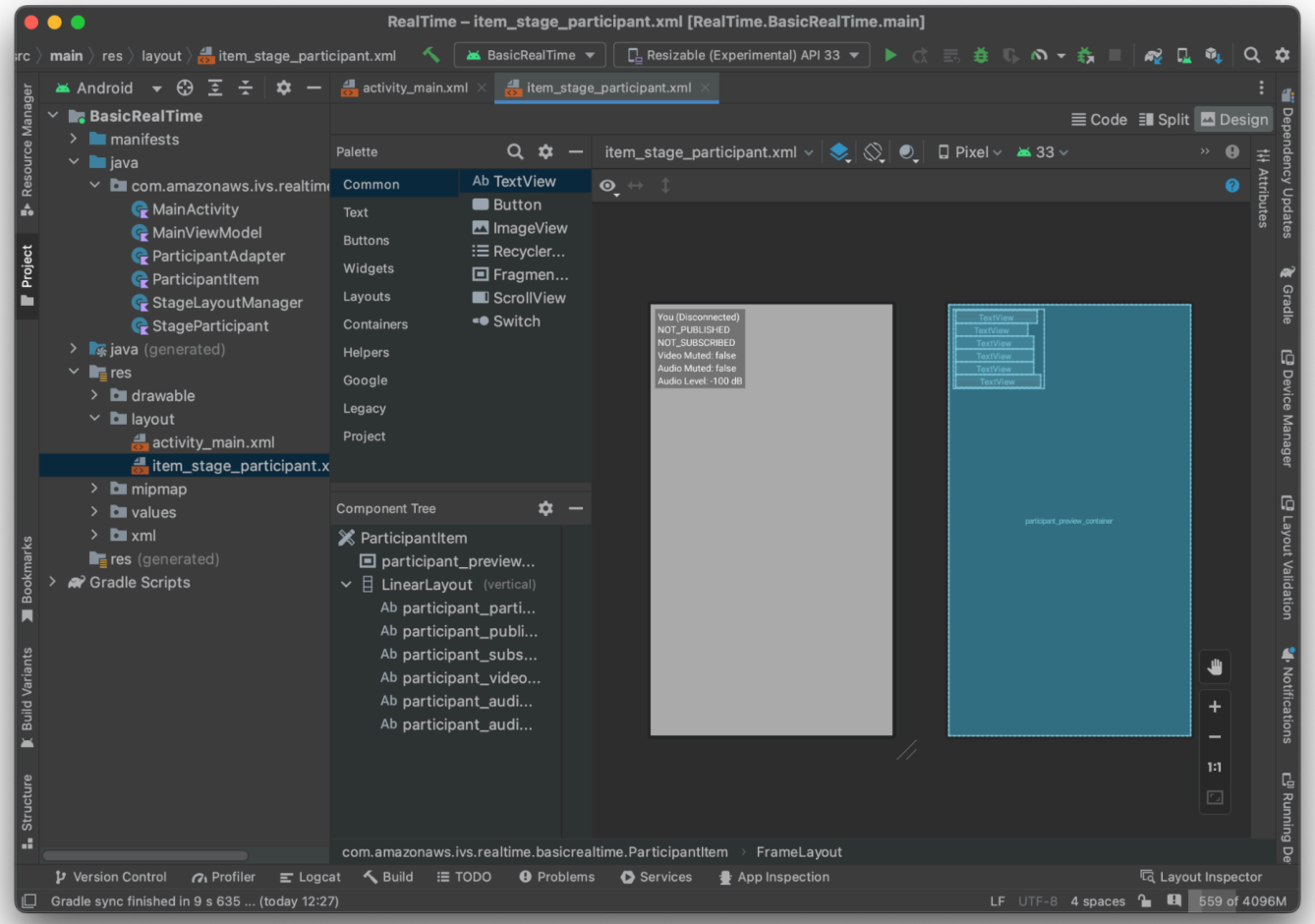
Lassen Sie uns diese Ansichten im XML mit unseren `MainActivity.kt` verknüpfen:

```
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)
}
```

Jetzt erstellen wir eine Elementansicht für unsere `RecyclerView`. Klicken Sie dazu mit der rechten Maustaste auf `res/layout`-Verzeichnis und wählen Sie Neu > Layout-Ressourcendatei. Benennen Sie diese Datei `item_stage_participant.xml`.



Das Layout für dieses Element ist einfach: Es enthält eine Ansicht zum Rendern des Videostreams eines Teilnehmers und eine Liste von Labels zur Anzeige von Informationen über den Teilnehmer:



Hier ist das 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>
```

Diese XML-Datei generiert eine Klasse, die wir noch nicht erstellt haben, `ParticipantItem`. Da das XML den vollständigen Namespace enthält, sollten Sie diese XML-Datei unbedingt in Ihren Namespace aktualisieren. Lassen Sie uns diese Klasse erstellen und die Ansichten einrichten, aber ansonsten lassen wir das Feld vorerst leer.

Erstellen Sie eine neue Kotlin-Klasse, `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)
    }
}
```

Berechtigungen

Um die Kamera und das Mikrofon verwenden zu können, müssen Sie vom Benutzer Berechtigungen anfordern. Dafür folgen wir einem standardmäßigen Berechtigungsablauf:

```
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
    }
}
```

Anwendungsstatus

Unsere Anwendung verfolgt die Teilnehmer vor Ort in einer `MainViewModel.kt` und der Status wird an die `MainActivity` zurückgemeldet mit Kotlins StateFlow.

Erstellen Sie eine neue Kotlin-Klasse `MainViewModel`:

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

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

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

}
```

In `MainActivity.kt` verwalten wir unser Ansicht-Modell:

```
import androidx.activity.viewModels

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

Um `AndroidViewModel` und diese Kotlin-`ViewModel`-Erweiterungen zu verwenden, müssen Sie Folgendes zur `build.gradle`-Datei Ihres Moduls hinzufügen:

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

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

RecyclerView-Adapter

Wir werden eine einfache `RecyclerView.Adapter`-Unterklasse erstellen, um unsere Teilnehmer zu verfolgen und unsere `RecyclerView` auf Stageereignisse zu aktualisieren. Aber zuerst brauchen wir eine Klasse, die einen Teilnehmer repräsentiert. Erstellen Sie eine neue Kotlin-Klasse `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)
            }
        }
}
```

Wir verwenden diese Klasse in der `ParticipantAdapter`-Klasse, die wir als Nächstes erstellen werden. Wir beginnen damit, die Klasse zu definieren und eine Variable zu erstellen, um die Teilnehmer zu verfolgen:

```
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>()
```

Wir müssen auch unsere `RecyclerView.ViewHolder` definieren bevor Sie die restlichen Überschreibungen implementieren:

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

Damit können wir die Standard-`RecyclerView.Adapter`-Überschreibung implementieren:

```
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)
    }
}
```

Schließlich fügen wir neue Methoden hinzu, die wir von unserem `MainViewModel` abrufen, wenn Änderungen an den Teilnehmern vorgenommen werden. Bei diesen Methoden handelt es sich um Standard-CRUD-Operationen auf dem Adapter.

```
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])
    }
}
```

Wieder in `MainViewModel` müssen wir einen Verweis auf diesen Adapter erstellen und speichern:

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

Stage-Status

Wir müssen auch einige Stagestatus innerhalb `MainViewModel` verfolgen. Definieren wir jetzt diese Eigenschaften:

```
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>()
```

Um Ihre eigene Vorschau zu sehen, bevor Sie eine Stage betreten, erstellen wir sofort einen lokalen Teilnehmer:

```
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)
}
```

Wir wollen sicherstellen, dass wir diese Ressourcen bereinigen, wenn unsere `ViewModel` bereinigt ist. Wir überschreiben `onCleared()` sofort, damit wir nicht vergessen, diese Ressourcen zu reinigen.

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

Jetzt füllen wir unsere lokale `streams`-Eigenschaft auf, sobald die Berechtigungen erteilt sind, und implementieren die `permissionsGranted`-Methode, die wir zuvor aufgerufen haben:

```
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)
    }
}
```

Implementierung des Stage-SDK

Drei Kernkonzepte liegen der Echtzeit-Funktionalität zugrunde: Stage, Strategie und Renderer. Das Designziel besteht in der Minimierung der Menge an clientseitiger Logik, die für die Entwicklung eines funktionierenden Produkts erforderlich ist.

Stage.Strategy

Die `Stage.Strategy`-Implementierung ist einfach:

```
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
}
```

Zusammenfassend lässt sich sagen, dass wir auf der Grundlage unseres internen Status `publishEnabled` veröffentlichen, und wenn wir veröffentlichen, veröffentlichen wir die zuvor gesammelten Streams. Schließlich abonnieren wir für dieses Beispiel immer andere Teilnehmer und erhalten sowohl ihr Audio als auch ihr Video.

StageRenderer

Die `StageRenderer`-Implementierung ist ebenfalls ziemlich einfach, obwohl sie angesichts der Anzahl der Funktionen reichlich mehr Code enthält. Der allgemeine Ansatz in diesem Renderer besteht darin, unseren `ParticipantAdapter` zu aktualisieren, wenn das SDK uns über eine Änderung an einem Teilnehmer informiert. Es gibt bestimmte Szenarien, in denen wir mit lokalen Teilnehmern anders umgehen, weil wir beschlossen haben, sie selbst zu verwalten, sodass sie ihre Kameravorschau sehen können, bevor sie beitreten.

```
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) {}
}
```

Implementierung eines benutzerdefinierten RecyclerView-LayoutManagers

Die Festlegung verschiedener Teilnehmerzahlen kann komplex sein. Sie möchten, dass sie den gesamten Frame der übergeordneten Ansicht einnehmen, aber Sie möchten nicht jede Teilnehmerkonfiguration unabhängig voneinander handhaben. Um dies zu vereinfachen, führen wir die Implementierung eines `RecyclerView.LayoutManager` durch.

Erstelle eine weitere neue Klasse `StageLayoutManager`, welche `GridLayoutManager` erweitern soll. In diesem Kurs wird das Layout für jeden Teilnehmer anhand der Anzahl der Teilnehmer in einem flussbasierten Zeilen-/Spaltenlayout berechnet. Jede Zeile hat dieselbe Höhe wie die anderen, aber Spalten können pro Zeile unterschiedlich breit sein. Sehen Sie den Code-Kommentar über der `layouts`-Variable für eine Beschreibung, wie dieses Verhalten angepasst werden kann.

```
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
}
```

Wieder zurück in `MainActivity.kt` müssen wir den Adapter und den Layoutmanager für unsere `RecyclerView` einrichten:

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

UI-Aktionen verbinden

Wir sind nah dran; es gibt nur ein paar UI-Aktionen, die wir verbinden müssen.

Zuerst haben wir unsere `MainActivity`, die die `StateFlow`-Änderungen von`MainViewModel` beobachtet:

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

Als Nächstes fügen wir Listener zu unseren Schaltflächen „Beitreten“ und „Veröffentlichen“ hinzu:

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

Beide der oben genannten Anruffunktionen in unserer `MainViewModel`, die wir jetzt implementieren:

```
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
}
```

Rendern der Teilnehmer

Schließlich müssen wir die Daten, die wir vom SDK erhalten, auf das zuvor erstellte Teilnehmerelement übertragen. Wir haben die Logik von `RecyclerView` bereits fertig, also müssen wir nur noch die API von `bind` in `ParticipantItem` implementieren.

Wir beginnen mit dem Hinzufügen der leeren Funktion und gehen sie dann Schritt für Schritt durch:

```
fun bind(participant: StageParticipant) {

}
```

Zuerst kümmern wir uns um den Status „Einfach“, die Teilnehmer-ID, den Veröffentlichungsstatus und den Abonnementstatus. Für diese aktualisieren wir einfach unsere `TextViews` direkt:

```
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
```

Als Nächstes aktualisieren wir die stummgeschalteten Audio- und Videozustände. Um den Stummschaltzustand zu erhalten, müssen wir den `ImageDevice` und `AudioDevice` aus dem Streams-Array finden. Um die Leistung zu optimieren, erinnern wir uns an die zuletzt angehängten Geräte-IDs.

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

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

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

Abschließend wollen wir eine Vorschau für das `imageDevice` rendern:

```
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
```

Und wir zeigen Audiostatistiken von der `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
```

Mit dem IVS iOS Broadcast SDK veröffentlichen und abonnieren

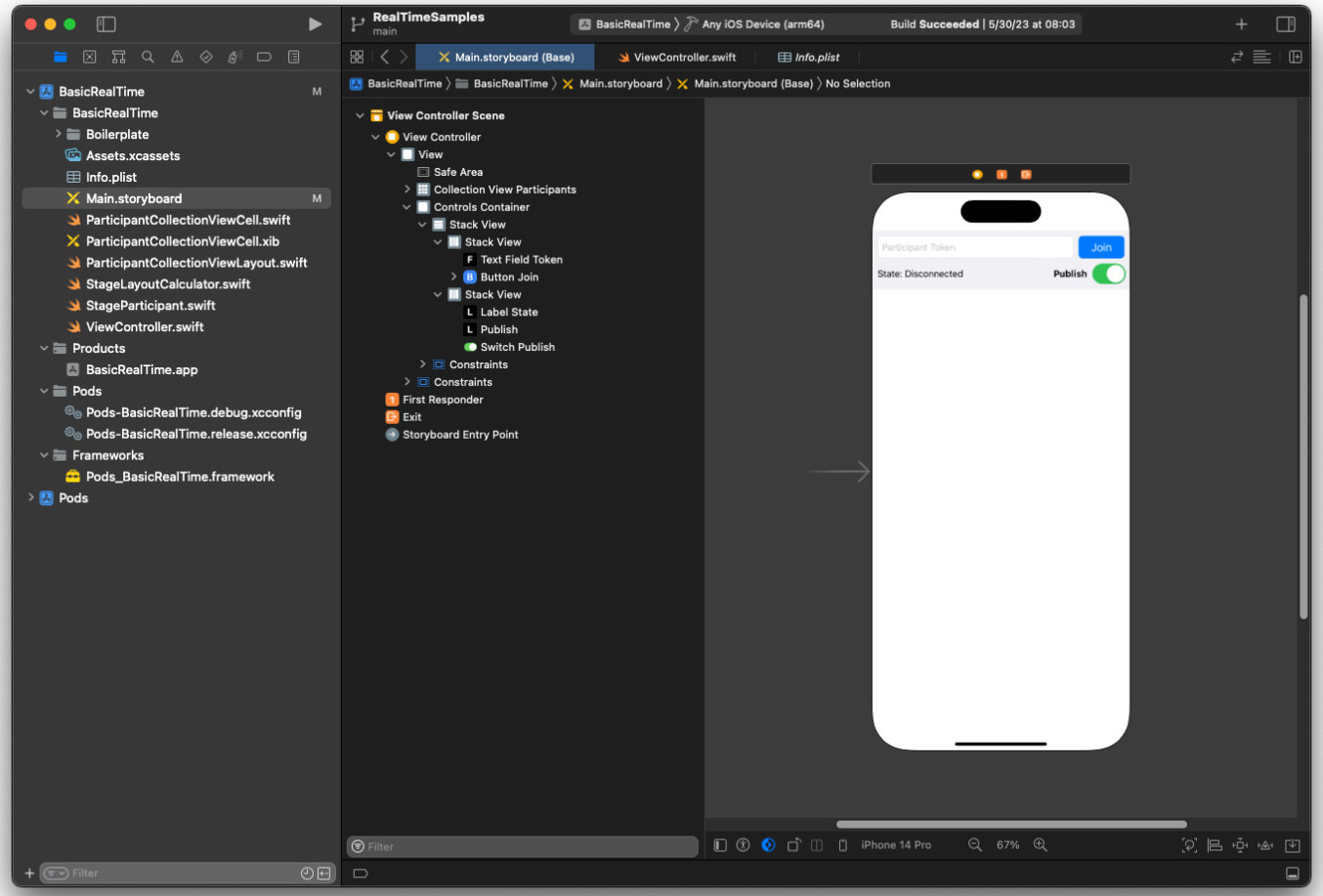
Dieser Abschnitt führt Sie durch die Schritte zur Veröffentlichung und zum Abonnieren einer Stage mithilfe Ihrer iOS-App.

Ansichten erstellen

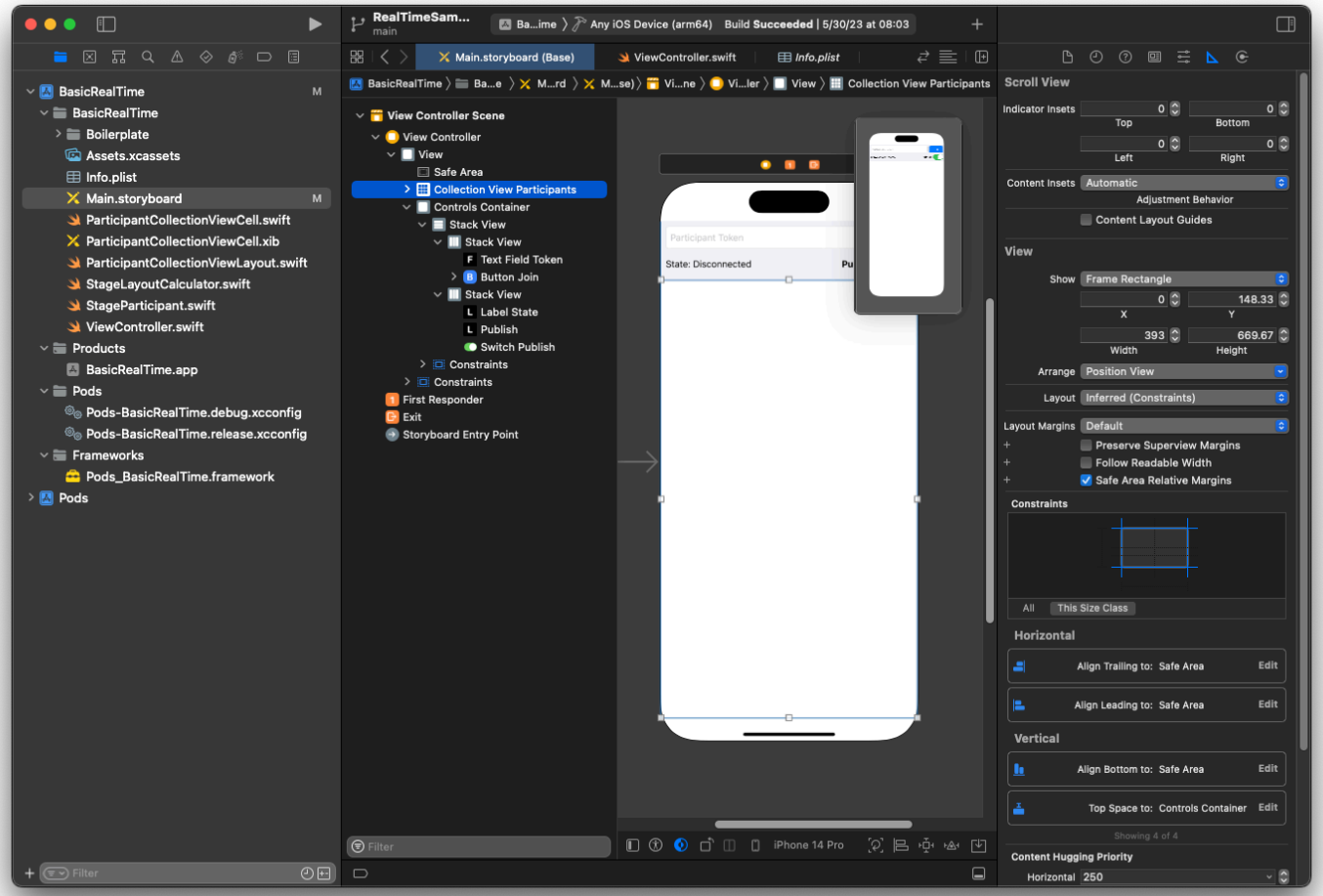
Wir beginnen mit der automatisch erstellten `ViewController.swift`-Datei, um `AmazonIVSBroadcast` zu importieren und dann fügen wir etwas `@IBOutlets` hinzu zum Verlinken:

```
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!
```

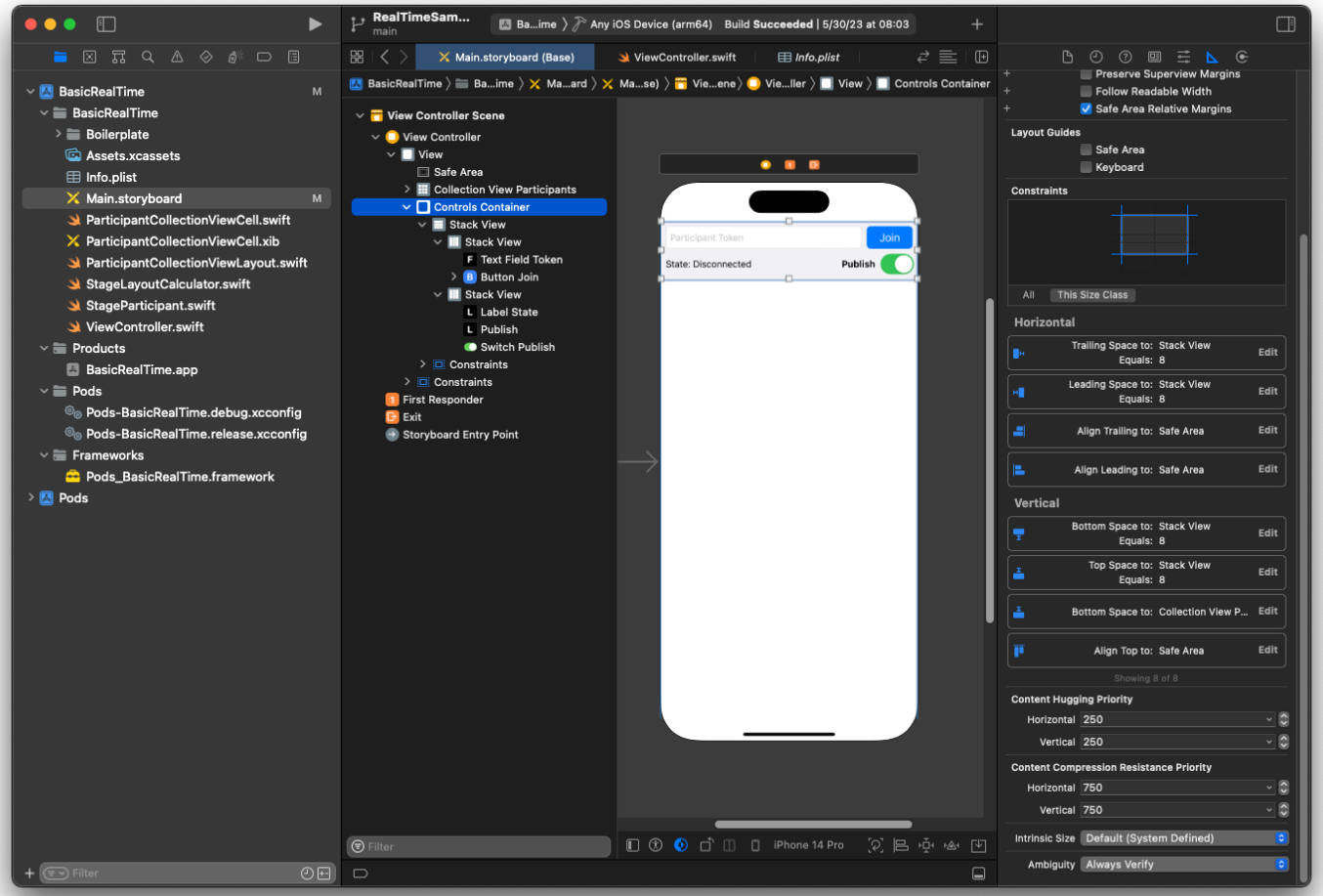
Jetzt erstellen wir diese Ansichten und verknüpfen sie in `Main.storyboard`. Hier ist die Ansichtsstruktur, die wir verwenden werden:



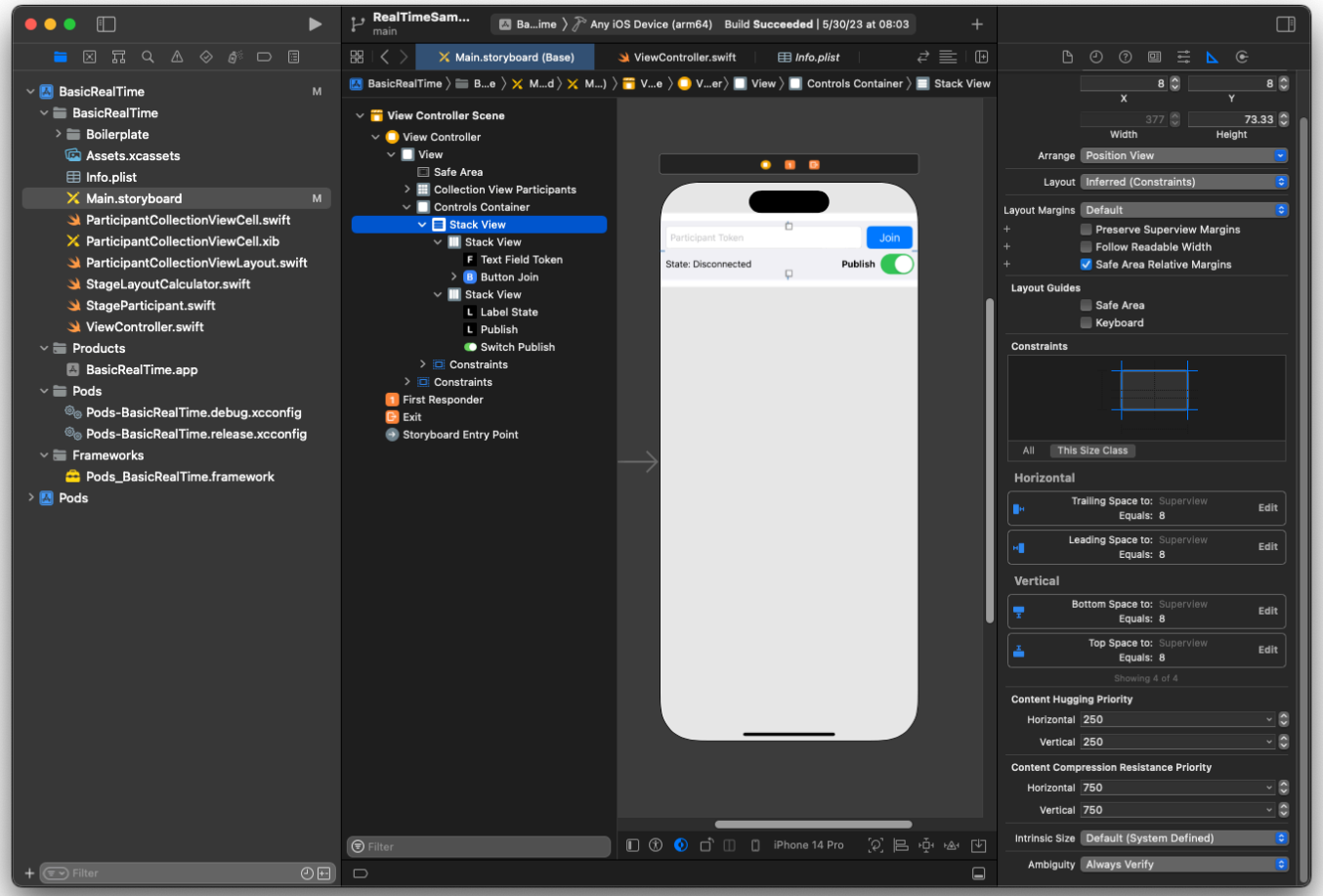
Für die AutoLayout-Konfiguration müssen wir drei Ansichten anpassen. Die erste Ansicht ist Sammlungsansicht der Teilnehmer (ein `UICollectionView`). Binden Sie Führend, Verfolgend, und Unterseite zu Sicherer Bereich. Binden Sie auch Oben zu Steuert Container.



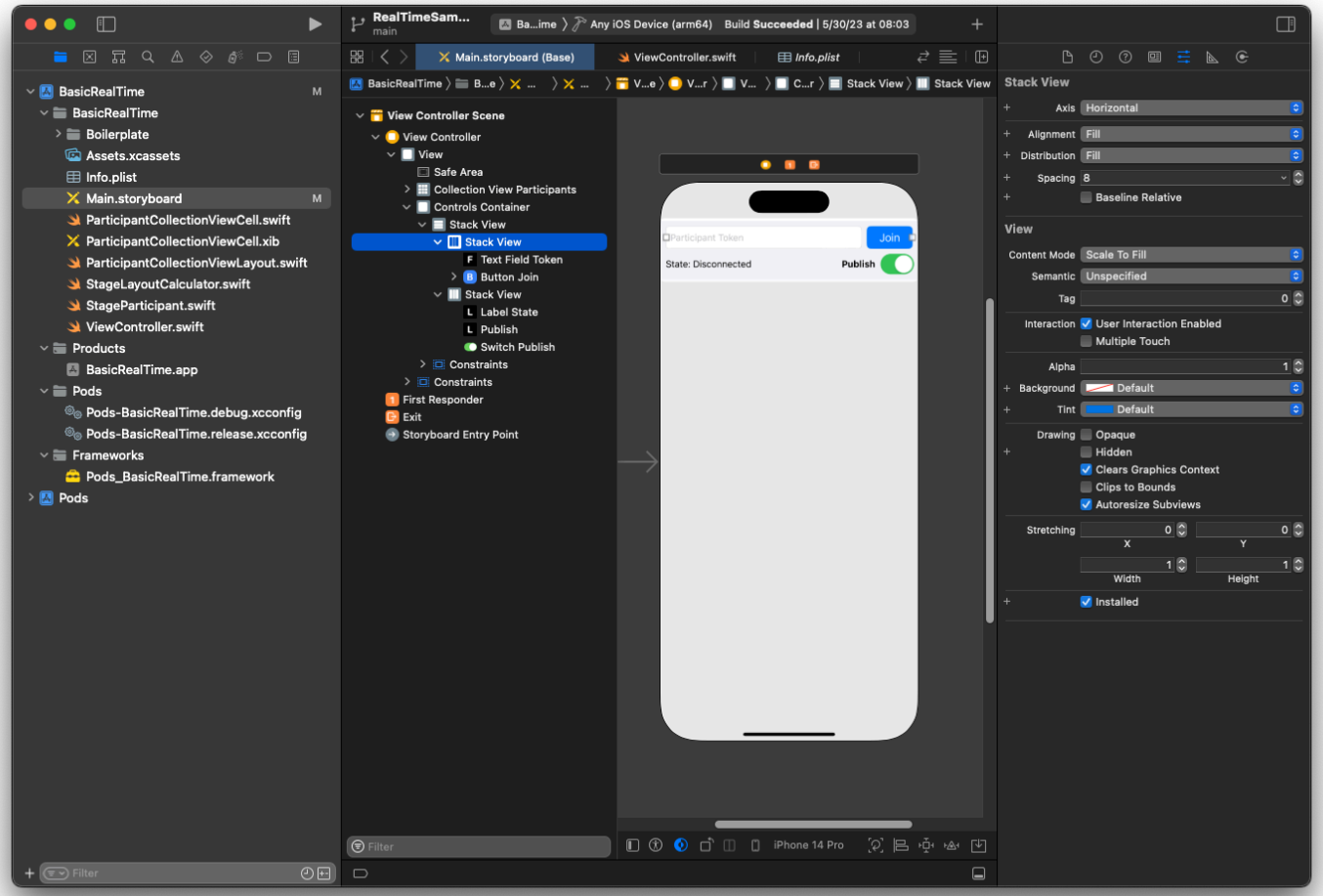
Die zweite Ansicht ist Steuert Container. Binden Sie Führend, Verfolgend, und Unterseite zu Sicherer Bereich:



Die dritte und letzte Ansicht ist Vertikale Stapelansicht. Binden Sie Oben,Führend,Verfolgend, und Unterseite zu Superansicht. Stellen Sie für das Styling den Abstand auf 8 statt auf 0 ein.



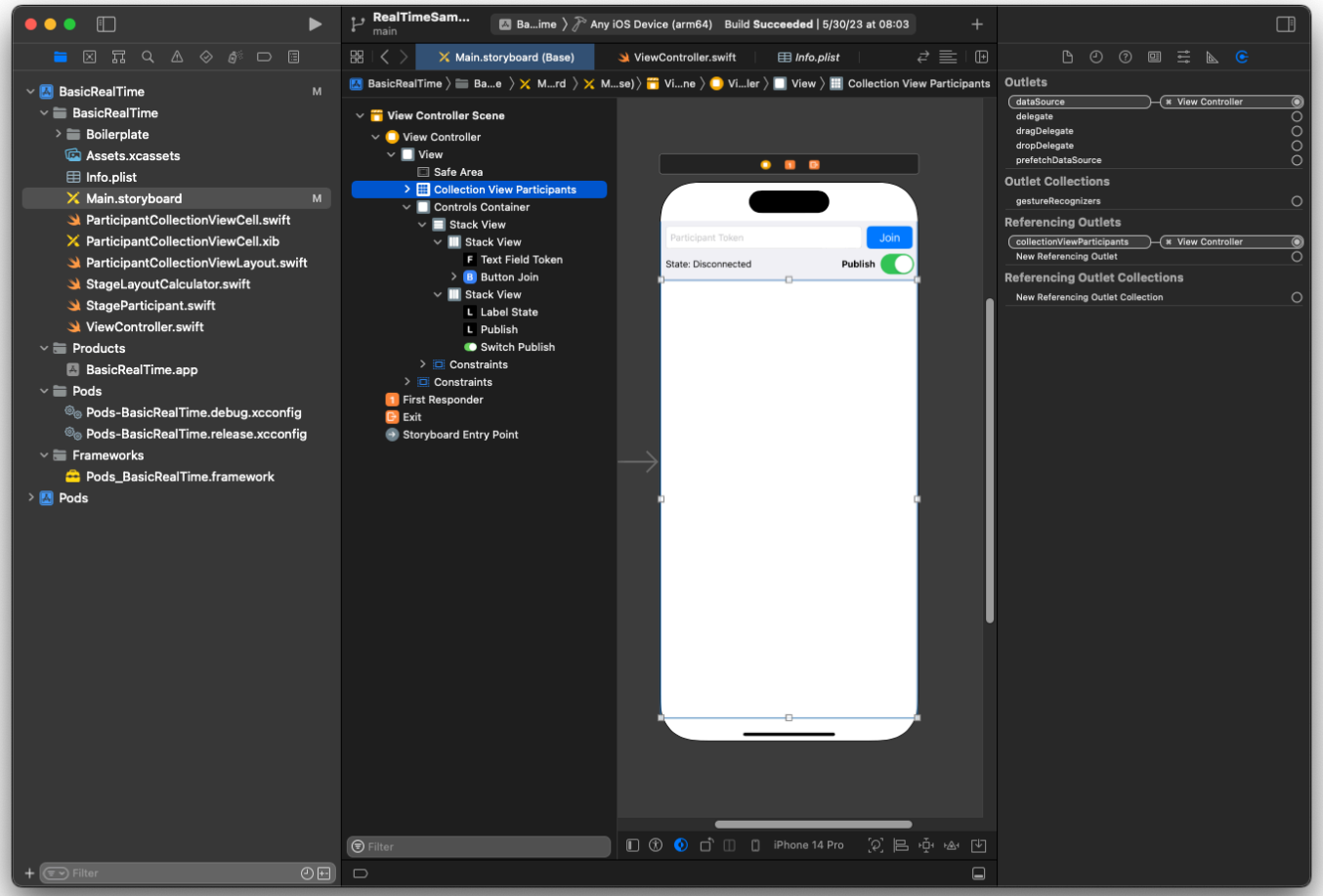
UIStack-Ansichten kümmert sich um das Layout der verbleibenden Ansichten. Für alle drei UIStack-Ansichten, verwenden Sie Füllen, sowie Ausrichtung und Verteilung.



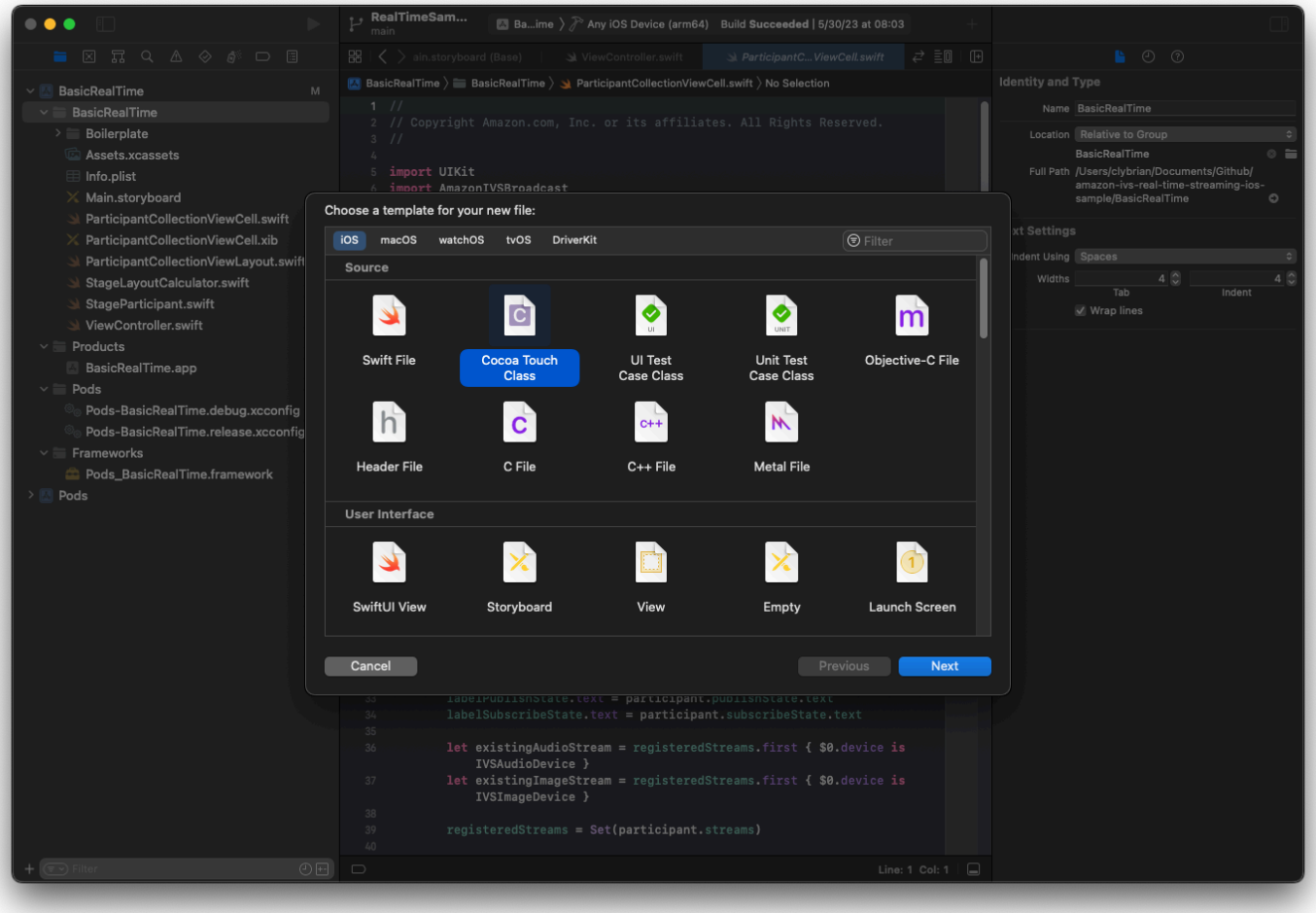
Lassen Sie uns abschließend diese Ansichten mit unserem `ViewController` verknüpfen. Kartieren Sie von oben die folgenden Ansichten:

- Textfeld-Verknüpfung bindet an `textFieldToken`.
- Schaltfläche Beitreten bindet an `buttonJoin`.
- Status beschriften bindet an `labelState`.
- Veröffentlichen wechseln bindet an `switchPublish`.
- Sammlungsansicht der Teilnehmer bindet an `collectionViewParticipants`.

Nutzen Sie diese Zeit auch, um das `dataSource` des Elements Sammlungsansicht der Teilnehmer auf den Besitz von `ViewController` einzustellen:



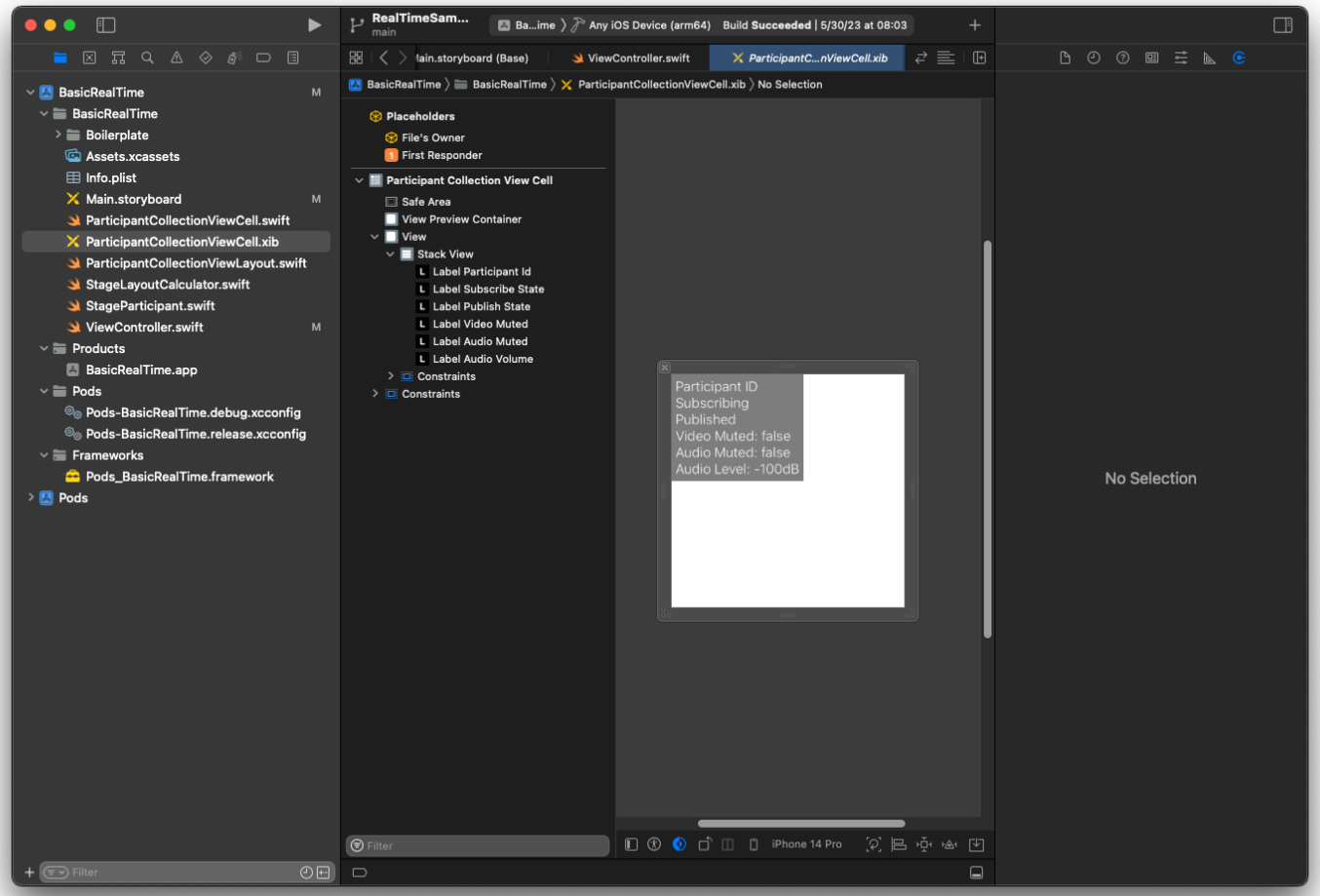
Jetzt erstellen wir die `UICollectionViewCell`-Unterklasse, in welcher die Teilnehmer gerendert werden sollen. Erstellen Sie zunächst eine neue Cocoa-Touch-Class-Datei:



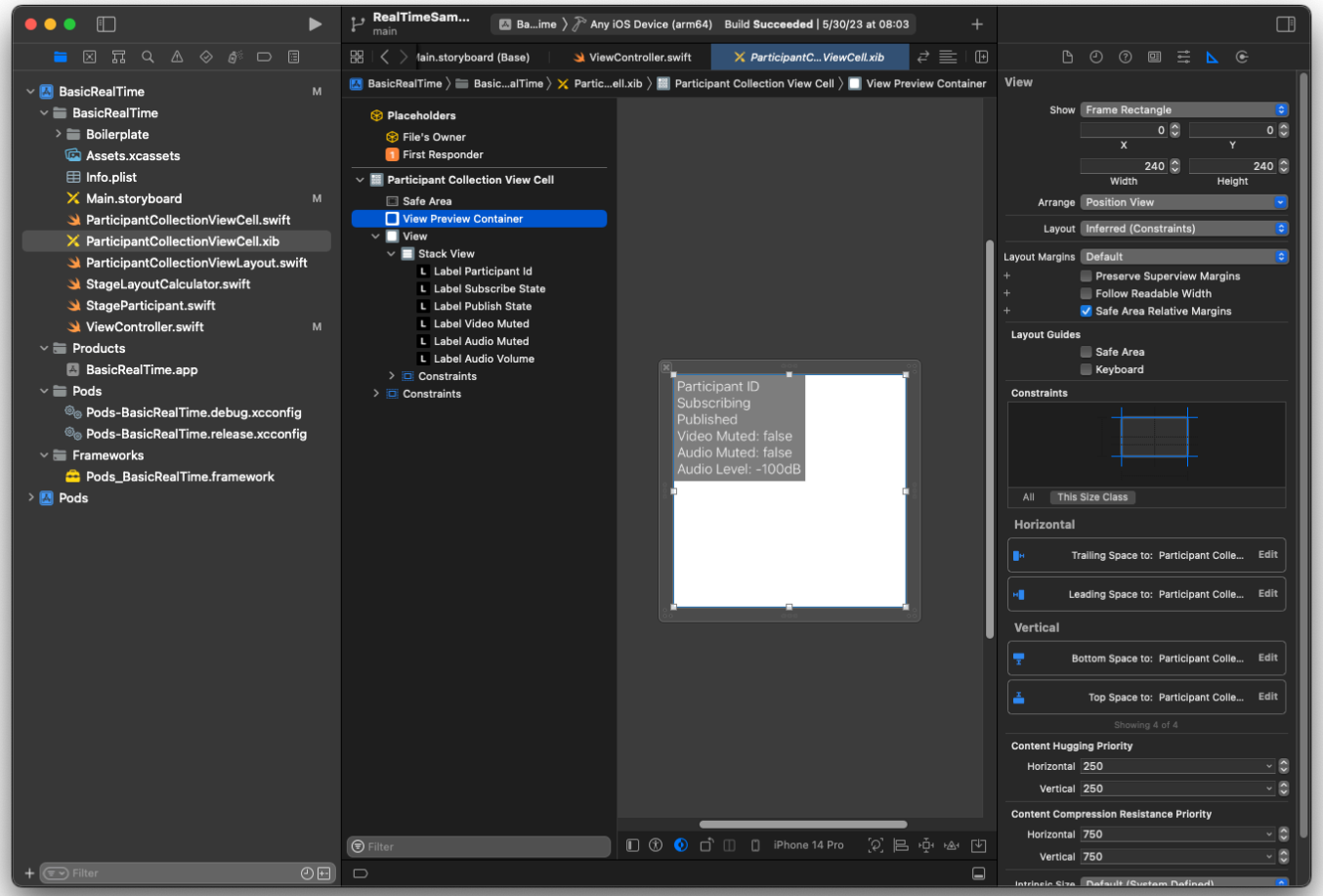
Nennen Sie sie `ParticipantUICollectionViewCell` und machen Sie es zu einer Unterklasse von `UICollectionViewCell` in Swift. Wir beginnen erneut in der Swift-Datei und erstellen unsere `@IBOutlets` zum Verlinken:

```
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!
```

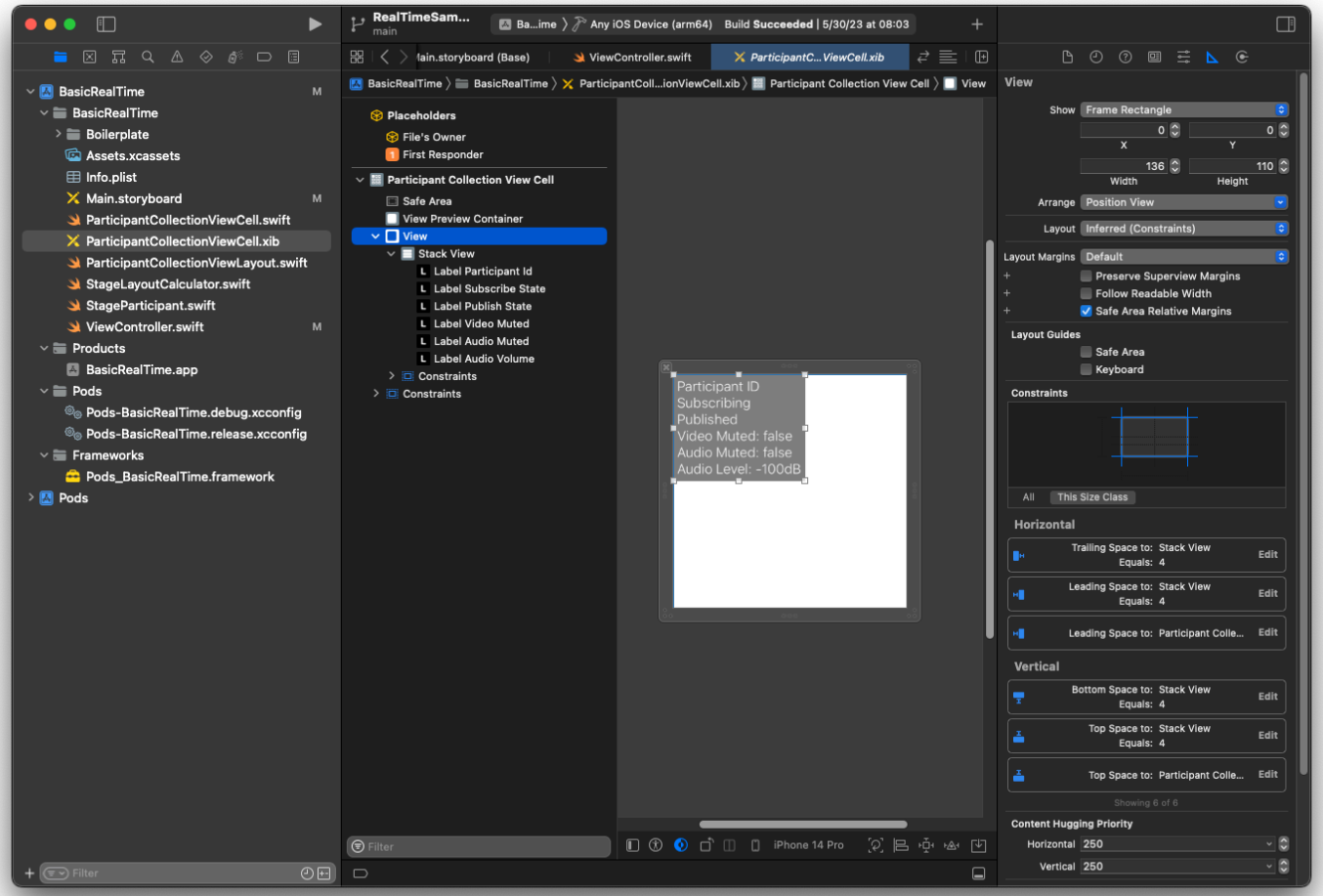
Erstellen Sie in der zugehörigen XIB-Datei diese Ansichtshierarchie:



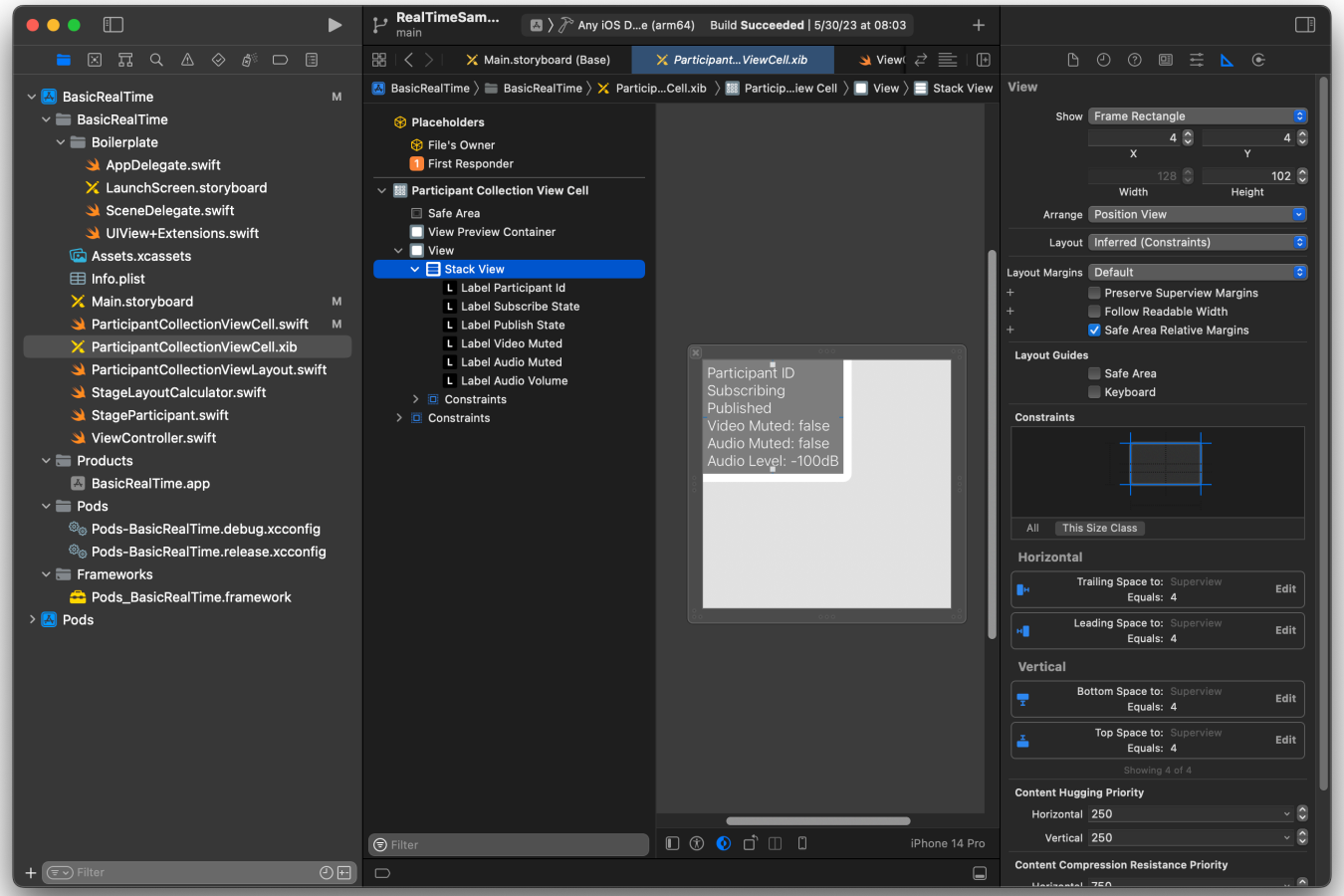
Für AutoLayout ändern wir erneut drei Ansichten. Die erste Ansicht ist Vorschaucontainer anzeigen. Setzen Sie Verfolgend, Führend, Oben und Unterseite zu Sammlungsansicht der Teilnehmer Zelle.



Die zweite Ansicht ist Ansicht. Setzen Sie Führend und Oben zu Sammlungsansicht der Teilnehmer Zelle und ändern Sie den Wert auf 4.



Die dritte Ansicht ist Stapelansicht. Setzen Sie Verfolgend, Führend, Oben und Unten auf Superansicht und ändern Sie den Wert auf 4.



Berechtigungen und Idle Timer

Zurück zu unserem `ViewController`. Wir werden den System-Leerlauf-Timer deaktivieren, um zu verhindern, dass das Gerät in den Ruhemodus wechselt, während unsere Anwendung verwendet wird:

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

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

Als Nächstes fordern wir Kamera- und Mikrofonberechtigungen vom System an:

```
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)
    }
}
```

Anwendungsstatus

Wir müssen unsere `collectionViewParticipants` konfigurieren mit der Layout-Datei, die wir zuvor erstellt haben:

```
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")
}
```

Um jeden Teilnehmer zu repräsentieren, erstellen wir eine einfache Struktur namens `StageParticipant`. Diese kann enthalten sein in der `ViewController.swift`-Datei, oder es kann eine neue Datei erstellt werden.

```
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
    }
}
```

Um diese Teilnehmer zu verfolgen, verwahren wir eine Reihe von ihnen als Privateigentum in unserem `ViewController`:

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

Diese Eigenschaft wird verwendet, um unsere `UICollectionViewDataSource` anzutreiben, die früher vom Storyboard aus verlinkt wurde:

```
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'")
        }
    }

}
```

Um Ihre eigene Vorschau zu sehen, bevor Sie eine Stage betreten, erstellen wir sofort einen lokalen Teilnehmer:

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

Dies führt dazu, dass sofort nach dem Ausführen der App eine Teilnehmerzelle gerendert wird, die den lokalen Teilnehmer darstellt.

Die Benutzer möchten sich selbst sehen können, bevor sie einer Stage beitreten. Deshalb implementieren wir als Nächstes die `setupLocalUser()`-Methode, die zuvor aus dem Code zur Bearbeitung von Berechtigungen aufgerufen wurde. Wir speichern die Kamera- und Mikrofonreferenz als `IVSLocalStageStream`-Objekte.

```
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)
}
```

Hier haben wir die Kamera und das Mikrofon des Geräts über das SDK gefunden und sie in unserem lokalen `streams`-Objekt gespeichert, dann zum `streams`-Array des ersten Teilnehmers (des lokalen Teilnehmers, den wir zuvor erstellt haben) zu unserem `streams` zugewiesen. Schließlich rufen wir `participantsChanged` mit einem `index` von 0 und `changeType` von `updated` auf. Diese Funktion ist eine Hilfsfunktion für die Aktualisierung unserer `UICollectionView` mit hübschen Animationen. So sieht es aus:

```
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)])
    }
}
```

Machen Sie sich jetzt keine Sorgen über `cell.set`. Darauf kommen wir später zurück, aber dort werden wir den Inhalt der Zelle basierend auf dem Teilnehmer rendern.

Der `ChangeType` ist eine einfache Aufzählung:

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

Schließlich möchten wir verfolgen, ob die Stage angeschlossen ist. Wir verwenden eine einfache `bool`, um das zu verfolgen, wodurch unsere Benutzeroberfläche automatisch aktualisiert wird, wenn sie selbst aktualisiert wird.

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

Implementierung des Stage-SDK

Drei Kernkonzepte liegen der Echtzeit-Funktionalität zugrunde: Stage, Strategie und Renderer. Das Designziel besteht in der Minimierung der Menge an clientseitiger Logik, die für die Entwicklung eines funktionierenden Produkts erforderlich ist.

IVSStageStrategy

Die `IVSStageStrategy`-Implementierung ist einfach:

```
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
    }
}
```

Zusammenfassend lässt sich sagen, dass wir nur veröffentlichen, wenn die Option „Veröffentlichen“ aktiviert ist, und wenn wir veröffentlichen, veröffentlichen wir die Streams, die wir zuvor gesammelt haben. Schließlich abonnieren wir für dieses Beispiel immer andere Teilnehmer und erhalten sowohl ihr Audio als auch ihr Video.

IVSStageRenderer

Die `IVSStageRenderer`-Implementierung ist ebenfalls ziemlich einfach, obwohl sie angesichts der Anzahl der Funktionen reichlich mehr Code enthält. Der allgemeine Ansatz in diesem Renderer besteht darin, unser `participants`-Array zu aktualisieren, wenn das SDK uns über eine Änderung an einen Teilnehmer informiert. Es gibt bestimmte Szenarien, in denen wir mit lokalen Teilnehmern anders umgehen, weil wir beschlossen haben, sie selbst zu verwalten, sodass sie ihre Kameravorschau sehen können, bevor sie beitreten.

```
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)
    }
}
```

Dieser Code verwendet eine Erweiterung, um den Verbindungsstatus in menschenfreundlichen Text umzuwandeln:

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

Implementierung eines benutzerdefinierten UICollectionViewLayouts

Die Festlegung verschiedener Teilnehmerzahlen kann komplex sein. Sie möchten, dass sie den gesamten Frame der übergeordneten Ansicht einnehmen, aber Sie möchten nicht jede Teilnehmerkonfiguration unabhängig voneinander handhaben. Um dies zu vereinfachen, führen wir die Implementierung eines `UICollectionViewLayout` durch.

Erstellen Sie eine weitere neue Datei, `ParticipantCollectionViewLayout.swift`, welche `UICollectionViewLayout` erweitern soll. Diese Klasse verwendet eine andere Klasse namens `StageLayoutCalculator`, was wir bald behandeln werden. Die Klasse erhält berechnete Rahmenwerte für jeden Teilnehmer und generiert dann die notwendigen `UICollectionViewLayoutAttributes`-Objekte.

```
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))
            }
        }
    }
```

```
}
```

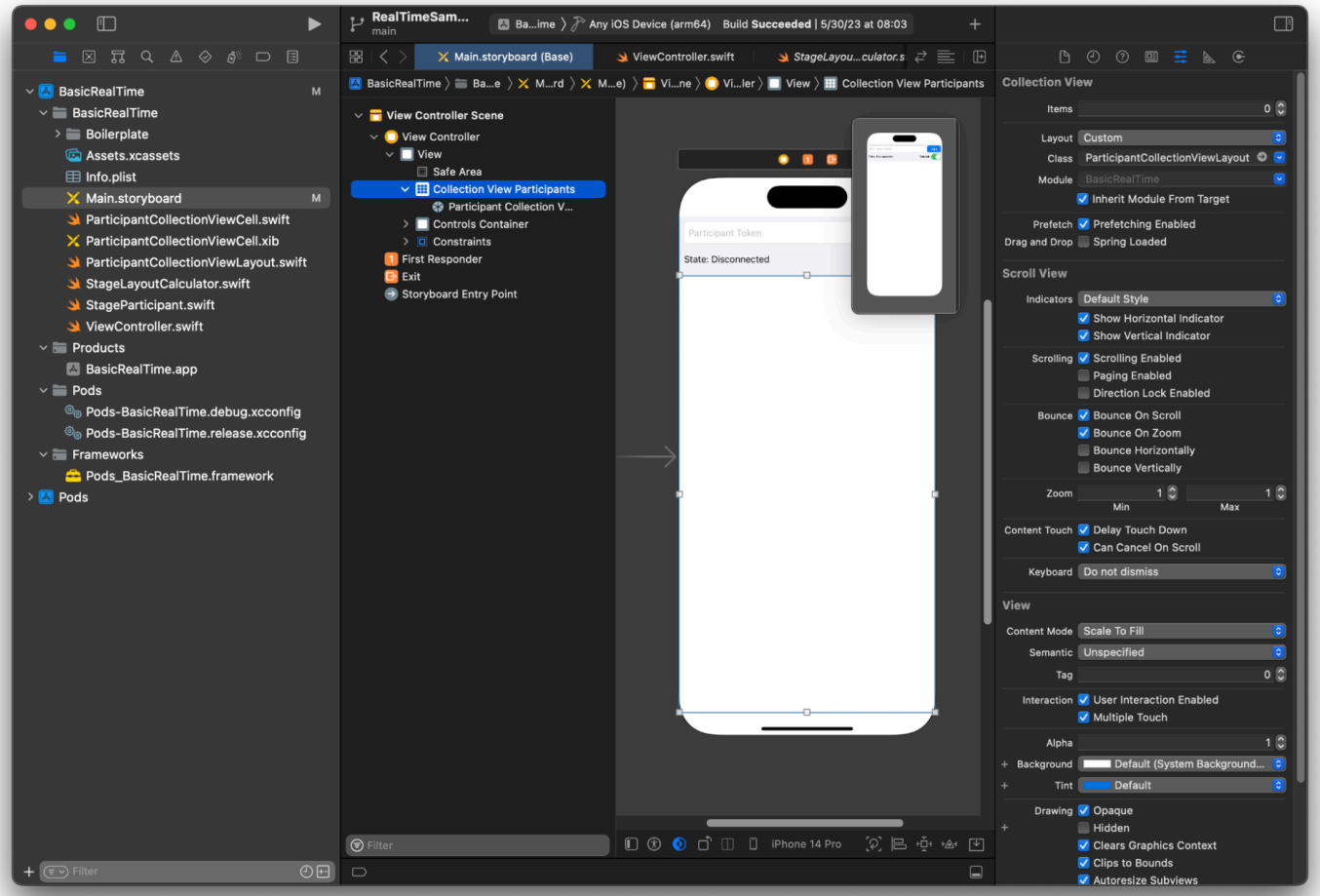
Wichtiger ist die `StageLayoutCalculator.swift`-Klasse. Es ist so konzipiert, dass es die Rahmen für jeden Teilnehmer auf der Grundlage der Anzahl der Teilnehmer in einem fließenden Zeilen-/Spaltenlayout berechnet. Jede Zeile hat dieselbe Höhe wie die anderen, aber Spalten können pro Zeile unterschiedlich breit sein. Sehen Sie den Code-Kommentar über der `layouts`-Variable für eine Beschreibung, wie dieses Verhalten angepasst werden kann.

```
import Foundation
import UIKit

class StageLayoutCalculator {

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

```
        // 7 participants
        [ 2, 2, 3 ], // 3 rows, 1st and 2nd row's columns are 1/2 width, 3rd row's
columns are 1/3rd width
        // 8 participants
        [ 2, 3, 3 ],
        // 9 participants
        [ 3, 3, 3 ],
        // 10 participants
        [ 2, 3, 2, 3 ],
        // 11 participants
        [ 2, 3, 3, 3 ],
        // 12 participants
        [ 3, 3, 3, 3 ],
    ]

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

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

        let halfPadding = padding / 2.0

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

        var frames = [CGRect]()
        for row in 0 ..< layout.count {
            // layout[row] is the number of columns in a layout
            let itemWidth = (isVertical ? width : height) / CGFloat(layout[row])
```

```
            let segmentFrame = CGRect(x: (isVertical ? 0 : lastFrame.maxX) +
 halfPadding,
                                      y: (isVertical ? lastFrame.maxY : 0) +
 halfPadding,
                                      width: (isVertical ? itemWidth : rowHeight) -
 padding,
                                      height: (isVertical ? rowHeight : itemWidth) -
 padding)

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

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

}
```

Wieder in `Main.storyboard`, stellen Sie sicher, dass Sie die Layoutklasse für die `UICollectionView` festlegen zu der Klasse, die wir gerade erstellt haben:



UI-Aktionen verbinden

Wir sind nah dran, es gibt ein paar `IBActions` die wir erstellen müssen.

Zuerst kümmern wir uns um die „Beitreten“-Schaltfläche. Sie reagiert unterschiedlich je nach Wert von `connectingOrConnected`. Wenn sie bereits angeschlossen ist, verlässt sie einfach die Stage. Wenn die Verbindung unterbrochen ist, liest sie den Text aus dem Token `UITextFieldund` schafft eine neue `IVSStage` mit diesem Text. Dann fügen wir unseren `ViewController` hinzu als `strategy`, `errorDelegate`, und Renderer für `IVSStage`, und schließlich treten wir asynchron der Stage bei.

```
@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)")
        }
    }
}
```

Die andere UI-Aktion, die wir anschließen müssen, ist der Publish-Switch:

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

Rendern der Teilnehmer

Schließlich müssen wir die Daten, die wir vom SDK erhalten, in die Teilnehmerzelle rendern, die wir zuvor erstellt haben. Wir haben die Logik von `UICollectionView` bereits fertig, also müssen wir nur noch die API von `set` in `ParticipantCollectionViewCell.swift` implementieren.

Wir beginnen mit dem Hinzufügen der `empty`-Funktion und gehen Sie sie dann Schritt für Schritt durch:

```
func set(participant: StageParticipant) {

}
```

Zuerst kümmern wir uns um den Status „Einfach“, die Teilnehmer-ID, den Veröffentlichungsstatus und den Abonnementstatus. Für diese aktualisieren wir einfach unsere `UILabels` direkt:

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

Die Texteigenschaften der Publish- und Subscribe-Enums stammen aus lokalen Erweiterungen:

```
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()
        }
    }
}
```

Als Nächstes aktualisieren wir die stummgeschalteten Audio- und Videozustände. Um den Stummschaltzustand zu erhalten, müssen wir den `IVSImageDevice` und `IVSAudioDevice` aus dem `streams`-Array finden. Um die Leistung zu optimieren, erinnern wir uns an die zuletzt angehängten Geräte-IDs.

```
// 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)"
```

Abschließend wollen wir eine Vorschau für das `imageDevice` rendern und Audiostatistiken der `audioDevice` anzeigen:

```
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"
}
```

Die letzte Funktion, die wir erstellen müssen, ist `updatePreview()`, was unserer Ansicht eine Vorschau des Teilnehmers hinzufügt:

```
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)
        }
    }
}
```

Das Obige verwendet eine Hilfsfunktion auf `UIView`, um das Einbetten von Unteransichten zu vereinfachen:

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

Überwachung von Amazon IVS Echtzeit-Streaming

Dieses Dokument enthält Einzelheiten zu den verfügbaren Optionen für die Überwachung Ihrer IVS-Echtzeit-Streaming-Anwendung.

Was ist eine Stagesitzung?

Eine Stagesitzung beginnt, wenn der erste Teilnehmer eine Stage betritt und endet einige Minuten, nachdem der letzte Teilnehmer die Veröffentlichung auf der Stage beendet hat. Stagesitzungen helfen bei der Fehlerbehebung in langfristigen Stages, indem Ereignisse und Teilnehmer in kurzlebige Sitzungen aufgeteilt werden.

Stagesitzungen und Teilnehmer anzeigen

Anleitung für die Konsole

1. Öffnen Sie die Amazon-IVS-Konsole.

   (Sie können auf die Amazon-IVS-Konsole auch über die AWS-Managementkonsole zugreifen.)
2. Klicken Sie im Navigationsbereich auf Stage. (Wenn der Navigationsbereich eingeklappt ist, öffnen Sie es zunächst, indem Sie das Hamburger-Symbol auswählen.)
3. Wählen Sie die Stage aus, um ihre Detailseite aufzurufen.
4. Scrollen Sie auf der Seite nach unten, bis Sie den Abschnitt Stagesitzungen sehen, und wählen Sie dann eine Stagesitzung aus, um die zugehörige Detailseite aufzurufen.
5. Um die Teilnehmer der Sitzung anzuzeigen, scrollen Sie nach unten, bis Sie den Abschnitt Teilnehmer sehen. Wählen Sie dann einen Teilnehmer aus, um dessen Detailseite anzuzeigen, einschließlich Diagrammen für Amazon-CloudWatch-Metriken.

Ereignisse für einen Teilnehmer anzeigen

Ereignisse werden gesendet, wenn sich der Status eines Teilnehmers in einer Stage ändert, z. B. wenn er einer Stage beitritt oder beim Versuch, auf einer Stage zu veröffentlichen, ein Fehler auftritt. Nicht alle Fehler verursachen Ereignisse; z. B. werden clientseitige Netzwerkfehler und Tokensignaturfehler nicht als Ereignisse gesendet. Um diese Fehler in Ihrer Client-Anwendung zu behandeln, verwenden Sie dieIVS-Broadcast-SDKs.

Anleitung für die Konsole

1. Navigieren Sie wie oben beschrieben zur Seite mit den Teilnehmerdetails.
2. Scrollen Sie nach unten, bis Sie den Abschnitt Ereignisse sehen. Daraufhin wird eine geordnete Liste der Teilnehmer-Ereignisse angezeigt. Unter Verwendung von Amazon EventBridge mit Amazon IVS finden Sie Details zu Ereignissen, die für Teilnehmer ausgegeben werden.

CLI-Anweisungen

Der Zugriff auf Stagesitzungs-Ereignisse mit der AWS-CLI ist eine erweiterte Option und erfordert, dass Sie zuerst die CLI auf Ihrem Computer herunterladen und konfigurieren. Informationen zu den ersten Schritten finden Sie im AWS-Benutzerhandbuch für die Befehlszeilenschnittstelle.

1. Listen Sie die Stagesitzungen auf, um eine Stagesession zu finden:

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

2. Listen Sie die Teilnehmer für eine Stagesitzung auf, um einen Teilnehmer zu finden:

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

3. Ereignisse für eine Stagesitzung und einen Teilnehmer auflisten:

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

Hier ist eine Beispielantwort auf den `list-participant-events`-Aufruf:

```
{
    "events": [
        {
            "eventTime": "2023-04-04T22:48:41+00:00",
            "name": "JOINED",
            "participantId": "AdRezBl021t0"
        },
        {
            "eventTime": "2023-04-04T22:48:41+00:00",
            "name": "SUBSCRIBE_STARTED",
            "participantId": "AdRezBl021t0",
```

```
            "remoteParticipantId": "Ou5b5n5XLMdC"
        },
        {
            "eventTime": "2023-04-04T22:49:45+00:00",
            "name": "SUBSCRIBE_STOPPED",
            "participantId": "AdRezBl021t0",
            "remoteParticipantId": "Ou5b5n5XLMdC"
        },
        {
            "eventTime": "2023-04-04T22:49:45+00:00",
            "name": "LEFT",
            "participantId": "AdRezBl021t0"
        }
    ]
}
```

Zugreifen auf CloudWatch-Metriken

Damit die CloudWatch-Metriken verfügbar sind, sind die folgenden IVS-Broadcast-SDK-Versionen erforderlich: Web 1.5.0 oder höher, Android 1.12.0 oder höher, oder iOS 1.12.0 oder höher.

Anleitung für die CloudWatch-Konsole

1. Öffnen Sie die CloudWatch-Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Erweitern Sie in der Seitennavigation das Dropdown Metriken und wählen Sie dannAlle Metriken aus.
3. Wählen Sie in der Registerkarte Durchsuchen über das unbeschriftete Dropdown-Menü auf der linken Seite Ihre Heimatregion aus, in der Ihre Kanäle erstellt wurden. Weitere Informationen zu Regionen finden Sie unter Globale Lösung, regionale Kontrolle. Eine Liste der unterstützten Regionen finden Sie auf der Amazon-IVS-Seite in der Allgemeinen AWS-Referenz.
4. Wählen Sie unten in der Registerkarte Durchsuchen den IVSRealTime-Namespace aus.
5. Führen Sie eine der folgenden Aktionen aus:
   a. Geben Sie in der Suchleiste Ihre Ressourcen-ID (Teil der ARN, `arn:::ivs:stage/<resource id>`) ein.

      Wählen Sie dann IVSRealTime > Stage-Metriken aus.
   b. Wenn IVSRealTime als auswählbarer Service unter AWS Namespaces erscheint, wählen Sie ihn aus. Er wird aufgeführt, wenn Sie Amazon-IVS-Streaming in Echtzeit verwenden und es

Metriken an Amazon CloudWatch sendet. (Wenn IVSRealTime nicht aufgeführt ist, haben Sie keine Amazon IVS-Metriken.)

Wählen Sie dann nach Bedarf eine Dimensionsgruppierung aus. Die verfügbaren Dimensionen sind unten in CloudWatch-Metriken aufgeführt.

6. Wählen Sie Metriken aus, die dem Diagramm hinzugefügt werden sollen. Verfügbare Metriken sind unten unter CloudWatch-Metriken aufgeführt.

Sie können auch auf der Detailseite des Stream-Vortrags auf das CloudWatch-Diagramm Ihres Stream-Vortrags zugreifen, indem Sie das Feld Anzeigen in CloudWatch auswählen.

CLI-Anweisungen

Sie können auf die Metriken auch über die AWS CLI zugreifen. Dies erfordert, dass Sie zuerst die CLI auf Ihrem Computer herunterladen und konfigurieren. Informationen zu den ersten Schritten finden Sie im Benutzerhandbuch für die AWS-Befehlszeilenschnittstelle.

So greifen Sie dann über AWS CLI auf das Amazon-IVS-Streaming in Echtzeit zu:

- Führen Sie an der Eingabeaufforderung Folgendes aus:

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

Weitere Informationen finden Sie unter Amazon CloudWatch verwenden im Amazon CloudWatch-Benutzerhandbuch.

CloudWatch-Metriken: IVS-Echtzeit-Streaming

Amazon IVS bietet die folgenden Metriken im AWS/IVSRealTime-Namespace.

Damit CloudWatch-Metriken verfügbar sind, muss Web Broadcast SDK 1.5.2 oder höher verwendet werden.

Die Dimension kann die folgenden gültigen Werte haben:

- Die `Stage`-Dimension ist eine Ressourcen-ID (Teil des ARN, `arn:::stage/<resource id>`).
- Die `Participant`-Dimension ist eine `participantID`.

- Das `SimulcastLayer` ist „hi“, „mid“, „low“ oder „none“ für den `MediaType` „Video“ oder „none“ für den `MediaType` „Audio“. Dieser Wert kann auch leer sein.
- Die `MediaType`-Dimension ist „Video“ oder „Audio“ (Zeichenfolge).

Im Fall der Teilnehmerreplikation umfassen die vorhandenen Metriken zum Stagezustand für die Ziel-Stage alle replizierten Teilnehmer (Publisher in der Quell-Stage, die Replikatteilnehmer in der Ziel-Stage sind).

Metrik	Dimensionen	Beschreibung
`ConcurrentPublishers`	—	Anzahl von Teilnehmern, die Inhalte auf allen Stages in einer AWS-Region veröffentlichen. Einheit: Anzahl Gültige Statistiken: Durchschnitt, Maximum, Minimum
`ConcurrentSubscriptions`	—	Anzahl gleichzeitiger Verbindungen zwischen Publishern und Abonnenten für alle Stages in einer AWS-Region. Einheit: Anzahl Gültige Statistiken: Durchschnitt, Maximum, Minimum
`DownloadPacketLoss`	—	Prozentsatz der Pakete, die beim Herunterladen vom IVS-Server durch Abonnenten verloren gegangen sind. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Paketverluste über das konfigurierte Intervall.
`DownloadPacketLoss`	`Platform`	Filtert `DownloadPacketLoss` nach der Plattform des Abonnenten. Einheit: Prozent

Metrik	Dimensionen	Beschreibung
		Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Paketverluste über das konfigurierte Intervall.
`DownloadPacketLoss`	`Platform, SDKVersion`	Filtert `DownloadPacketLoss` nach der Plattform und SDK-Version des Abonnenten. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Paketverluste über das konfigurierte Intervall.
`DownloadPacketLoss`	`Stage`	Filtert `DownloadPacketLoss` nach der Stage des Abonnenten. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Paketverluste über das konfigurierte Intervall.
`DownloadPacketLoss`	`Stage, Participant`	Filtert `DownloadPacketLoss` nach Teilnehmer, für Subscriber, die auch Publisher sind. Die Beispiele stellen den Prozentsatz der Pakete dar, die vom Subscriber während des Herunterladens vom IVS-Server verloren gingen. Beispiele werden nur ausgegeben, wenn der Teilnehmer auch ein Publisher ist. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Frameverluste über das konfigurierte Intervall.

Metrik	Dimensionen	Beschreibung
`DownloadPacketLoss`	`Stage, Platform`	Filtert `DownloadPacketLoss` nach der Stage und Plattform des Abonnenten. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Paketverluste über das konfigurierte Intervall.
`DownloadPacketLoss`	`Stage, Platform, SDKVersion`	Filtert `DownloadPacketLoss` nach der Stage, Plattform und SDK-Version des Abonnenten. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Paketverluste über das konfigurierte Intervall.
`DownloadPacketLoss`	`Stage, SubscriberCountryCode`	Filtert `DownloadPacketLoss` nach der Stage und dem Ländercode des Abonnenten (ISO 3166). Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Paketverluste über das konfigurierte Intervall.
`DownloadPacketLoss`	`SubscriberCountryCode`	Filtert `DownloadPacketLoss` nach dem Ländercode des Abonnenten (ISO 3166). Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Paketverluste über das konfigurierte Intervall.

Metrik	Dimensionen	Beschreibung
`DroppedFrames`	–	Für Abonnenten: Der Prozentsatz der verlorenen Videoframes, berechnet durch die Aggregation der empfangenen und abgebrochenen Frames für alle Publisher, bei denen der Subscriber abonniert. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Frameverluste über das konfigurierte Intervall.
`DroppedFrames`	`Platform`	Filtert `DroppedFrames` nach der Plattform des Subscribers. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Frameverluste über das konfigurierte Intervall.
`DroppedFrames`	`Platform, SDKVersion`	Filtert `DroppedFrames` nach der Plattform und SDK-Version des Subscribers. Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Frameverluste über das konfigurierte Intervall.
`DroppedFrames`	`Stage`	Filtert `DroppedFrames` nach der Stage. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Frameverluste über das konfigurierte Intervall.

Metrik	Dimensionen	Beschreibung
DroppedFrames	Stage, Participant	Filtert DroppedFrames nach der Stage und dem Teilnehmer. Wird nur für Subscriber ausgegeben, die gleichzeitig Publisher sind. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Frameverluste über das konfigurierte Intervall.
DroppedFrames	Stage, Platform	Filtert DroppedFrames nach der Stage und Plattform des Subscribers. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Frameverluste über das konfigurierte Intervall.
DroppedFrames	Stage, Platform, SDKVersion	Filtert DroppedFrames nach der Stage, Plattform und SDK-Version des Subscribers. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Frameverluste über das konfigurierte Intervall.
DroppedFrames	Stage, SubscriberCountryCode	Filtert DroppedFrames nach der Stage und dem Ländercode des Subscribers. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Frameverluste über das konfigurierte Intervall.

Metrik	Dimensionen	Beschreibung
`DroppedFrames`	`SubscriberCountryCode`	Filtert `DroppedFrames` nach dem Land des Subscribers. Einheit: Prozent Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Frameverluste über das konfigurierte Intervall.
`PublishBitrate`	–	Die Gesamtrate, mit der ein Publisher sowohl Video- als auch Audiodaten sendet (summiert über alle Simulcast-Ebenen). Dies umfasst erneut übertragene Daten. Die Bitrate kann durch Paketverluste beim Hochladen und erneute Übertragungen aufgebläht werden, da sie die vom Publisher gesendeten Daten widerspiegelt und möglicherweise nicht dem entspricht, was IVS empfängt oder an die Abonnenten übermittelt. Bits/Sekunden Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Bitrate über das konfigurierte Intervall.
`PublishBitrate`	`Platform`	Filtert `PublishBitrate` nach der Plattform des Publishers. Bits/Sekunden Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Bitrate über das konfigurierte Intervall.

Metrik	Dimensionen	Beschreibung
`PublishBitrate`	`Stage`	Filtert `PublishBitrate` nach der Stage. Einheit: Bits/Sekunde Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Bitrate über das konfigurierte Intervall.
`PublishBitrate`	`Stage, Participant, SimulcastLayer, MediaType`	Filtert `PublishBitrate` nach Stage, Teilnehmer, Simulcast-Ebene und Medientyp. Die Simulcast-Layer-ID wird vom Broadcast-SDK festgelegt. Wenn Simulcast deaktiviert ist, wird diese Layer-ID auf „deaktiviert“ gesetzt. Der Medientyp ist entweder „Video“ oder „Audio“. Einheit: Bits/Sekunde Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Bitrate über das konfigurierte Intervall.
`Publishers`	`Stage`	Anzahl der Teilnehmer, die auf der Stage veröffentlichen. Einheit: Anzahl Gültige Statistiken: Durchschnitt, Maximum, Minimum
`PublishFramerate`	`Stage, Participant`	Wie oft Videoframes von einem bestimmten Publisher empfangen werden. Diese Metrik ist nur für Teilnehmer verfügbar, die über RTMP veröffentlichen. Einheit: Anzahl/Sekunde Gültige Statistiken: Durchschnitt, Maximum, Minimum – durchschnittliche Anzahl, größte bzw. kleinste Framerate-Anzahl über das konfigurierte Intervall.

Metrik	Dimensionen	Beschreibung
`PublishFramerate`	`Stage`, `Participant`, `SimulcastLayer`, `MediaType`	Wie oft Videoframes von einem bestimmten Publisher empfangen werden. Diese Metrik ist nur für Teilnehmer verfügbar, die über RTMP veröffentlichen. Einheit: Anzahl/Sekunde Gültige Statistiken: Durchschnitt, Maximum, Minimum – durchschnittliche Anzahl, größte bzw. kleinste Framerate-Anzahl über das konfigurierte Intervall.
`PublishResolution`	`Stage`, `Participant`, `SimulcastLayer`, `MediaType`	Anzahl der Pixel in der Breite oder Höhe des Frames, je nachdem, welcher Wert kleiner ist. Für einen Frame im Querformat mit einer Größe von 1920x1080 beträgt die PublishResolution beispielsweise 1080. Für einen Frame im Hochformat mit einer Größe von 720x1280 beträgt die PublishResolution 720. Einheit: Anzahl Gültige Statistiken: Durchschnitt, Maximum, Minimum
`SubscribeBitrate`	–	Die Gesamtrate, mit der ein Subscriber sowohl Video- als auch Audiodaten empfängt. Einheit: Bits/Sekunde Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Bitrate über das konfigurierte Intervall.
`SubscribeBitrate`	`Platform`	Filtert `SubscribeBitrate` nach der Plattform des Subscribers. Einheit: Bits/Sekunde Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Bitrate über das konfigurierte Intervall.

Metrik	Dimensionen	Beschreibung
`SubscribeBitrate`	`Platform`, `SDKVersion`	Filtert `SubscribeBitrate` nach der Plattform und SDK-Version des Subscribers. Einheit: Bits/Sekunde Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Bitrate über das konfigurierte Intervall.
`SubscribeBitrate`	`Stage`	Filtert `SubscribeBitrate` nach der Stage. Einheit: Bits/Sekunde Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Bitrate über das konfigurierte Intervall.
`SubscribeBitrate`	`Stage`, `Participant`, `MediaType`	Filtert `SubscribeBitrate` nach Stage, Teilnehmer und Medientyp. Der Medientyp ist entweder „Video“ oder „Audio“. Diese Metrik nur ausgegeben, während der abonnierende Teilnehmer auch veröffentlicht. Einheit: Bits/Sekunde Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Bitrate über das konfigurierte Intervall.
`SubscribeBitrate`	`Stage`, `Platform`	Filtert `SubscribeBitrate` nach der Stage und Plattform des Subscribers. Bits/Sekunden Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Bitrate über das konfigurierte Intervall.

Metrik	Dimensionen	Beschreibung
`SubscribeBitrate`	`Stage, Platform, SDKVersion`	Filtert `SubscribeBitrate` nach der Stage, Plattform und SDK-Version des Subscribers. Bits/Sekunden Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Bitrate über das konfigurierte Intervall.
`SubscribeBitrate`	`Stage, SubscriberCountryCode`	Filtert `SubscribeBitrate` nach der Stage und dem Ländercode des Subscribers. Bits/Sekunden Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Bitrate über das konfigurierte Intervall.
`SubscribeBitrate`	`SubscriberCountryCode`	Filtert `SubscribeBitrate` nach dem Ländercode des Abonnenten (ISO 3166-1 alpha-2). Bits/Sekunden Gültige Statistiken: Durchschnitt, Maximum, Minimum – Durchschnittliche Anzahl, größte bzw. kleinste Anzahl der Bitrate über das konfigurierte Intervall.
`Subscribers`	`Stage`	Anzahl der Teilnehmer, die Subscriber der Stage sind. Beachten Sie, dass Teilnehmer, die aktiv veröffentlichen und abonnieren, sowohl als Publisher als auch als Subscriber gezählt werden. Einheit: Anzahl Gültige Statistiken: Durchschnitt, Maximum, Minimum

IVS-Broadcast-SDK | Echtzeit-Streaming

Das Amazon Interactive Video Services (IVS)-Broadcast-SDK ist für Entwickler gedacht, die Anwendungen mit Amazon IVS erstellen. Dieses SDK wurde entwickelt, um die Amazon-IVS-Architektur zu nutzen und bietet neben Amazon IVS kontinuierliche Verbesserungen und neue Funktionen. Als natives Broadcast-SDK wurde es entwickelt, um die Leistungsauswirkungen auf Ihre Anwendung und auf die Geräte, mit denen Ihre Benutzer auf Ihre Anwendung zugreifen, zu minimieren.

Beachten Sie, dass das Broadcast-SDK sowohl für das Senden als auch für das Empfangen von Videos verwendet wird. Sie verwenden also dasselbe SDK für Hosts und Zuschauer. Kein separates Player-SDK erforderlich.

Ihre Anwendung kann die wichtigsten Funktionen des Amazon-IVS-Broadcast-SDK nutzen:

- Hochqualitatives Streaming – Das Broadcast-SDK unterstützt qualitativ hochwertiges Streaming. Nehmen Sie Videos von Ihrer Kamera auf und kodieren Sie sie mit bis zu 720p.
- Automatische Bitratenanpassungen – Smartphone-Nutzer sind mobil, so dass sich ihre Netzwerkbedingungen im Laufe einer Sendung ändern können. Das Amazon-IVS-Broadcast-SDK passt die Videobitrate automatisch an sich ändernde Netzwerkbedingungen an.
- Hoch- und Quer-Support – Unabhängig davon, wie Ihre Benutzer ihre Geräte halten, wird das Image mit der rechten Seite nach oben und richtig skaliert angezeigt. Das Broadcast-SDK unterstützt sowohl die Leinwandgröße im Hoch- als auch im Querformat. Es verwaltet automatisch das Seitenverhältnis, wenn die Benutzer ihr Gerät von der konfigurierten Ausrichtung weg drehen.
- Sicheres Streaming – Die Übertragungen Ihrer Benutzer werden mit TLS verschlüsselt, sodass sie ihre Streams sicher halten können.
- Externe Audiogeräte – Das Amazon-IVS-Broadcast-SDK unterstützt externe Audiobuchse, USB und Bluetooth-SCO-Mikrofone.

Plattform-Anforderungen

Native Plattformen

Plattform	Unterstützte Versionen
Android	9.0 und höher – Hinweis: Kunden können mit Version 6.0 und höher entwickeln, können aber die Echtzeit-Streaming-Funktion nicht nutzen.
iOS	14 und höher

IVS unterstützt mindestens 4 Hauptversionen von iOS und 6 Hauptversionen von Android. Unsere aktuelle Versionsunterstützung kann über diese Mindestanforderungen hinausgehen. Kunden werden über SDK-Versionshinweise mindestens 3 Monate im Voraus benachrichtigt, wenn eine Hauptversion nicht mehr unterstützt wird.

Desktop-Browser

Browser	Unterstützte Plattformen	Unterstützte Versionen
Chrome	Windows, macOS	Zwei Hauptversionen (aktuelle und neueste Vorversion)
Firefox	Windows, macOS	Zwei Hauptversionen (aktuelle und neueste Vorversion)
Edge	Windows 8.1 und höher	Zwei Hauptversionen (aktuelle und neueste Vorversion) Schließt Edge Legacy aus
Safari	macOS	Zwei Hauptversionen (aktuelle und neueste Vorversion)

Mobile Browser (iOS und Android)

Browser	Unterstützte Plattformen	Unterstützte Versionen
Chrome	iOS, Android	Zwei Hauptversionen (aktuelle und neueste Vorversion)
Firefox	Android	Zwei Hauptversionen (aktuelle und neueste Vorversion)
Safari	iOS	Zwei Hauptversionen (aktuelle und neueste Vorversion)

Bekannte Beschränkungen

- Auf allen mobilen Webbrowsern empfehlen wir, nicht mehr als drei Publisher gleichzeitig zu veröffentlichen/zu abonnieren, weil es sonst zu Leistungsproblemen kommen kann, die Videoartefakte und schwarze Bildschirme verursachen. Wenn Sie mehr Publisher benötigen, konfigurieren Sie die reine Audio-Veröffentlichung und -Abonnierung.
- Aus Gründen der Leistung und möglicher Abstürze raten wir davon ab, eine Stufe zusammenzustellen und an einen Kanal im Android Mobile Web zu übertragen. Wenn Broadcast-Funktionalität erforderlich ist, integrieren Sie das Android-Broadcast-SDK für IVS-Echtzeit-Streaming.

Webansichten

Das Web-Broadcast-SDK bietet keine Unterstützung für Webviews oder webähnliche Umgebungen (TV, Konsolen usw.). Informationen zu mobilen Implementierungen finden Sie im Broadcast-SDK-Handbuch für Echtzeit-Streaming mit niedriger Latenz für Android und iOS.

Erforderlicher Gerätezugriff

Das Broadcast-SDK erfordert Zugriff auf die Kameras und Mikrofone des Geräts, sowohl auf die im Gerät integrierten als auch auf die über Bluetooth, USB oder eine Audiobuchse angeschlossenen.

Support

Hinweis: Das Broadcast-SDK wird ständig verbessert. Siehe Versionshinweise zu Amazon IVS für verfügbare Versionen und behobene Probleme. Aktualisieren Sie gegebenenfalls Ihre Version des Broadcast-SDK, bevor Sie sich an den Support wenden und prüfen Sie, ob das Problem dadurch behoben wird.

Versionsverwaltung

Die Amazon-IVS-Broadcast-SDKs verwenden Semantisches Versioning.

Nehmen Sie für diese Diskussion an:

- Die neueste Version ist 4.1.3.
- Die neueste Version der vorherigen Hauptversion ist 3.2.4.
- Die neueste Version 1.x ist 1.5.6.

Rückwärtskompatible neue Funktionen werden als Nebenversionen der neuesten Version hinzugefügt. In diesem Fall wird der nächste Satz neuer Funktionen als Version 4.2.0 hinzugefügt.

Rückwärtskompatible, kleinere Fehlerbehebungen werden als Patch-Releases der neuesten Version hinzugefügt. Hier wird der nächste Satz von kleineren Fehlerbehebungen als Version 4.1.4 hinzugefügt.

Rückwärtskompatible, große Fehlerbehebungen werden unterschiedlich behandelt; diese werden zu mehreren Versionen hinzugefügt:

- Patch-Version der neuesten Version. Hier ist das Version 4.1.4.
- Patch-Version der vorherigen Nebenversion. Hier ist das Version 3.2.5.
- Patch-Version der neuesten Version 1.x. Hier ist das Version 1.5.7.

Wichtige Fehlerbehebungen werden vom Amazon IVS-Produktteam definiert. Typische Beispiele sind kritische Sicherheitsupdates und ausgewählte andere Korrekturen, die für Kunden erforderlich sind.

Hinweis: In den obigen Beispielen werden freigegebene Versionen inkrementiert, ohne dass Zahlen übersprungen werden (z. B. von 4.1.3 auf 4.1.4). In Wirklichkeit können eine oder mehrere Patch-Nummern intern bleiben und nicht veröffentlicht werden, so dass die freigegebene Version von 4.1.3 auf, sagen wir, 4.1.6 steigen könnte.

IVS-Broadcast-SDK: Web-Handbuch | Echtzeit-Streaming

Das Web-Broadcast-SDK von IVS gibt Entwicklern die Werkzeuge an die Hand, um interaktive Echtzeit-Erlebnisse im Web zu schaffen. Dieses SDK ist für Entwickler gedacht, die Webanwendungen mit Amazon IVS erstellen.

Das Web-Broadcast-SDK ermöglicht es den Teilnehmern, Videos zu senden und zu empfangen. Das SDK unterstützt die folgenden Vorgänge:

- Einer Stage beitreten
- Medien für andere Teilnehmer auf der Stage veröffentlichen
- Medien anderer Teilnehmer auf der Stage abonnieren
- Auf der Stage veröffentlichte Videos und Audios verwalten und überwachen
- WebRTC-Statistiken für jede Peer-Verbindung beziehen
- Alle Operationen aus dem Web-Broadcast-SDK für IVS-Streaming mit niedriger Latenz

Aktuelle Version des Web-Broadcast-SDK: 1.33.0 (Versionshinweise)

Referenzdokumentation: Informationen zu den wichtigsten Methoden, die im Amazon IVS Web Broadcast SDK verfügbar sind, finden Sie unter https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference. Stellen Sie sicher, dass die neueste Version des SDK ausgewählt ist.

Beispielcode: Die folgenden Beispiele sind ein guter Ausgangspunkt, um schnell mit dem SDK loszulegen:

- Einfache Wiedergabe
- Einfaches Publishing und Abonnieren
- Umfassende Demo zur Echtzeit-Zusammenarbeit in React

Plattformanforderungen: Eine Liste der unterstützten Plattformen finden Sie unter Amazon IVS Broadcast SDK.

Hinweis: Die Veröffentlichung über einen Browser ist für Endbenutzer praktisch, da keine zusätzliche Software installiert werden muss. Die browserbasierte Veröffentlichung unterliegt jedoch den Einschränkungen und Schwankungen der Browser-Umgebungen. Wenn Stabilität für Sie Priorität hat (beispielsweise für Ereignis-Streaming) empfehlen wir generell die Veröffentlichung über eine Quelle

außerhalb des Browsers (z. B. OBS Studio oder andere dedizierte Encoder), die oft direkten Zugriff auf Systemressourcen haben und Browser-Beschränkungen umgehen. Weitere Informationen zu Optionen der Veröffentlichung außerhalb des Browsers finden Sie in der Dokumentation zu Stream Ingest.

Erste Schritte mit dem IVS Web Broadcast SDK | Streaming in Echtzeit

Dieses Dokument führt Sie durch die Schritte zum Einstieg in das Web Broadcast SDK von IVS-Streaming in Echtzeit.

Importe

Die Bausteine für Echtzeit befinden sich in einem anderen Namespace als die Root-Broadcasting-Module.

Verwenden eines Skript-Tags

Das Web Broadcast SDK wird als JavaScript-Bibliothek verteilt und kann unter https://web-broadcast.live-video.net/1.33.0/amazon-ivs-web-broadcast.js abgerufen werden.

Die in den folgenden Beispielen definierten Klassen und Aufzählungen befinden sich im globalen Objekt `IVSBroadcastClient`:

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

Verwenden von npm

So installieren Sie das `npm`-Paket:

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

Die Klassen, Enums und Typen können auch aus dem Paketmodul importiert werden:

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

Serverseitige Rendering-Unterstützung

Die Stagebibliothek des Web-Broadcast-SDK kann nicht in einem serverseitigen Kontext geladen werden, da sie auf Browser-Primitive verweist, die für das Funktionieren der Bibliothek beim Laden

erforderlich sind. Um dieses Problem zu umgehen, laden Sie die Bibliothek dynamisch, wie in der Web-Broadcast-Demo mit „Next“ und „React“ gezeigt.

Berechtigungen anfordern

Ihre App muss die Berechtigung für den Zugriff auf die Kamera und das Mikrofon des Benutzers anfordern und muss über HTTPS bereitgestellt werden. (Das gilt nicht nur für Amazon IVS, sondern für alle Websites, die Zugriff auf Kameras und Mikrofone benötigen.)

Die folgende Beispielfunktion zeigt, wie Sie Berechtigungen für Audio- und Videogeräte anfordern und erfassen können:

```
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.');
   } else if (!permissions.audio) {
       console.error('Failed to get audio permissions.');
   }
}
```

Weitere Informationen finden Sie in der Berechtigungs-API und unter MediaDevices.getUserMedia().

Auflisten der verfügbaren Geräte

Um festzustellen, welche Geräte für die Erfassung verfügbar sind, fragen Sie die Methode MediaDevices.enumerateDevices() des Browsers ab:

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

Abrufen eines MediaStream von einem Gerät

Nachdem Sie die Liste der verfügbaren Geräte erfasst haben, können Sie einen Stream von einer beliebigen Anzahl von Geräten abrufen. Sie können zum Beispiel mit der Methode `getUserMedia()` einen Stream von einer Kamera abrufen.

Wenn Sie angeben möchten, von welchem Gerät der Stream erfasst werden soll, können Sie die `deviceId` im Bereich `audio` oder `video` der Medieneinschränkungen explizit festlegen. Alternativ können Sie die `deviceId` weglassen und Benutzer ihre Geräte über die Eingabeaufforderung des Browsers auswählen lassen.

Zudem können Sie mithilfe der Einschränkungen `width` und `height` eine ideale Kameraauflösung angeben. (Mehr über diese Einschränkungen erfahren Sie hier.) Das SDK wendet automatisch die Einschränkungen für die Breite und Höhe an, die Ihrer maximalen Übertragungsauflösung entsprechen. Es empfiehlt sich jedoch, diese auch selbst anzuwenden, damit das Seitenverhältnis der Quelle nicht geändert wird, nachdem Sie sie dem SDK hinzugefügt haben.

Stellen Sie für Echtzeit-Streaming sicher, dass die Medienauflösung auf 720p beschränkt ist. Insbesondere dürfen Ihre `getUserMedia`- und `getDisplayMedia`-Beschränkungswerte für Breite und Höhe 921 600 (1280*720) nicht überschreiten, wenn sie miteinander multipliziert werden.

```
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 },
});
```

Veröffentlichen und Abonnieren mit dem IVS Web Broadcast SDK | Streaming in Echtzeit

Dieses Dokument führt Sie durch die Schritte zum Veröffentlichen und Abonnieren einer Stage mit dem Web Broadcast SDK von IVS-Streaming in Echtzeit.

Konzepte

Drei Kernkonzepte liegen der Echtzeit-Funktionalität zugrunde: Stage, Strategie und Ereignisse. Das Designziel besteht in der Minimierung der Menge an clientseitiger Logik, die für die Entwicklung eines funktionierenden Produkts erforderlich ist.

Stage

Die Klasse `Stage` ist der Hauptinteraktionspunkt zwischen der Hostanwendung und dem SDK. Sie stellt die Stage selbst dar und dient dazu, der Stage beizutreten und sie zu verlassen. Für das Erstellen einer Stage und das Beitreten ist eine gültige, noch nicht abgelaufene Token-Zeichenfolge aus der Steuerebene erforderlich (dargestellt als `token`). Einer Stage beizutreten und sie zu verlassen, ist ganz einfach:

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

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

stage.leave();
```

Strategie

Über die Schnittstelle `StageStrategy` kann die Hostanwendung dem SDK den gewünschten Status der Stage mitteilen. Drei Funktionen müssen implementiert

werden: `shouldSubscribeToParticipant`, `shouldPublishParticipant` und `stageStreamsToPublish`. Alle werden im Folgenden behandelt.

Um eine definierte Strategie zu verwenden, übergeben Sie sie an den `Stage`-Konstruktor. Im Folgenden finden Sie ein vollständiges Beispiel für eine Anwendung, die mithilfe einer Strategie die Webcam eines Teilnehmers auf der Stage veröffentlicht und alle Teilnehmer abonniert. Der Zweck der einzelnen erforderlichen Strategiefunktionen wird in den folgenden Abschnitten ausführlich erläutert.

```
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();
```

Abonnieren von Teilnehmern

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

Wenn ein Remote-Teilnehmer der Stage beitritt, fragt das SDK die Hostanwendung nach dessen gewünschtem Abonnementstatus. Die Optionen lauten `NONE`, `AUDIO_ONLY` und `AUDIO_VIDEO`. Wenn ein Wert für diese Funktion zurückgegeben wird, muss sich die Hostanwendung nicht um den Veröffentlichungs-, den aktuellen Abonnement- oder den Verbindungsstatus des Stage kümmern. Bei Rückgabe von `AUDIO_VIDEO` wartet das SDK mit dem Abonnieren, bis der Remote-Teilnehmer etwas veröffentlicht. Außerdem aktualisiert es die Hostanwendung, indem es während des gesamten Prozesses Ereignisse ausgibt.

Hier folgt ein Beispiel für eine Implementierung:

```
const strategy = {

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

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

Hierbei handelt es sich um die vollständige Implementierung dieser Funktion für eine Hostanwendung, bei der sich alle Teilnehmer stets gegenseitig sehen sollen; z. B. eine Video-Chat-Anwendung.

Weitergehende Implementierungen sind ebenfalls möglich. Beispiel: Angenommen, die Anwendung stellt bei der Erstellung des Tokens mit CreateParticipantToken ein `role`-Attribut bereit. Die

Anwendung könnte die `attributes`-Eigenschaft für `StageParticipantInfo` nutzen, um Teilnehmer anhand der vom Server bereitgestellten Attribute selektiv zu abonnieren:

```
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
}
```

Hiermit kann eine Stage erstellt werden, auf der Moderatoren alle Gäste überwachen können, ohne selbst gesehen oder gehört zu werden. Die Hostanwendung könnte eine zusätzliche Geschäftslogik nutzen, damit Moderatoren sich gegenseitig sehen können, für Gäste aber unsichtbar bleiben.

Konfiguration für das Abonnieren von Teilnehmern

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

Wenn ein Remote-Teilnehmer abonniert wird (siehe Teilnehmer abonnieren), fragt das SDK die Host-Anwendung nach einer benutzerdefinierten Abonnementkonfiguration für diesen Teilnehmer ab. Diese Konfiguration ist optional und ermöglicht es der Hostanwendung, bestimmte Aspekte des Subscriber-Verhaltens zu steuern. Informationen darüber, was konfiguriert werden kann, finden Sie unter SubscribeConfiguration in der SDK-Referenzdokumentation.

Hier folgt ein Beispiel für eine Implementierung:

```
const strategy = {

   subscribeConfiguration: (participant) => {
      return {
         jitterBuffer: {
            minDelay: JitterBufferMinDelay.MEDIUM
         }
```

```
      }

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

Diese Implementierung aktualisiert die Mindestverzögerung für den Jitter-Buffer für alle abonnierten Teilnehmer auf die Voreinstellung `MEDIUM`.

Wie bei `shouldSubscribeToParticipant` sind auch hier weitergehende Implementierungen möglich. Die `ParticipantInfo`-Angaben können verwendet werden, um die Abonnementkonfiguration für bestimmte Teilnehmer selektiv zu aktualisieren.

Wir empfehlen die Verwendung der Standardverhaltensweisen. Geben Sie die benutzerdefinierte Konfiguration nur an, wenn Sie ein bestimmtes Verhalten ändern möchten.

Veröffentlichen

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

Sobald die Verbindung zur Stage hergestellt ist, überprüft das SDK per Anfrage an die Hostanwendung, ob ein bestimmter Teilnehmer etwas veröffentlichen soll. Dies wird nur bei lokalen Teilnehmern aufgerufen, die auf Grundlage des bereitgestellten Tokens zur Veröffentlichung berechtigt sind.

Hier folgt ein Beispiel für eine Implementierung:

```
const strategy = {

   shouldPublishParticipant: (participant) => {
      return true;
   }

   // . . . other strategies properties
}
```

Sie ist für eine normale Video-Chat-Anwendung gedacht, bei der Benutzer immer etwas veröffentlichen möchten. Sie können die Audio- und Videowiedergabe stummschalten und die Stummschaltung aufheben, um umgehend ausgeblendet oder gesehen/gehört zu werden. (Sie können auch „Veröffentlichen/Veröffentlichung aufheben“ verwenden, was aber viel langsamer ist.

„Stummschalten/Stummschalten aufheben“ ist für Anwendungsfälle vorzuziehen, in denen eine häufige Änderung der Sichtbarkeit wünschenswert ist.)

Auswählen von Streams zur Veröffentlichung

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

Beim Veröffentlichen wird hiermit bestimmt, welche Audio- und Videostreams veröffentlicht werden sollen. Dieser Punkt wird später unter Veröffentlichen eines Medienstreams ausführlicher behandelt.

Aktualisieren der Strategie

Die Strategie soll dynamisch sein: Die von einer der oben genannten Funktionen zurückgegebenen Werte lassen sich jederzeit ändern. Wenn die Hostanwendung beispielsweise erst veröffentlichen soll, wenn der Endbenutzer auf eine Schaltfläche tippt, können Sie eine Variable aus `shouldPublishParticipant` zurückgeben (zum Beispiel `hasUserTappedPublishButton`). Wenn sich diese Variable aufgrund einer Interaktion des Endbenutzers ändert, signalisieren Sie dem SDK per Aufruf von `stage.refreshStrategy()`, dass es die Strategie nach den neuesten Werten abfragen und nur Dinge anwenden soll, die sich geändert haben. Wenn das SDK feststellt, dass sich der Wert `shouldPublishParticipant` geändert hat, startet es den Veröffentlichungsprozess. Wenn alle Funktionen bei einer SDK-Abfrage den gleichen Wert zurückgeben wie zuvor, wird die Stage mit dem Aufruf von `refreshStrategy` nicht geändert.

Ändert sich der Rückgabewert von `shouldSubscribeToParticipant` von `AUDIO_VIDEO` in `AUDIO_ONLY`, wird der Videostream für alle Teilnehmer mit geänderten Rückgabewerten entfernt, sofern zuvor ein Videostream vorhanden war.

Im Allgemeinen nutzt die Stage die Strategie, um den Unterschied zwischen der vorherigen und der aktuellen Strategie am effizientesten anzuwenden. Dabei muss sich die Hostanwendung nicht um die ganzen Status kümmern, die für eine ordnungsgemäße Verwaltung erforderlich sind. Stellen Sie sich den Aufruf von `stage.refreshStrategy()` daher als einen ressourcenschonenden Vorgang vor, da nur bei einer Änderung der Strategie etwas unternommen wird.

Ereignisse

Eine `Stage`-Instance ist ein Ereignis-Emitter. Mit `stage.on()` wird der Hostanwendung der Status der Stage mitgeteilt. Aktualisierungen in der Benutzeroberfläche der Hostanwendung können in der Regel vollständig durch die Ereignisse unterstützt werden. Folgende Ereignisse werden unterstützt:

```
stage.on(StageEvents.STAGE_CONNECTION_STATE_CHANGED, (state) => {})
```

```
stage.on(StageEvents.STAGE_PARTICIPANT_JOINED, (participant) => {})
stage.on(StageEvents.STAGE_PARTICIPANT_LEFT, (participant) => {})
stage.on(StageEvents.STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED, (participant, state) =>
 {})
stage.on(StageEvents.STAGE_PARTICIPANT_SUBSCRIBE_STATE_CHANGED, (participant, state) =>
 {})
stage.on(StageEvents.STAGE_PARTICIPANT_STREAMS_ADDED, (participant, streams) => {})
stage.on(StageEvents.STAGE_PARTICIPANT_STREAMS_REMOVED, (participant, streams) => {})
stage.on(StageEvents.STAGE_STREAM_ADAPTION_CHANGED, (participant, stream, isAdapting)
 => ())
stage.on(StageEvents.STAGE_STREAM_LAYERS_CHANGED, (participant, stream, layers) => ())
stage.on(StageEvents.STAGE_STREAM_LAYER_SELECTED, (participant, stream, layer, reason)
 => ())
stage.on(StageEvents.STAGE_STREAM_MUTE_CHANGED, (participant, stream) => {})
stage.on(StageEvents.STAGE_STREAM_SEI_MESSAGE_RECEIVED, (participant, stream) => {})
```

Für die meisten dieser Ereignisse wird die entsprechende `ParticipantInfo` bereitgestellt.

Es wird nicht erwartet, dass sich die von den Ereignissen bereitgestellten Informationen auf die Rückgabewerte der Strategie auswirken. Es wird beispielsweise nicht erwartet, dass sich der Rückgabewert von `shouldSubscribeToParticipant` beim Aufruf von `STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED` ändert. Wenn die Hostanwendung einen bestimmten Teilnehmer abonnieren möchte, muss sie unabhängig von dessen Veröffentlichungsstatus den gewünschten Abonnementtyp zurückgeben. Das SDK muss dafür sorgen, dass entsprechend dem Status der Stage und dem gewünschten Status der Strategie zum richtigen Zeitpunkt gehandelt wird.

Veröffentlichen eines Medienstreams

Lokale Geräte wie Mikrofone und Kameras werden mit den gleichen Schritten abgerufen, die oben unter Abrufen eines MediaStream von einem Gerät beschrieben sind. In dem Beispiel erstellen wir mit `MediaStream` eine Liste von `LocalStageStream`-Objekten, die für die Veröffentlichung durch das SDK verwendet werden:

```
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
    }
}
```

Veröffentlichen einer Bildschirmfreigabe

Häufig müssen Anwendungen zusätzlich zur Webkamera des Benutzers eine Bildschirmfreigabe veröffentlichen. Das Veröffentlichen einer Bildschirmfreigabe erfordert die Erstellung eines zusätzlichen Tokens für die Stage, insbesonder für die Veröffentlichung der Medien der Bildschirmfreigabe. Verwenden Sie `getDisplayMedia` und beschränken Sie die Auflösung auf maximal 720p. Danach sind die Schritte ähnlich wie beim Veröffentlichen einer Kamera für die Stage.

```
// 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();
```

Anzeigen und Entfernen von Teilnehmern

Nach Abschluss von Abonnements erhalten Sie über das Ereignis `STAGE_PARTICIPANT_STREAMS_ADDED` eine Reihe von `StageStream`-Objekten. Zudem stellt das Ereignis Teilnehmerinformationen bereit, die Ihnen beim Anzeigen von Medienstreams helfen:

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

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

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

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

Wenn ein Teilnehmer die Veröffentlichung beendet oder dessen Abonnement eines Streams beendet wird, wird die Funktion `STAGE_PARTICIPANT_STREAMS_REMOVED` mit den Streams aufgerufen, die entfernt wurden. Hostanwendungen sollten dies als Signal nutzen, um den Videostream des Teilnehmers aus dem DOM zu entfernen.

`STAGE_PARTICIPANT_STREAMS_REMOVED` wird für alle Szenarien aufgerufen, in denen ein Stream entfernt werden könnte, darunter:

- Der Remote-Teilnehmer beendet die Veröffentlichung.
- Ein lokales Gerät beendet das Abonnement oder ändert das Abonnement von `AUDIO_VIDEO` in `AUDIO_ONLY`.
- Der Remote-Teilnehmer verlässt die Stage.
- Der lokale Teilnehmer verlässt die Stage.

Da `STAGE_PARTICIPANT_STREAMS_REMOVED` bei allen Szenarien aufgerufen wird, ist keine benutzerdefinierte Geschäftslogik erforderlich, um Teilnehmer beim remoten oder lokalen Verlassen aus der Benutzeroberfläche zu entfernen.

Stummschalten von Medienstreams und Aufheben der Stummschaltung

`LocalStageStream`-Objekte verfügen über eine `setMuted`-Funktion, die das Stummschalten des Streams steuert. Diese Funktion kann für den Stream aufgerufen werden, bevor oder nachdem er von der Strategiefunktion `stageStreamsToPublish` zurückgegeben wird.

Wichtig: Wenn nach einem Aufruf von `refreshStrategy` eine neue `LocalStageStream`-Objekt-Instance von `stageStreamsToPublish` zurückgegeben wird, wird der Stummschaltungsstatus des neuen Streamobjekts auf die Stage angewendet. Seien Sie vorsichtig beim Erstellen neuer `LocalStageStream`-Instances, um sicherzustellen, dass der erwartete Stummschaltungsstatus beibehalten wird.

Überwachen des Medien-Stummschaltungsstatus von Remote-Teilnehmern

Wenn Teilnehmer den Stummschaltungsstatus ihres Videos oder Audios ändern, wird das Ereignis `STAGE_STREAM_MUTE_CHANGED` mit einer Liste der Streams ausgelöst, die sich geändert haben. Verwenden Sie die Eigenschaft `isMuted` für `StageStream`, um die Benutzeroberfläche entsprechend zu aktualisieren:

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

Sie können auch unter StageParticipantInfo nach Statusinformationen darüber suchen, ob Audio oder Video stummgeschaltet sind:

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

Abrufen von WebRTC-Statistiken

Die Methode `requestQualityStats()` bietet Zugriff auf detaillierte WebRTC-Statistiken für lokale und Remote-Streams. Sie stehen für LocalStageStream- und für RemoteStageStream-Objekte zur Verfügung. Die Methode gibt umfassende Qualitätsmetriken zurück, darunter Netzwerkqualität, Paketstatistiken, Informationen zur Bitrate und einzelbildbezogene Metriken.

Hierbei handelt es sich um eine asynchrone Methode, mit der Sie Statistiken entweder über await oder durch Verkettung eines Promise abrufen können. Sind keine Statistiken verfügbar, gibt sie `undefined` zurück, z. B. wenn der Stream nicht aktiv ist oder interne Statistiken nicht verfügbar sind. Wenn Statistiken verfügbar sind, gibt die Methode je nach Stream (remote oder lokal, Video oder Audio) das Objekt LocalVideoStats, LocalAudioStats, RemoteVideoStats oder RemoteAudioStats zurück.

Beachten Sie, dass das Array bei Videostreams mit Simulcast mehrere Statistikobjekte enthält (eines pro Schicht).

Bewährte Methoden

- Abfragehäufigkeit – Rufen Sie `requestQualityStats()` in angemessenen Intervallen (1 bis 5 Sekunden) auf, um Leistungseinbußen zu vermeiden.
- Fehlerbehandlung – Prüfen Sie vor der Verarbeitung stets, ob der zurückgegebene Wert `undefined` lautet.
- Verwaltung des Arbeitsspeichers – Löschen Sie Intervalle/Timeouts, wenn Streams nicht mehr benötigt werden.
- Netzwerkqualität – Nutzen Sie `networkQuality` für Benutzerfeedback zu möglichen Beeinträchtigungen, die vom Netzwerk verursacht werden. Einzelheiten finden Sie unter NetworkQuality.

Beispielverwendung

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

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

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

Optimieren von Medien

Für eine optimale Leistung wird empfohlen, Aufrufe von `getUserMedia` und `getDisplayMedia` entsprechend den folgenden Einschränkungen zu begrenzen:

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

Sie können die Medien durch zusätzliche Optionen, die an den `LocalStageStream`-Konstruktor übergeben werden, weiter einschränken:

```
const localStreamOptions = {
    minBitrate?: number;
    maxBitrate?: number;
    maxFramerate?: number;
    simulcast: {
```

```
        enabled: boolean
    }
}
const localStream = new LocalStageStream(track, localStreamOptions)
```

Im obigen Code:

- `minBitrate` legt eine Mindestbitrate fest, die der Browser voraussichtlich verwenden sollte. Ein Videostream mit geringer Komplexität kann jedoch dazu führen, dass der Encoder diese Bitrate unterschreitet.
- `maxBitrate` legt eine maximale Bitrate fest, von der erwartet werden sollte, dass sie vom Browser für diesen Stream nicht überschritten wird.
- `maxFramerate` legt eine maximale Framerate fest, von der erwartet werden sollte, dass sie vom Browser für diesen Stream nicht überschritten wird.
- Die Option `simulcast` ist nur in Chromium-basierten Browsern verwendbar. Sie ermöglicht das Senden von drei Wiedergabeebenen des Streams.
  - Auf diese Weise kann der Server anhand ihrer Netzwerkbeschränkungen auswählen, welche Wiedergabeversion an andere Teilnehmer gesendet werden soll.
  - Wenn `simulcast` zusammen mit einem `maxBitrate` und/oder `maxFramerate` Wert angegeben wird, wird erwartet, dass die höchste Wiedergabe-Ebene unter Berücksichtigung dieser Werte konfiguriert wird, vorausgesetzt, `maxBitrate` unterschreitet nicht die Standardeinstellung der zweithöchsten Ebene des internen SDK-Standardwerts `maxBitrate` von 900 kbps.
  - Wenn `maxBitrate` im Vergleich zum Standardwert der zweithöchsten Ebene als zu niedrig angegeben wird, wird `simulcast` deaktiviert.
  - `simulcast` kann nicht ein- und ausgeschaltet werden, ohne die Medien erneut zu veröffentlichen, indem `shouldPublishParticipant` `false` zurückgibt, `refreshStrategy` aufruft, `shouldPublishParticipant` `true` zurückgibt, und `refreshStrategy` wieder aufruft.

Abrufen von Teilnehmerattributen

Wenn Sie Attribute in der Vorgangsanfrage `CreateParticipantToken` angeben, können Sie die Attribute in den Eigenschaften von `StageParticipantInfo` einsehen:

```
stage.on(StageEvents.STAGE_PARTICIPANT_JOINED, (participant) => {
```

```
    console.log(`Participant ${participant.id} info:`, participant.attributes);
})
```

SEI (Supplemental Enhancement Information, Ergänzende Informationen zur Verbesserung)

Die NAL-Einheit für Supplemental Enhancement Information (SEI) wird verwendet, um Frame-orientierte Metadaten zusammen mit dem Video zu speichern. Sie können beim Veröffentlichen und Abonnieren von H.264-Videostreams verwendet werden. Es kann nicht garantiert werden, dass SEI-Nutzdaten bei Subscribern ankommen, insbesondere bei schlechten Netzwerkbedingungen. Da die SEI-Nutzdaten direkt in der H.264-Frame-Struktur gespeichert werden, kann diese Funktion nicht für reine Audio-Streams genutzt werden.

Einfügen von SEI-Nutzdaten

Veröffentlichende Clients können SEI-Nutzdaten in einen Stage-Stream einfügen, der gerade veröffentlicht wird, indem sie den LocalStageStream ihres Videos so konfigurieren, dass `inBandMessaging` aktiviert wird, und anschließend die Methode `insertSeiMessage` aufrufen. Beachten Sie, dass die Aktivierung von `inBandMessaging` die SDK-Speichernutzung erhöht.

Nutzdaten müssen vom Typ ArrayBuffer sein. Die Nutzdaten müssen größer als 0 KB und kleiner als 1 KB sein. Die Anzahl der pro Sekunde eingefügten SEI-Nachrichten darf 10 KB pro Sekunde nicht überschreiten.

```
const config = {
    inBandMessaging: { enabled: true }
};
const vidStream = new LocalStageStream(videoTrack, config);
const payload = new TextEncoder().encode('hello world').buffer;
vidStream.insertSeiMessage(payload);
```

Sich wiederholende SEI-Nutzdaten

Geben Sie optional eine `repeatCount` an, um das Einfügen von SEI-Nutzdaten für die nächsten N gesendeten Frames zu wiederholen. Dies könnte hilfreich sein, um den inhärenten Verlust zu verringern, der aufgrund des zugrunde liegenden UDP-Transportprotokolls entstehen kann, das zum Senden von Videos verwendet wird. Beachten Sie, dass dieser Wert zwischen 0 und 30 liegen muss. Empfangende Clients müssen über eine Logik verfügen, um die Nachricht zu deduplizieren.

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

Lesen von SEI-Nutzdaten

Abonnierende Clients können SEI-Nutzdaten von einem Publisher lesen, der H.264-Videos veröffentlicht, sofern vorhanden. Dazu wird das `SubscribeConfiguration`-Element der Subscriber für die Aktivierung von `inBandMessaging` konfiguriert und auf das `StageEvents.STAGE_STREAM_SEI_MESSAGE_RECEIVED`-Ereignis gelauscht, wie im folgenden Beispiel gezeigt:

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

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

Mehrschichtige Kodierung mit Simulcast

Bei der mehrschichtigen Kodierung mit Simulcast handelt es sich um ein Feature für IVS-Echtzeit-Streaming, mit dessen Hilfe Publisher mehrere Videoschichten unterschiedlicher Qualität senden können. Subscriber können diese Schichten dynamisch oder manuell ändern. Das Feature wird im Dokument Streaming-Optimierungen ausführlicher beschrieben.

Konfigurieren mehrschichtiger Kodierung (Publisher)

Um als Publisher die mehrschichtige Kodierung mit Simulcast zu aktivieren, fügen Sie dem `LocalStageStream` bei der Instanziierung die folgende Konfiguration hinzu:

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

```
})
```

Je nach Eingangsauflösung der Kamera wird eine festgelegte Anzahl von Schichten kodiert und gesendet, wie im Abschnitt Standardmäßige Schichten, Qualitäten und Bildraten von Streaming-Optimierungen definiert.

Außerdem können Sie optional einzelne Ebenen innerhalb der Simulcast-Konfiguration konfigurieren:

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

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

Alternativ können Sie eigene benutzerdefinierte Ebenenkonfigurationen für bis zu drei Ebenen erstellen. Wenn Sie ein leeres Array oder keinen Wert angeben, werden die oben beschriebenen Standardwerte verwendet. Ebenen werden mit den folgenden erforderlichen Eigenschaften beschrieben:

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

Ausgehend von den Voreinstellungen können Sie entweder einzelne Eigenschaften überschreiben oder eine völlig neue Konfiguration erstellen:

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

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

```
}

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

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

Informationen zu Höchstwerten, Grenzwerten und Fehlern, die bei der Konfiguration einzelner Ebenen ausgelöst werden können, finden Sie in der SDK-Referenzdokumentation.

Konfigurieren mehrschichtiger Kodierung (Subscriber)

Subscriber müssen nichts unternehmen, um die mehrschichtige Kodierung zu aktivieren. Wenn ein Publisher Simulcast-Schichten sendet, passt sich der Server standardmäßig dynamisch den Schichten an, um je nach Gerät und Netzwerkbedingungen des Subscribers die optimale Qualität auszuwählen.

Alternativ gibt es mehrere nachfolgend beschriebene Optionen, um explizite Schichten auszuwählen, die der Publisher sendet.

Option 1: Einstellung für die Qualität der Anfangsschicht

Mit der Strategie `subscribeConfiguration` können Sie auswählen, welche Anfangsschicht Sie als Subscriber erhalten möchten:

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

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

Standardmäßig wird Subscribern zunächst immer die Schicht mit der niedrigsten Qualität gesendet. Nach und nach wird die Qualität gesteigert, bis die Schicht mit der höchsten Qualität erreicht ist. Das optimiert den Bandbreitenverbrauch der Endbenutzer, verkürzt die Zeit bis zum Abspielen des Videos und verringert das anfängliche Einfrieren von Videos bei Benutzern in Netzwerken mit geringerer Bandbreite.

Folgende Optionen sind für `InitialLayerPreference` verfügbar:

- `LOWEST_QUALITY` – Der Server stellt zuerst die Videoschicht mit der niedrigsten Qualität bereit. Dadurch werden der Bandbreitenverbrauch und die Zeit bis zum Abspielen von Medien optimiert. Die Qualität ist definiert als die Kombination aus Größe, Bitrate und Bildrate des Videos. Beispielsweise weisen 720p-Videos eine geringere Qualität auf als 1080p-Videos.
- `HIGHEST_QUALITY` – Der Server stellt zuerst die Videoschicht mit der höchsten Qualität bereit. Das optimiert die Qualität, kann aber die Zeit bis zum Abspielen von Medien verlängern. Die Qualität ist definiert als die Kombination aus Größe, Bitrate und Bildrate des Videos. Beispielsweise weisen 1080p-Videos eine höhere Qualität auf als 720p-Videos.

Hinweis: Damit die anfänglichen Schichteinstellungen (der Aufruf `initialLayerPreference`) wirksam werden, muss ein neues Abonnement abgeschlossen werden, da diese Updates für das aktive Abonnement nicht gelten.

Option 2: Bevorzugte Schicht für Streams

Sobald ein Stream gestartet wurde, können Sie die Strategiemethode `preferredLayerForStream` nutzen. Diese Strategiemethode legt den Teilnehmer und die Stream-Informationen offen.

Die Strategiemethode kann mit folgenden Elementen zurückgegeben werden:

- dem Schichtobjekt direkt, basierend auf der Rückgabe von `RemoteStageStream.getLayers`
- der Bezeichnung des Schichtobjekts, basierend auf `StageStreamLayer.label`
- undefiniert oder null, was bedeutet, dass keine Schicht ausgewählt werden sollte und eine dynamische Anpassung bevorzugt wird

Bei der folgenden Strategie wählen die Benutzer beispielsweise immer die Videoschicht mit der niedrigsten verfügbaren Qualität aus:

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

Um die Schichtauswahl zurückzusetzen und zur dynamischen Anpassung zurückzukehren, geben Sie in der Strategie null oder undefiniert zurück. In diesem Beispiel ist `appState` eine Dummy-Variable, die den möglichen Anwendungsstatus darstellt.

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

Option 3: Helferobjekte für RemoteStageStream-Schicht

`RemoteStageStream` weist mehrere Helferobjekte auf, mit deren Hilfe Entscheidungen über die Schichtauswahl getroffen und Endbenutzern die entsprechende Auswahl angezeigt werden kann:

- Schichtereignisse – Neben `StageEvents` verfügt das Objekt `RemoteStageStream` selbst über Ereignisse, die Änderungen bei der Schicht- und Simulcast-Anpassung kommunizieren:
  - `stream.on(RemoteStageStreamEvents.ADAPTION_CHANGED, (isAdapting) => {})`
  - `stream.on(RemoteStageStreamEvents.LAYERS_CHANGED, (layers) => {})`
  - `stream.on(RemoteStageStreamEvents.LAYER_SELECTED, (layer, reason) => {})`
- Schichtmethoden – `RemoteStageStream` verfügt über mehrere Helfermethoden, mit denen Informationen über den Stream und die präsentierten Schichten abgerufen werden können. Diese Methoden sind sowohl für den in der Strategie `preferredLayerForStream`

bereitgestellten Remote-Stream als auch für Remote-Streams verfügbar, die über `StageEvents.STAGE_PARTICIPANT_STREAMS_ADDED` verfügbar gemacht werden.

- `stream.getLayers`
- `stream.getSelectedLayer`
- `stream.getLowestQualityLayer`
- `stream.getHighestQualityLayer`

Einzelheiten finden Sie im Abschnitt zur Klasse `RemoteStageStream` in der SDK-Referenzdokumentation. Falls als Grund für `LAYER_SELECTED UNAVAILABLE` zurückgegeben wird, bedeutet das, dass die angeforderte Schicht nicht ausgewählt werden konnte. Stattdessen wird eine bestmögliche Auswahl getroffen. Dabei handelt es sich in der Regel um eine Schicht mit niedrigerer Qualität, um die Stabilität des Streams zu gewährleisten.

Umgang mit Netzwerkproblemen

Bei Unterbrechung der Netzwerkverbindung des lokalen Geräts versucht das SDK intern, die Verbindung ohne Benutzeraktion wiederherzustellen. In einigen Fällen ist das SDK nicht erfolgreich, weshalb eine Benutzeraktion erforderlich ist.

Generell kann der Status der Stage über das Ereignis `STAGE_CONNECTION_STATE_CHANGED` gesteuert werden:

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

Im Allgemeinen können Sie einen fehlerhaften Status ignorieren, der nach dem erfolgreichen Beitritt zu einer Stage auftritt, da das SDK versuchen wird, die Verbindung intern wiederherzustellen. Wenn das SDK einen `ERRORED`-Status meldet und die Stage über einen längeren Zeitraum (z. B. 30 Sekunden oder länger) im `CONNECTING`-Status verbleibt, sind Sie wahrscheinlich vom Netzwerk getrennt worden.

Übertragung der Stage auf einen IVS-Kanal

Zum Übertragen einer Stage erstellen Sie eine separate `IVSBroadcastClient`-Sitzung und folgen Sie dann den oben beschriebenen üblichen Anweisungen für die Übertragung mit dem SDK. Mithilfe der Liste der über `STAGE_PARTICIPANT_STREAMS_ADDED` offengelegten `StageStream` können die Medienstreams der Teilnehmer abgerufen werden, die wie folgt auf die Zusammensetzung der übertragenen Streams angewendet werden können:

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

Optional können Sie eine Stage zusammenstellen und sie auf einen IVS-Kanal mit niedriger Latenz übertragen, um ein größeres Publikum zu erreichen. Sehen Sie Aktivierung mehrerer Hosts in einem Amazon-IVS-Stream im Benutzerhandbuch für IVS-Streaming mit niedriger Latenz.

Bekannte Probleme und Behelfslösungen im IVS Web Broadcast SDK | Streaming in Echtzeit

In diesem Dokument werden bekannte Probleme aufgeführt, die bei der Verwendung des Web Broadcast SDK von Amazon-IVS-Streaming in Echtzeit auftreten können, und es werden mögliche Problemumgehungen vorgeschlagen.

- Wenn Browser-Tabs oder Browser ohne Aufruf von `stage.leave()` geschlossen werden, können Benutzer noch bis zu 10 Sekunden lang mit einem eingefrorenen Frame oder einem schwarzen Bildschirm in der Sitzung zu sehen sein.

  Problemumgehung: Keine.

- Safari-Sitzungen werden für Benutzer, die nach Beginn einer Sitzung beitreten, mitunter mit einem schwarzen Bildschirm angezeigt.

  Problemumgehung: Aktualisieren Sie den Browser und stellen Sie die Verbindung zur Sitzung erneut her.

- Safari stellt Sitzungen bei einem Netzwerkwechsel nicht ordnungsgemäß wieder her.

  Problemumgehung: Aktualisieren Sie den Browser und stellen Sie die Verbindung zur Sitzung erneut her.

- Die Entwicklerkonsole wiederholt den Fehler `Error: UnintentionalError at StageSocket.onClose`.

  Problemumgehung: Pro Teilnehmertoken kann nur eine Stage erstellt werden. Dieser Fehler tritt auf, wenn mehr als eine `Stage`-Instance mit demselben Teilnehmertoken erstellt wird, unabhängig davon, ob sich die Instance auf einem oder mehreren Geräten befindet.

- Es kann zu Problemen bei der Aufrechterhaltung eines `StageParticipantPublishState.PUBLISHED`-Status kommen und Sie können wiederholte `StageParticipantPublishState.ATTEMPTING_PUBLISH`-Status erhalten, wenn Sie das Ereignis `StageEvents.STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED` abhören.

  Umgehung: Beschränken Sie die Videoauflösung auf 720p, wenn Sie `getUserMedia` oder `getDisplayMedia` aufrufen. Insbesondere dürfen Ihre `getUserMedia`- und `getDisplayMedia`-Beschränkungswerte für Breite und Höhe 921 600 (1280*720) nicht überschreiten, wenn sie miteinander multipliziert werden.

- Wenn `stage.leave()` aufgerufen wird oder ein Remote-Teilnehmer die Stage verlässt, wird in der Debug-Konsole des Browsers der Fehler „404 DELETE“ angezeigt.

  Problemumgehung: Keine. Hierbei handelt es sich um einen harmlosen Fehler.

Einschränkungen von Safari

- Wenn bei einer entsprechenden Aufforderung die Erteilung einer Berechtigung verweigert wird, muss die Berechtigung in den Einstellungen auf der Safari-Website auf Betriebssystemebene zurückgesetzt werden.
- Safari erkennt nicht alle Geräte nativ so effektiv wie Firefox oder Chrome. OBS Virtual Camera wird beispielsweise nicht erkannt.

Einschränkungen von Firefox

- Damit Firefox den Bildschirm freigeben kann, müssen Systemberechtigungen aktiviert sein. Nach deren Aktivierung muss Firefox neu gestartet werden, damit es ordnungsgemäß funktioniert. Andernfalls löst der Browser eine NotFoundError-Ausnahme aus, wenn Berechtigungen als gesperrt betrachtet werden.
- Die Methode `getCapabilities` fehlt. Das bedeutet, dass Benutzer die Auflösung oder das Seitenverhältnis der Medienspur nicht abrufen können. Weitere Informationen finden Sie in diesem Bugzilla-Thread.
- Es fehlen mehrere `AudioContext`-Eigenschaften, z. B. die Latenz und die Kanalanzahl. Dies könnte für erfahrene Benutzer, die die Audiospuren bearbeiten möchten, ein Problem darstellen.

- Kamera-Feeds von `getUserMedia` sind unter macOS auf ein Seitenverhältnis von 4:3 beschränkt. Weitere Informationen finden Sie im Bugzilla-Thread 1 und im Bugzilla-Thread 2.

- Die Audioerfassung wird mit `getDisplayMedia` nicht unterstützt. Weitere Informationen finden Sie in diesem Bugzilla-Thread.
- Die Framerate bei der Bildschirmerfassung ist suboptimal (ungefähr 15 Bilder pro Sekunde?). Weitere Informationen finden Sie in diesem Bugzilla-Thread.

Einschränkungen im mobilen Web

- Die Bildschirmfreigabe von getDisplayMedia wird auf Mobilgeräten nicht unterstützt.

  Problemumgehung: Keine.
- Beim Schließen eines Browsers dauert es 15 bis 30 Sekunden, bis der Teilnehmer den Browser verlässt, ohne `leave()` aufzurufen.

  Problemumgehung: Fügen Sie eine Benutzeroberfläche hinzu, die Benutzer dazu ermutigt, die Verbindung ordnungsgemäß zu trennen.
- Die Hintergrund-App führt dazu, dass die Veröffentlichung von Videos beendet wird.

  Problemumgehung: Zeigen Sie ein UI-Slate an, wenn der Publisher angehalten ist.
- Nach dem Aufheben der Stummschaltung einer Kamera auf Android-Geräten sinkt die Video-Framerate für etwa 5 Sekunden.

  Problemumgehung: Keine.
- Der Video-Feed wird bei der Rotation für iOS 16.0 gestreckt.

  Problemumgehung: Zeigen Sie eine Benutzeroberfläche an, die dieses bekannte Betriebssystemproblem beschreibt.
- Beim Wechseln des Audio-Eingabegeräts wird automatisch auch das Audio-Ausgabegerät umgeschaltet.

  Problemumgehung: Keine.
- Wenn der Browser in den Hintergrund gestellt wird, wird der Veröffentlichungsstream schwarz und es wird nur Audio erzeugt.

  Problemumgehung: Keine. Dies geschieht aus Sicherheitsgründen.

Fehlerbehandlung im IVS Web Broadcast SDK | Streaming in Echtzeit

Dieser Abschnitt gibt einen Überblick über die Fehlerbedingungen, wie das Web-Broadcast-SDK sie an die Anwendung meldet und wie eine Anwendung reagieren sollte, wenn diese Fehler auftreten. Fehler werden vom SDK an die Listener des `StageEvents.ERROR`-Ereignisses gemeldet:

```
stage.on(StageEvents.ERROR, (error: StageError) => {
    // log or handle errors here
```

```
    console.log(`${error.code}, ${error.category}, ${error.message}`);
});
```

Stagefehler

Ein StageError-Fehler wird gemeldet, wenn das SDK auf ein Problem stößt, das nicht behoben werden kann und für dessen Behebung im Allgemeinen ein Eingreifen der App und/oder eine erneute Netzwerkverbindung erforderlich ist.

Jeder gemeldete `StageError` hat einen Code (oder `StageErrorCode`), eine Meldung (Zeichenfolge) und eine Kategorie (`StageErrorCategory`). Alles davon bezieht sich auf eine zugrunde liegende Vorgangskategorie.

Die Vorgangskategorie des Fehlers wird danach bestimmt, ob er mit der Verbindung zur Stage (`JOIN_ERROR`), dem Senden von Medien an die Stage (`PUBLISH_ERROR`) oder dem Empfangen eines eingehenden Medienstreams von der Stage (`SUBSCRIBE_ERROR`) zusammenhängt.

Die Codeeigenschaft eines `StageError` meldet das spezifische Problem:

Name	Code	Empfohlene Aktion
TOKEN_MALFORMED	1	Erstellen Sie ein gültiges Token und versuchen Sie erneut, die Stage zu instanziieren.
TOKEN_EXPIRED	2	Erstellen Sie ein noch nicht abgelaufenes Token und versuchen Sie erneut, die Stage zu instanziieren.
TIMEOUT	3	Bei der Operation ist eine Zeitüberschreitung aufgetreten. Wenn die Stage vorhanden und das Token gültig ist, handelt es sich bei diesem Fehler wahrscheinlich um ein Netzwerkproblem. Warten Sie in diesem Fall, bis die Konnektivität des Geräts wiederhergestellt ist.
FEHLGESCHLAGEN	4	Beim Versuch, einen Vorgang auszuführen, ist ein schwerwiegender Fehler aufgetreten. Überprüfen Sie Fehlerdetails. Wenn die Stage vorhanden und das Token gültig ist, handelt es sich bei diesem Fehler wahrscheinlich um

Name	Code	Empfohlene Aktion
		ein Netzwerkproblem. Warten Sie in diesem Fall, bis die Konnektivität des Geräts wiederhergestellt ist. Bei den meisten Fehlern im Zusammenhang mit der Netzwerkstabilität versucht das SDK intern für einen Zeitraum von bis zu 30 Sekunden erneut, eine Verbindung herzustellen, bevor ein FAILED-Fehler ausgegeben wird.
CANCELED	5	Überprüfen Sie den Anwendungscode und stellen Sie sicher, dass es keine wiederholten `join`-, `refreshStrategy` - oder `replaceStrategy` -Aufrufe gibt, die dazu führen könnten, dass wiederholte Vorgänge gestartet und vor Abschluss abgebrochen werden.
STAGE_AT_CAPACITY	6	Dieser Fehler weist darauf hin, dass die Stage oder Ihr Konto voll ausgelastet ist. Falls die Stage das Teilnehmerlimit erreicht hat, wiederholen Sie den Vorgang, wenn die Stage nicht mehr voll ausgelastet ist, indem Sie die Strategie aktualisieren. Wenn Ihr Konto das Kontingent für gleichzeitige Abonnements oder Publisher erreicht hat, reduzieren Sie die Nutzung oder fordern Sie über die AWS-Service-Quotas-Konsole eine Erhöhung des Kontingents an.
CODEC_MISMATCH	7	Der Codec wird von der Stage nicht unterstützt. Überprüfen Sie den Browser und die Plattform auf Codec-Unterstützung. Für IVS-Echtzeit-Streaming müssen Browser den H.264-Codec für Video und den Opus-Codec für Audio unterstützen.
TOKEN_NOT_ALLOWED	8	Das Token hat keine Berechtigung für den Vorgang. Erstellen Sie das Token mit den richtigen Berechtigungen neu und versuchen Sie es erneut.

Name	Code	Empfohlene Aktion
STAGE_DELETED	9	Keine; der Versuch, einer gelöschten Stage beizutreten, löst diesen Fehler aus.
PARTICIPANT_DISCONNECTED	10	Keine; der Versuch, mit dem Token eines Teilnehmers beizutreten, dessen Verbindung getrennt wurde, löst diesen Fehler aus.

Beispiel für die Behandlung von StageError

Verwenden Sie den StageError-Code, um festzustellen, ob der Fehler auf ein abgelaufenes Token zurückzuführen ist:

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

Netzwerkfehler, wenn bereits eine Verbindung hergestellt wurde

Wenn die Netzwerkverbindung des Geräts ausfällt, verliert das SDK möglicherweise die Verbindung zu den Stageservern. Möglicherweise werden in der Konsole Fehler angezeigt, da das SDK die Backend-Dienste nicht mehr erreichen kann. POSTs auf https://broadcast.stats.live-video.net schlagen fehl.

Wenn Sie etwas veröffentlichen und/oder abonnieren, werden in der Konsole Fehler angezeigt, die sich auf Versuche beziehen, etwas zu veröffentlichen/zu abonnieren.

Intern versucht das SDK, die Verbindung mithilfe einer exponentiellen Backoff-Strategie wiederherzustellen.

Aktion: Warten Sie, bis die Konnektivität des Geräts wiederhergestellt ist.

Fehlerstatus

Es wird empfohlen, diese Status für die Anwendungsprotokollierung zu verwenden und Benutzern Nachrichten anzuzeigen, die sie über Verbindungsprobleme mit der Stage für einen bestimmten Teilnehmer informieren.

Veröffentlichen

Das SDK meldet `ERRORED`, wenn eine Veröffentlichung fehlschlägt.

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

Abonnieren

Das SDK meldet `ERRORED`, wenn ein Abonnement fehlschlägt. Dies kann auf Netzwerkbedingungen zurückzuführen sein oder wenn eine Stage für Abonnenten ausgelastet ist.

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

IVS-Broadcast-SDK: Android-Handbuch | Echtzeit-Streaming

Das Android-Broadcast-SDK für IVS-Echtzeit-Streaming ermöglicht es den Teilnehmern, Videos auf Android zu senden und zu empfangen.

Das Paket `com.amazonaws.ivs.broadcast` implementiert die in diesem Dokument beschriebene Schnittstelle. Das SDK unterstützt die folgenden Vorgänge:

- Einer Stage beitreten
- Medien für andere Teilnehmer auf der Stage veröffentlichen
- Medien anderer Teilnehmer auf der Stage abonnieren
- Auf der Stage veröffentlichte Videos und Audios verwalten und überwachen
- WebRTC-Statistiken für jede Peer-Verbindung beziehen
- Alle Vorgänge aus dem IVS-Streaming-SDK für Android-Übertragungen mit niedriger Latenz

Aktuelle Version des Broadcast-SDK für Android: 1.40.0 (Versionshinweise)

Informationen zu den wichtigsten Methoden, die im Amazon-IVS-Android-Broadcast-SDK verfügbar sind, finden Sie in der Referenzdokumentation unter https://aws.github.io/amazon-ivs-broadcast-docs/1.40.0/android/.

Beispiel-Code: Siehe das Android-Beispiel-Repository auf GitHub: https://github.com/aws-samples/amazon-ivs-real-time-streaming-android-samples.

Plattformanforderungen: Android 9.0 und höher

Erste Schritte mit dem IVS Android Broadcast SDK | Streaming in Echtzeit

Dieses Dokument führt Sie durch die Schritte zum Einstieg in das Android Broadcast SDK von IVS-Streaming in Echtzeit.

Bibliothek installieren

Es gibt mehrere Möglichkeiten, die Android-Broadcast-Bibliothek von Amazon IVS Ihrer Android-Entwicklungsumgebung hinzuzufügen: direkte Verwendung von Gradle, Verwendung von Gradle-Versionskatalogen oder manuelle Installation des SDK.

Direkte Verwendung von Gradle: Fügen Sie die Bibliothek zur `build.gradle`-Datei Ihres Moduls hinzu, wie hier gezeigt (für die aktuelle Version des IVS-Broadcast-SDK):

```
repositories {
    mavenCentral()
}

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

Verwendung von Gradle-Versionskatalogen: Fügen Sie zunächst Folgendes in die `build.gradle`-Datei Ihres Moduls ein:

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

Fügen Sie anschließend Folgendes in die `libs.version.toml`-Datei ein (für die aktuelle Version des IVS-Broadcast-SDK):

```
[versions]
ivs="1.40.0"

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

Manuelle Installation des SDK: Laden Sie die aktuelle Version von diesem Speicherort herunter:

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

Laden Sie unbedingt die `aar` mit `-stages` angehängt herunter.

Außerdem die Kontrolle der Freisprecheinrichtung durch das SDK zulassen: Unabhängig davon, welche Installationsmethode Sie wählen, fügen Sie Ihrem Manifest auch die folgende Berechtigung hinzu, damit das SDK die Freisprecheinrichtung aktivieren und deaktivieren kann:

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

Verwenden des SDK mit Debug-Symbolen

Wir veröffentlichen auch eine Version des Android-Broadcast-SDK, die Debug-Symbole enthält. Sie können diese Version verwenden, um die Qualität von Debug-Berichten (Stack-Traces) in Firebase Crashlytics zu verbessern, falls im IVS-Broadcast-SDK Abstürze auftreten, d. h. `libbroadcastcore.so`. Wenn Sie diese Abstürze dem SDK-Team von IVS melden, erleichtern die qualitativ hochwertigeren Stack-Traces die Behebung der Probleme.

Um diese Version des SDK zu verwenden, fügen Sie Folgendes in Ihre Gradle-Build-Dateien ein:

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

Verwenden Sie die obige Zeile anstelle von:

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

Hochladen von Symbolen zu Firebase Crashlytics

Stellen Sie sicher, dass Ihre Gradle-Build-Dateien für Firebase Crashlytics eingerichtet sind. Folgen Sie den Anweisungen von Google hier:

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

Achten Sie darauf, `com.google.firebase:firebase-crashlytics-ndk` als Abhängigkeit anzugeben.

Wenn Sie Ihre App für die Veröffentlichung erstellen, muss das Firebase-Crashlytics-Plugin Symbole automatisch hochladen. Führen Sie einen der folgenden Befehle aus, um Symbole manuell hochzuladen:

```
gradle uploadCrashlyticsSymbolFileRelease
```

```
./gradlew uploadCrashlyticsSymbolFileRelease
```

(Es schadet nicht, wenn Symbole zweimal hochgeladen werden, sowohl automatisch als auch manuell.)

Verhindern, dass Ihre APK-Version größer wird

Vor dem Verpacken der `.apk`-Release-Datei versucht das Android-Gradle-Plugin automatisch, Debug-Informationen aus gemeinsam genutzten Bibliotheken (einschließlich der `libbroadcastcore.so`-Bibliothek des IVS-Broadcast-SDK) zu entfernen. Manchmal geschieht dies jedoch nicht. Infolgedessen könnte Ihre `.apk`-Datei größer werden und Sie könnten vom Android-Gradle-Plugin eine Warnmeldung erhalten, dass es Debug-Symbole nicht entfernen kann und die `.so`-Dateien unverändert verpackt. Wenn dies passiert, gehen Sie wie folgt vor:

- Installieren Sie ein Android-NDK. Jede aktuelle Version funktioniert.
- Fügen Sie `ndkVersion <your_installed_ndk_version_number>` zur `build.gradle`-Datei Ihrer Anwendung hinzu. Tun Sie dies auch dann, wenn Ihre Anwendung selbst keinen nativen Code enthält.

Weitere Informationen finden Sie in diesem Problembericht.

Berechtigungen anfordern

Ihre App muss die Berechtigung für den Zugriff auf die Kamera und das Mikrofon des Benutzers anfordern. (Dies ist nicht spezifisch für Amazon IVS; es ist für alle Anwendungen erforderlich, die Zugriff auf Kameras und Mikrofone benötigen.)

Hier prüfen wir, ob der Benutzer bereits Berechtigungen erteilt hat und fragen, wenn nicht, nach ihnen:

```
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;
    }
}
```

Hier erhalten wir die Antwort des Benutzers:

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

Veröffentlichen und Abonnieren mit dem IVS Android Broadcast SDK | Streaming in Echtzeit

Dieses Dokument führt Sie durch die Schritte zum Veröffentlichen und Abonnieren einer Stufe mit dem Android Broadcast SDK von IVS-Streaming in Echtzeit.

Konzepte

Drei Kernkonzepte liegen der Echtzeit-Funktionalität zugrunde: Stage, Strategie und Renderer. Das Designziel besteht in der Minimierung der Menge an clientseitiger Logik, die für die Entwicklung eines funktionierenden Produkts erforderlich ist.

Stage

Die Klasse `Stage` ist der Hauptinteraktionspunkt zwischen der Hostanwendung und dem SDK. Sie stellt die Stage selbst dar und dient dazu, der Stage beizutreten und sie zu verlassen. Für das Erstellen einer Bühne und das Beitreten ist eine gültige, noch nicht abgelaufene Token-Zeichenfolge aus der Steuerebene erforderlich (dargestellt als `token`). Einer Bühne beizutreten und sie zu verlassen, ist ganz einfach.

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

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

stage.leave();
```

In der Klasse `Stage` erfolgt auch das Anhängen des `StageRenderer`:

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

Strategie

Über die Schnittstelle `Stage.Strategy` kann die Hostanwendung dem SDK den gewünschten Status der Bühne mitteilen. Drei Funktionen müssen implementiert werden: `shouldSubscribeToParticipant`, `shouldPublishFromParticipant` und `stageStreamsToPublishForParticipant`. Alle werden im Folgenden behandelt.

Abonnieren von Teilnehmern

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

Wenn ein Remote-Teilnehmer der Stage beitritt, fragt das SDK die Hostanwendung nach dessen gewünschtem Abonnementstatus. Die Optionen lauten `NONE`, `AUDIO_ONLY` und `AUDIO_VIDEO`. Wenn ein Wert für diese Funktion zurückgegeben wird, muss sich die Hostanwendung nicht um den Veröffentlichungs-, den aktuellen Abonnement- oder den Verbindungsstatus des Stage kümmern. Bei Rückgabe von `AUDIO_VIDEO` wartet das SDK mit dem Abonnieren, bis der Remote-Teilnehmer etwas veröffentlicht. Außerdem aktualisiert das SDK die Hostanwendung während des gesamten Prozesses über den Renderer.

Hier folgt ein Beispiel für eine Implementierung:

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

Hierbei handelt es sich um die vollständige Implementierung dieser Funktion für eine Hostanwendung, bei der sich alle Teilnehmer stets gegenseitig sehen sollen; z. B. eine Video-Chat-Anwendung.

Weitergehende Implementierungen sind ebenfalls möglich. Nutzen Sie die Eigenschaft `userInfo` für `ParticipantInfo`, um Teilnehmer anhand der vom Server bereitgestellten Attribute selektiv zu abonnieren:

```
@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;
 }
}
```

Hiermit kann eine Bühne erstellt werden, auf der Moderatoren alle Gäste überwachen können, ohne selbst gesehen oder gehört zu werden. Die Hostanwendung könnte eine zusätzliche Geschäftslogik nutzen, damit Moderatoren sich gegenseitig sehen können, für Gäste aber unsichtbar bleiben.

Konfiguration für das Abonnieren von Teilnehmern

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

Wenn ein Remote-Teilnehmer abonniert wird (siehe Teilnehmer abonnieren), fragt das SDK die Host-Anwendung nach einer benutzerdefinierten Abonnementkonfiguration für diesen Teilnehmer ab. Diese Konfiguration ist optional und ermöglicht es der Hostanwendung, bestimmte Aspekte des Subscriber-Verhaltens zu steuern. Informationen darüber, was konfiguriert werden kann, finden Sie unter SubscribeConfiguration in der SDK-Referenzdokumentation.

Hier folgt ein Beispiel für eine Implementierung:

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


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

    return config;
}
```

Diese Implementierung aktualisiert die Mindestverzögerung für den Jitter-Buffer für alle abonnierten Teilnehmer auf die Voreinstellung `MEDIUM`.

Wie bei `shouldSubscribeToParticipant` sind auch hier weitergehende Implementierungen möglich. Die `ParticipantInfo`-Angaben können verwendet werden, um die Abonnementkonfiguration für bestimmte Teilnehmer selektiv zu aktualisieren.

Wir empfehlen die Verwendung der Standardverhaltensweisen. Geben Sie die benutzerdefinierte Konfiguration nur an, wenn Sie ein bestimmtes Verhalten ändern möchten.

Veröffentlichen

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

Sobald die Verbindung zur Stage hergestellt ist, überprüft das SDK per Anfrage an die Hostanwendung, ob ein bestimmter Teilnehmer etwas veröffentlichen soll. Dies wird nur bei lokalen

Teilnehmern aufgerufen, die auf Grundlage des bereitgestellten Tokens zur Veröffentlichung berechtigt sind.

Hier folgt ein Beispiel für eine Implementierung:

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

Sie ist für eine normale Video-Chat-Anwendung gedacht, bei der Benutzer immer etwas veröffentlichen möchten. Sie können die Audio- und Videowiedergabe stummschalten und die Stummschaltung aufheben, um umgehend ausgeblendet oder gesehen/gehört zu werden. (Sie können auch „Veröffentlichen/Veröffentlichung aufheben“ verwenden, was aber viel langsamer ist. „Stummschalten/Stummschalten aufheben“ ist für Anwendungsfälle vorzuziehen, in denen eine häufige Änderung der Sichtbarkeit wünschenswert ist.)

Auswählen von Streams zur Veröffentlichung

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

Beim Veröffentlichen wird hiermit bestimmt, welche Audio- und Videostreams veröffentlicht werden sollen. Dieser Punkt wird später unter Veröffentlichen eines Medienstreams ausführlicher behandelt.

Aktualisieren der Strategie

Die Strategie soll dynamisch sein: Die von einer der oben genannten Funktionen zurückgegebenen Werte lassen sich jederzeit ändern. Wenn die Hostanwendung beispielsweise erst veröffentlichen soll, wenn der Endbenutzer auf eine Schaltfläche tippt, können Sie eine Variable aus `shouldPublishFromParticipant` zurückgeben (zum Beispiel `hasUserTappedPublishButton`). Wenn sich diese Variable aufgrund einer Interaktion des Endbenutzers ändert, signalisieren Sie dem SDK per Aufruf von `stage.refreshStrategy()`, dass es die Strategie nach den neuesten Werten abfragen und nur Dinge anwenden soll, die sich geändert haben. Wenn das SDK feststellt, dass sich der Wert `shouldPublishFromParticipant` geändert hat, startet es den Veröffentlichungsprozess. Wenn alle Funktionen bei einer SDK-Abfrage

den gleichen Wert zurückgeben wie zuvor, werden mit dem Aufruf von `refreshStrategy` keine Änderungen an der Bühne durchgeführt.

Ändert sich der Rückgabewert von `shouldSubscribeToParticipant` von `AUDIO_VIDEO` in `AUDIO_ONLY`, wird der Videostream für alle Teilnehmer mit geänderten Rückgabewerten entfernt, sofern zuvor ein Videostream vorhanden war.

Im Allgemeinen nutzt die Stage die Strategie, um den Unterschied zwischen der vorherigen und der aktuellen Strategie am effizientesten anzuwenden. Dabei muss sich die Hostanwendung nicht um die ganzen Status kümmern, die für eine ordnungsgemäße Verwaltung erforderlich sind. Stellen Sie sich den Aufruf von `stage.refreshStrategy()` daher als einen ressourcenschonenden Vorgang vor, da nur bei einer Änderung der Strategie etwas unternommen wird.

Renderer

Die Schnittstelle `StageRenderer` teilt der Hostanwendung den Status der Bühne mit. Aktualisierungen in der Benutzeroberfläche der Hostanwendung können in der Regel vollständig über die vom Renderer bereitgestellten Ereignisse gesteuert werden. Der Renderer stellt die folgenden Funktionen bereit:

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

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

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

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

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

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

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

void onError(@NonNull BroadcastException exception);
```

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

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

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

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

Für die meisten dieser Methoden werden die entsprechende `Stage` und `ParticipantInfo` bereitgestellt.

Es wird nicht erwartet, dass sich die vom Renderer bereitgestellten Informationen auf die Rückgabewerte der Strategie auswirken. Es wird beispielsweise nicht erwartet, dass sich der Rückgabewert von `shouldSubscribeToParticipant` beim Aufruf von `onParticipantPublishStateChanged` ändert. Wenn die Hostanwendung einen bestimmten Teilnehmer abonnieren möchte, muss sie unabhängig von dessen Veröffentlichungsstatus den gewünschten Abonnementtyp zurückgeben. Das SDK muss dafür sorgen, dass entsprechend dem Status der Bühne und dem gewünschten Status der Strategie zum richtigen Zeitpunkt gehandelt wird.

Der `StageRenderer` kann der Bühnenklasse angefügt werden:

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

Hinweis: Nur veröffentlichende Teilnehmer lösen `onParticipantJoined` aus. Wenn Teilnehmer die Veröffentlichung beenden oder die Bühnensitzung verlassen, wird `onParticipantLeft` ausgelöst.

Veröffentlichen eines Medienstreams

Lokale Geräte wie eingebaute Mikrofone und Kameras werden über `DeviceDiscovery` erkannt. Hier folgt ein Beispiel für die Auswahl der nach vorne gerichteten Kamera und des Standardmikrofons und deren anschließende Rückgabe als `LocalStageStreams` zur Veröffentlichung durch das 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) {
  front Camera = 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;
}
```

Anzeigen und Entfernen von Teilnehmern

Nach Abschluss von Abonnements erhalten Sie über die Funktion on`StreamsAdded` des Renderers eine Reihe von `StageStream`-Objekten. Sie können die Vorschau von einem `ImageStageStream` abrufen:

```
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);
```

Außerdem können Sie die Statistiken des Audiolevels von einem `AudioStageStream` abrufen:

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

Wenn ein Teilnehmer die Veröffentlichung beendet oder dessen Abonnement beendet wird, wird die Funktion `onStreamsRemoved` mit den Streams aufgerufen, die entfernt wurden. Hostanwendungen sollten dies als Signal nutzen, um den Videostream des Teilnehmers aus der Ansichtshierarchie zu entfernen.

`onStreamsRemoved` wird für alle Szenarien aufgerufen, in denen ein Stream entfernt werden könnte, darunter:

- Der Remote-Teilnehmer beendet die Veröffentlichung.
- Ein lokales Gerät beendet das Abonnement oder ändert das Abonnement von `AUDIO_VIDEO` in `AUDIO_ONLY`.
- Der Remote-Teilnehmer verlässt die Stage.
- Der lokale Teilnehmer verlässt die Stage.

Da `onStreamsRemoved` bei allen Szenarien aufgerufen wird, ist keine benutzerdefinierte Geschäftslogik erforderlich, um Teilnehmer beim remoten oder lokalen Verlassen aus der Benutzeroberfläche zu entfernen.

Stummschalten von Medienstreams und Aufheben der Stummschaltung

`LocalStageStream`-Objekte verfügen über eine `setMuted`-Funktion, die das Stummschalten des Streams steuert. Diese Funktion kann für den Stream aufgerufen werden, bevor oder nachdem er von der Strategiefunktion `streamsToPublishForParticipant` zurückgegeben wird.

Wichtig: Wenn nach einem Aufruf von `refreshStrategy` eine neue `LocalStageStream`-Objekt-Instance von `streamsToPublishForParticipant` zurückgegeben wird, wird der Stummschaltungsstatus des neuen Streamobjekts auf die Bühne angewendet. Seien Sie vorsichtig

beim Erstellen neuer `LocalStageStream`-Instances, um sicherzustellen, dass der erwartete Stummschaltungsstatus beibehalten wird.

Überwachen des Medien-Stummschaltungsstatus von Remote-Teilnehmern

Wenn ein Teilnehmer den Stummschaltungsstatus seines Video- oder Audiostreams ändert, wird die Funktion on`StreamMutedChanged` des Renderers mit einer Liste der Streams aufgerufen, die sich geändert haben. Verwenden Sie die Methode `getMuted` für `StageStream`, um die Benutzeroberfläche entsprechend zu aktualisieren.

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

Abrufen von WebRTC-Statistiken

Um die neuesten WebRTC-Statistiken für einen veröffentlichten oder abonnierten Stream abzurufen, verwenden Sie `requestRTCStats` für `StageStream`. Nach Abschluss einer Erfassung erhalten Sie Statistiken über den `StageStream.Listener`, der für `StageStream` eingestellt werden kann.

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

Abrufen von Teilnehmerattributen

Wenn Sie Attribute in der Vorgangsanfrage `CreateParticipantToken` angeben, können Sie die Attribute in den Eigenschaften von `ParticipantInfo` einsehen:

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

Eingebettete Nachrichten

Die `embedMessage`-Methode auf ImageDevice ermöglicht es Ihnen, Metadaten-Nutzdaten während der Veröffentlichung direkt in Videoframes einzufügen. Dies ermöglicht framesynchronisiertes Messaging für Echtzeitanwendungen. Das Einbetten von Nachrichten ist nur verfügbar, wenn das SDK für die Veröffentlichung in Echtzeit verwendet wird (nicht für Veröffentlichungen mit niedriger Latenz).

Es kann nicht garantiert werden, dass eingebettete Nachrichten bei Abonnenten ankommen, da sie direkt in Videoframes eingebettet und über UDP übertragen werden, wodurch die Paketzustellung nicht garantiert wird. Der Verlust von Paketen während der Übertragung kann zum Verlust von Nachrichten führen, insbesondere bei schlechten Netzwerkbedingungen. Um dem entgegenzuwirken, beinhaltet die `embedMessage`-Methode einen `repeatCount`-Parameter, der die Nachricht über mehrere aufeinanderfolgende Frames dupliziert und so die Zuverlässigkeit der Zustellung erhöht. Diese Funktion ist nur für Videostreams verfügbar.

Verwenden von embedMessage

Publishing-Clients können Nachrichten-Nutzdaten mithilfe der `embedMessage`-Methode auf ImageDevice in ihren Videostream einbetten. Die Nutzdaten müssen größer als 0 KB und kleiner als 1 KB sein. Die Anzahl der pro Sekunde eingebetteten Nachrichten darf 10 KB pro Sekunde nicht überschreiten.

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

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

Nachrichten-Nutzdaten

Verwenden Sie `repeatCount`, um die Nachricht über mehrere Frames hinweg zu duplizieren, um die Zuverlässigkeit zu erhöhen. Dieser Wert muss zwischen 0 und 30 liegen. Empfangende Clients müssen über eine Logik verfügen, um die Nachricht zu deduplizieren.

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

Lesen eingebetteter Nachrichten

Informationen zum Lesen eingebetteter Nachrichten aus eingehenden Streams finden Sie weiter unten unter „Zusätzliche Erweiterungsinformationen (SEI) abrufen“.

Abrufen von SEI-Daten (Supplemental Enhancement Information)

Die NAL-Einheit für Supplemental Enhancement Information (SEI) wird verwendet, um Frame-orientierte Metadaten zusammen mit dem Video zu speichern. Subscriber können SEI-Nutzlasten von einem Publisher lesen, der H.264-Video veröffentlicht, indem sie die Eigenschaften `embeddedMessages` der `ImageDeviceFrame`-Objekte aus dem `ImageDevice` des Publishers überprüfen. Erlangen Sie dazu das `ImageDevice` eines Publishers und beobachten Sie dann jeden Frame über einen Callback an `setOnFrameCallback`, wie im folgenden Beispiel gezeigt:

```
// in a StageRenderer’s onStreamsAdded function, after acquiring the new ImageStream

val imageDevice = imageStream.device as ImageDevice
imageDevice.setOnFrameCallback(object : ImageDevice.FrameCallback {
 override fun onFrame(frame: ImageDeviceFrame) {
      for (message in frame.embeddedMessages) {
          if (message is UserDataUnregisteredSeiMessage) {
              val seiMessageBytes = message.data
              val seiMessageUUID = message.uuid

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

```
})
```

Fortsetzen der Sitzung im Hintergrund

Wenn die App in den Hintergrund wechselt, können Sie die Veröffentlichung beenden oder das Abonnement auf das Audio anderer Remote-Teilnehmer beschränken. Dazu aktualisieren Sie die Implementierung Ihrer `Strategy`, um die Veröffentlichung zu beenden und `AUDIO_ONLY` zu abonnieren (oder gegebenenfalls `NONE`).

```
// 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;
 subscribeTpye = Stage.SubscribeType.AUDIO_ONLY;
 stage.refreshStrategy();
}
```

Mehrschichtige Kodierung mit Simulcast

Bei der mehrschichtigen Kodierung mit Simulcast handelt es sich um ein Feature für IVS-Echtzeit-Streaming, mit dessen Hilfe Publisher mehrere Videoschichten unterschiedlicher Qualität senden können. Subscriber können diese Schichten dynamisch oder manuell konfigurieren. Das Feature wird im Dokument Streaming-Optimierungen ausführlicher beschrieben.

Konfigurieren mehrschichtiger Kodierung (Publisher)

Um als Publisher die mehrschichtige Kodierung mit Simulcast zu aktivieren, fügen Sie dem `LocalStageStream` bei der Instanziierung die folgende Konfiguration hinzu:

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

ImageLocalStageStream cameraStream = new ImageLocalStageStream(frontCamera, config);

// Other Stage implementation code
```

Je nach der in der Videokonfiguration eingestellten Auflösung wird eine festgelegte Anzahl von Schichten kodiert und gesendet, wie im Abschnitt Standardmäßige Schichten, Qualitäten und Bildraten von Streaming-Optimierungen definiert.

Außerdem können Sie optional einzelne Ebenen innerhalb der Simulcast-Konfiguration konfigurieren:

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

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

config.simulcast.setLayers(simulcastLayers);

ImageLocalStageStream cameraStream = new ImageLocalStageStream(frontCamera, config);

// Other Stage implementation code
```

Alternativ können Sie eigene benutzerdefinierte Ebenenkonfigurationen für bis zu drei Ebenen erstellen. Wenn Sie ein leeres Array oder keinen Wert angeben, werden die oben beschriebenen Standardwerte verwendet. Ebenen werden mit den folgenden erforderlichen Eigenschaftssetzern beschrieben:

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

- `setMinBitrate: integer;`
- `setTargetFramerate: integer;`

Ausgehend von den Voreinstellungen können Sie entweder einzelne Eigenschaften überschreiben oder eine völlig neue Konfiguration erstellen:

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

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

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

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

config.simulcast.setLayers(simulcastLayers);

ImageLocalStageStream cameraStream = new ImageLocalStageStream(frontCamera, config);

// Other Stage implementation code
```

Informationen zu Höchstwerten, Grenzwerten und Fehlern, die bei der Konfiguration einzelner Ebenen ausgelöst werden können, finden Sie in der SDK-Referenzdokumentation.

Konfigurieren mehrschichtiger Kodierung (Subscriber)

Subscriber müssen nichts unternehmen, um die mehrschichtige Kodierung zu aktivieren. Wenn ein Publisher Simulcast-Schichten sendet, passt sich der Server standardmäßig dynamisch den Schichten an, um je nach Gerät und Netzwerkbedingungen des Subscribers die optimale Qualität auszuwählen.

Alternativ gibt es mehrere nachfolgend beschriebene Optionen, um explizite Schichten auszuwählen, die der Publisher sendet.

Option 1: Einstellung für die Qualität der Anfangsschicht

Mit der Strategie `subscribeConfigurationForParticipant` können Sie auswählen, welche Anfangsschicht Sie als Subscriber erhalten möchten:

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


 config.simulcast.setInitialLayerPreference(SubscribeSimulcastConfiguration.InitialLayerPrefere

    return config;
}
```

Standardmäßig wird Subscribern zunächst immer die Schicht mit der niedrigsten Qualität gesendet. Nach und nach wird die Qualität gesteigert, bis die Schicht mit der höchsten Qualität erreicht ist. Das optimiert den Bandbreitenverbrauch der Endbenutzer, verkürzt die Zeit bis zum Abspielen des Videos und verringert das anfängliche Einfrieren von Videos bei Benutzern in Netzwerken mit geringerer Bandbreite.

Folgende Optionen sind für `InitialLayerPreference` verfügbar:

- `LOWEST_QUALITY` – Der Server stellt zuerst die Videoschicht mit der niedrigsten Qualität bereit. Dadurch werden der Bandbreitenverbrauch und die Zeit bis zum Abspielen von Medien optimiert. Die Qualität ist definiert als die Kombination aus Größe, Bitrate und Bildrate des Videos. Beispielsweise weisen 720p-Videos eine geringere Qualität auf als 1080p-Videos.
- `HIGHEST_QUALITY` – Der Server stellt zuerst die Videoschicht mit der höchsten Qualität bereit. Das optimiert die Qualität, kann aber die Zeit bis zum Abspielen von Medien verlängern. Die Qualität ist definiert als die Kombination aus Größe, Bitrate und Bildrate des Videos. Beispielsweise weisen 1080p-Videos eine höhere Qualität auf als 720p-Videos.

Hinweis: Damit die anfänglichen Schichteinstellungen (der Aufruf `setInitialLayerPreference`) wirksam werden, muss ein neues Abonnement abgeschlossen werden, da diese Updates für das aktive Abonnement nicht gelten.

Option 2: Bevorzugte Schicht für Streams

Mit der Strategiemethode `preferredLayerForStream` können Sie eine Schicht auswählen, nachdem der Stream gestartet wurde. Diese Strategiemethode erhält die Teilnehmer- und Streaminformationen, sodass Sie eine Schicht für jeden Teilnehmer auswählen können. Das SDK ruft diese Methode als Reaktion auf bestimmte Ereignisse auf, z. B. wenn sich die Streamschichten ändern, sich der Teilnehmerstatus ändert oder die Hostanwendung die Strategie aktualisiert.

Der Strategiemodus gibt ein `RemoteStageStream.Layer`-Objekt zurück, wobei es sich um Folgendes handeln kann:

- ein Schichtobjekt, z. B. eines, das von `RemoteStageStream.getLayers` zurückgegeben wird
- null, was bedeutet, dass keine Schicht ausgewählt werden sollte und eine dynamische Anpassung bevorzugt wird

Bei der folgenden Strategie wählen die Benutzer beispielsweise immer die Videoschicht mit der niedrigsten verfügbaren Qualität aus:

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

Um die Schichtauswahl zurückzusetzen und zur dynamischen Anpassung zurückzukehren, geben Sie in der Strategie null oder undefiniert zurück. In diesem Beispiel ist `appState` eine Platzhaltervariable, die den Status der Hostanwendung darstellt.

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

Option 3: Helferobjekte für RemoteStageStream-Schicht

`RemoteStageStream` weist mehrere Helferobjekte auf, mit deren Hilfe Entscheidungen über die Schichtauswahl getroffen und Endbenutzern die entsprechende Auswahl angezeigt werden kann:

- Schichtereignisse – Neben `StageRenderer` verfügt der `RemoteStageStream.Listener` über Ereignisse, die Änderungen bei der Schicht- und Simulcast-Anpassung kommunizieren:
  - `void onAdaptionChanged(boolean adaption)`
  - `void onLayersChanged(@NonNull List<Layer> layers)`
  - `void onLayerSelected(@Nullable Layer layer, @NonNull LayerSelectedReason reason)`
- Schichtmethoden – `RemoteStageStream` verfügt über mehrere Helfermethoden, mit denen Informationen über den Stream und die präsentierten Schichten abgerufen werden können. Diese Methoden sind sowohl für den in der Strategie `preferredLayerForStream` bereitgestellten Remote-Stream als auch für Remote-Streams verfügbar, die über `StageRenderer.onStreamsAdded` verfügbar gemacht werden.
  - `stream.getLayers`
  - `stream.getSelectedLayer`
  - `stream.getLowestQualityLayer`
  - `stream.getHighestQualityLayer`
  - `stream.getLayersWithConstraints`

Einzelheiten finden Sie im Abschnitt zur Klasse `RemoteStageStream` in der SDK-Referenzdokumentation. Falls als Grund für `LayerSelected` `UNAVAILABLE` zurückgegeben wird, bedeutet das, dass die angeforderte Schicht nicht ausgewählt werden konnte. Stattdessen wird eine bestmögliche Auswahl getroffen. Dabei handelt es sich in der Regel um eine Schicht mit niedrigerer Qualität, um die Stabilität des Streams zu gewährleisten.

Einschränkungen der Videokonfiguration

Das SDK unterstützt kein Erzwingen des Hoch- oder Querformats mit `StageVideoConfiguration.setSize(BroadcastConfiguration.Vec2 size)`. Im Hochformat wird die kleinere Dimension als Breite verwendet, im Querformat als Höhe. Das bedeutet, dass die folgenden beiden Aufrufe von `setSize` die gleiche Auswirkung auf die Videokonfiguration haben:

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

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

Umgang mit Netzwerkproblemen

Bei Unterbrechung der Netzwerkverbindung des lokalen Geräts versucht das SDK intern, die Verbindung ohne Benutzeraktion wiederherzustellen. In einigen Fällen ist das SDK nicht erfolgreich, weshalb eine Benutzeraktion erforderlich ist. Es gibt zwei Hauptfehler im Zusammenhang mit der Unterbrechung der Netzwerkverbindung:

- Fehlercode 1400, Meldung: „Die PeerConnection wurde aufgrund eines unbekannten Netzwerkfehlers unterbrochen.“
- Fehlercode 1300, Meldung: „Die Zahl der Wiederholungsversuche ist ausgeschöpft.“

Wenn der erste Fehler empfangen wird, der zweite jedoch nicht, ist das SDK immer noch mit der Bühne verbunden und versucht, die Verbindungen automatisch wiederherzustellen. Zur Sicherheit können Sie `refreshStrategy` ohne Änderungen an den Rückgabewerten der Strategiemethode aufrufen, um einen manuellen Neuverbindungsversuch auszulösen.

Wenn der zweite Fehler empfangen wird, sind die Neuverbindungsversuche des SDK fehlgeschlagen und das lokale Gerät ist nicht mehr mit der Bühne verbunden. Versuchen Sie in diesem Fall, der Bühne erneut beizutreten, indem Sie `join` aufrufen, nachdem die Netzwerkverbindung wiederhergestellt wurde.

Im Allgemeinen deutet das Auftreten von Fehlern nach dem erfolgreichen Beitritt zu einer Bühne darauf hin, dass das SDK beim Wiederherstellen einer Verbindung nicht erfolgreich war. Erstellen Sie ein neues `Stage`-Objekt und versuchen Sie, der Bühne beizutreten, wenn sich die Netzwerkbedingungen verbessern.

Verwenden von Bluetooth-Mikrofonen

Um mit Bluetooth-Mikrofongeräten zu veröffentlichen, müssen Sie eine Bluetooth-SCO-Verbindung herstellen:

```
Bluetooth.startBluetoothSco(context);
// Now bluetooth microphones can be used
…
```

```
// Must also stop bluetooth SCO
Bluetooth.stopBluetoothSco(context);
```

Bekannte Probleme und Behelfslösungen im IVS Android Broadcast SDK | Streaming in Echtzeit

In diesem Dokument werden bekannte Probleme aufgeführt, die bei der Verwendung des Android Broadcast SDK von Amazon-IVS-Streaming in Echtzeit auftreten können, und es werden mögliche Problemumgehungen vorgeschlagen.

- Wenn ein Android-Gerät in den Ruhezustand wechselt und aufwacht, befindet sich die Vorschau möglicherweise in einem eingefrorenen Zustand.

  Problemumgehung: Erstellen und nutzen Sie eine neue `Stage`.
- Wenn ein Teilnehmer mit einem Token beitritt, das von einem anderen Teilnehmer verwendet wird, wird die erste Verbindung ohne einen bestimmten Fehler getrennt.

  Problemumgehung: Keine.
- Es gibt ein seltenes Problem, bei dem der Publisher etwas veröffentlicht, der Veröffentlichungsstatus, den Subscriber erhalten, jedoch `inactive` lautet.

  Problemumgehung: Versuchen Sie, die Sitzung zu verlassen und ihr wieder beizutreten. Wenn das Problem weiterhin besteht, erstellen Sie ein neues Token für den Publisher.
- Während einer Bühnensitzung kann zeitweise ein seltenes Problem mit Tonverzerrungen auftreten, in der Regel bei längeren Anrufen.

  Problemumgehung: Der Teilnehmer mit dem verzerrtem Ton kann die Sitzung entweder verlassen und erneut beitreten oder die Veröffentlichung des Audios aufheben und dann erneut veröffentlichen.
- Externe Mikrofone werden bei der Veröffentlichung auf einer Bühne nicht unterstützt.

  Problemumgehung: Verwenden Sie kein über USB angeschlossenes externes Mikrofon, um etwas auf einer Bühne zu veröffentlichen.
- Das Veröffentlichen auf einer Bühne mit der Bildschirmfreigabe über `createSystemCaptureSources` wird nicht unterstützt.

  Problemumgehung: Verwalten Sie die Systemerfassung manuell, indem Sie benutzerdefinierte Bild- und Audioeingangsquellen verwenden.

- Wenn eine `ImagePreviewView` in einem übergeordneten Element entfernt wird (`removeView()` wird z. B. im übergeordneten Element aufgerufen), wird die `ImagePreviewView` sofort freigegeben. Die `ImagePreviewView` zeigt keine Frames an, wenn sie einer anderen übergeordneten Ansicht hinzugefügt wird.

  Problemumgehung: Fordern Sie mit `getPreview` eine andere Vorschau an.

- Beim Beitritt zu einer Bühne mit einem Samsung Galaxy S22/+ mit Android 12 tritt möglicherweise ein 1401-Fehler auf. Das lokale Gerät kann der Bühne nicht beitreten oder tritt ihr bei, hat aber keinen Ton.

  Problemumgehung: Führen Sie ein Upgrade auf Android 13 durch.

- Beim Beitritt zu einer Bühne mit einem Nokia X20 unter Android 13 lässt sich die Kamera möglicherweise nicht öffnen und es wird eine Ausnahme ausgelöst.

  Problemumgehung: Keine.

- Geräte mit dem MediaTek-Helio-Chipsatz können Videos von Remote-Teilnehmern nicht richtig wiedergeben.

  Problemumgehung: Keine.

- Auf einigen Geräten wählt das Betriebssystem möglicherweise ein anderes Mikrofon als das, das im SDK ausgewählt wurde. Das liegt daran, dass das Amazon IVS Broadcast SDK nicht steuern kann, wie die Audioroute `VOICE_COMMUNICATION` definiert wird, da sie je nach Gerätehersteller unterschiedlich ist.

  Problemumgehung: Keine.

- Einige Android-Videoencoder können nicht mit einer Videogröße von weniger als 176 × 176 konfiguriert werden. Die Konfiguration einer kleineren Größe verursacht einen Fehler und verhindert das Streaming.

  Problemumgehung: Konfigurieren Sie die Videogröße nicht auf weniger als 176 × 176.

Fehlerbehandlung im IVS Android Broadcast SDK | Streaming in Echtzeit

Dieser Abschnitt gibt einen Überblick über die Fehlerbedingungen, wie das Android Broadcast SDK von IVS-Streaming in Echtzeit sie an die Anwendung meldet und wie eine Anwendung reagieren sollte, wenn diese Fehler auftreten.

Schwerwiegende und nicht schwerwiegende Fehler

Das Fehlerobjekt hat das boolesche Feld „ist fatal“ von `BroadcastException`.

Im Allgemeinen hängen schwerwiegende Fehler mit der Verbindung zum Stages-Server zusammen (entweder kann eine Verbindung nicht hergestellt werden oder sie ist verloren gegangen und kann nicht wiederhergestellt werden). Die Anwendung sollte die Stage neu erstellen und erneut beitreten, ggf. mit einem neuen Token oder wenn die Konnektivität des Geräts wiederhergestellt ist.

Fehler, die nicht schwerwiegend sind, hängen in der Regel mit dem Status „Veröffentlichen/Abonnieren“ zusammen und werden vom SDK behandelt, das den Vorgang zum Veröffentlichen/Abonnieren erneut versucht.

Sie können diese Eigenschaft überprüfen:

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

Beitrittsfehler

Fehlerhaft formatiertes Token

Dies passiert, wenn das Phasen-Token falsch formatiert ist.

Das SDK löst bei einem Aufruf von eine Java-Ausnahme mit dem Fehlercode = 1000 und fatal = true aus. `stage.join`

Aktion: Erstellen Sie ein gültiges Token und versuchen Sie erneut, Mitglied zu werden.

Abgelaufenes Token

Dies passiert, wenn das Phasen-Token abgelaufen ist.

Das SDK löst bei einem Aufruf von eine Java-Ausnahme mit dem Fehlercode = 1001 und fatal = true aus. `stage.join`

Aktion: Erstellen Sie ein neues Token und versuchen Sie erneut, Mitglied zu werden.

Ungültiges oder widerrufenes Token

Dies passiert, wenn das Stage-Token nicht falsch formatiert ist, sondern vom Stages-Server zurückgewiesen wird. Dies wird asynchron über den von der Anwendung bereitgestellten Phasen-Renderer gemeldet.

Das SDK ruft `onConnectionStateChanged` mit einer Ausnahme auf, mit dem Fehlercode = 1026 und fatal = true.

Aktion: Erstellen Sie ein gültiges Token und versuchen Sie erneut beizutreten.

Netzwerkfehler beim ersten Beitritt

Dies passiert, wenn das SDK den Stages-Server nicht kontaktieren kann, um eine Verbindung herzustellen. Dies wird asynchron über den von der Anwendung bereitgestellten Phasen-Renderer gemeldet.

Das SDK ruft `onConnectionStateChanged` mit einer Ausnahme auf, mit dem Fehlercode = 1300 und fatal = true.

Handlung: Warten Sie, bis die Konnektivität des Geräts wiederhergestellt ist, und versuchen Sie erneut, eine Verbindung herzustellen.

Netzwerkfehler, wenn bereits eine Verbindung hergestellt wurde

Wenn die Netzwerkverbindung des Geräts ausfällt, verliert das SDK möglicherweise die Verbindung zu den Stage-Servern. Dies wird asynchron über den von der Anwendung bereitgestellten Phasen-Renderer gemeldet.

Das SDK ruft `onConnectionStateChanged` mit einer Ausnahme auf, mit dem Fehlercode = 1300 und fatal = true.

Handlung: Warten Sie, bis die Konnektivität des Geräts wiederhergestellt ist, und versuchen Sie erneut, eine Verbindung herzustellen.

Fehler beim Veröffentlichen/Abonnieren

Anfänglich

Es gibt mehrere Arten von Fehlern:

- MultihostSessionOfferCreationFailPublish (1.020)
- MultihostSessionOfferCreationFailSubscribe (1.021)

- MultihostSessionNoIceCandidates (1.022)
- MultihostSessionStageAtCapacity (1.024)
- SignallingSessionCannotRead (1.201)
- SignallingSessionCannotSend (1.202)
- SignallingSessionBadResponse (1.203)

Diese werden asynchron über den von der Anwendung bereitgestellten Stage-Renderer gemeldet.

Das SDK wiederholt den Vorgang für eine begrenzte Anzahl von Malen. Bei Wiederholungen ist der Status „Veröffentlichen/Abonnieren“ `ATTEMPTING_PUBLISH` / `ATTEMPTING_SUBSCRIBE`. Wenn die Wiederholungsversuche erfolgreich sind, ändert sich der Status auf `PUBLISHED` / `SUBSCRIBED`.

Das SDK ruft `onError` mit dem entsprechenden Fehlercode und fatal = false auf.

Aktion: Es ist keine Aktion erforderlich, da das SDK es automatisch wiederholt. Optional kann die Anwendung die Strategie aktualisieren, um weitere Wiederholungsversuche zu erzwingen.

Bereits eingerichtet, dann gescheitert

Eine Veröffentlichung oder ein Abonnement kann nach der Einrichtung fehlschlagen, was höchstwahrscheinlich auf einen Netzwerkfehler zurückzuführen ist. Fehlercode 1400, Meldung: „Die Peer-Verbindung wurde aufgrund eines unbekannten Netzwerkfehlers unterbrochen.“

Dies wird asynchron über den von der Anwendung bereitgestellten Stage-Renderer gemeldet.

Das SDK versucht erneut, den Vorgang zu veröffentlichen/abonnieren. Bei Wiederholungen ist der Status „Veröffentlichen/Abonnieren“ `ATTEMPTING_PUBLISH` / `ATTEMPTING_SUBSCRIBE`. Wenn die Wiederholungsversuche erfolgreich sind, ändert sich der Status auf `PUBLISHED` / `SUBSCRIBED`.

Das SDK ruft `onError` mit dem Fehlercode = 1400 und fatal = false auf.

Aktion: Es ist keine Aktion erforderlich, da das SDK es automatisch wiederholt. Optional kann die Anwendung die Strategie aktualisieren, um weitere Wiederholungsversuche zu erzwingen. Im Falle eines vollständigen Verbindungsverlusts ist es wahrscheinlich, dass auch die Verbindung zu Stages fehlschlägt.

IVS-Broadcast-SDK: iOS-Handbuch | Echtzeit-Streaming

Das iOS-Broadcast-SDK für IVS Echtzeit-Streaming ermöglicht es den Teilnehmern, Videos auf iOS zu senden und zu empfangen.

Das Modul `AmazonIVSBroadcast` implementiert die in diesem Dokument beschriebene Schnittstelle. Folgende Operationen werden unterstützt:

- Einer Stage beitreten
- Medien für andere Teilnehmer auf der Stage veröffentlichen
- Medien anderer Teilnehmer auf der Stage abonnieren
- Auf der Stage veröffentlichte Videos und Audios verwalten und überwachen
- WebRTC-Statistiken für jede Peer-Verbindung beziehen
- Alle Vorgänge aus dem IVS-Streaming-SDK für iOS-Übertragungen mit niedriger Latenz

Aktuelle Version des Broadcast-SDK für iOS: 1.40.0 (Versionshinweise)

Informationen zu den wichtigsten Methoden, die im Amazon-IVS-iOs-Broadcast-SDK verfügbar sind, finden Sie in der Referenzdokumentation unter https://aws.github.io/amazon-ivs-broadcast-docs/1.40.0/ios/.

Beispiel-Code: Siehe das iOS-Beispiel-Repository auf GitHub: https://github.com/aws-samples/amazon-ivs-real-time-streaming-ios-samples.

Plattformanforderungen: iOS 14 und höher

Erste Schritte mit dem IVS iOS Broadcast SDK | Streaming in Echtzeit

Dieses Dokument führt Sie durch die Schritte zum Einstieg in das iOS Broadcast SDK von IVS-Streaming in Echtzeit.

Bibliothek installieren

Wir empfehlen Ihnen, das SDK über Swift Package Manager zu integrieren. (Alternativ können Sie die Framework manuell zu Ihrem Projekt hinzufügen.)

Empfohlen: Integrieren Sie das Broadcast-SDK (Swift Package Manager)

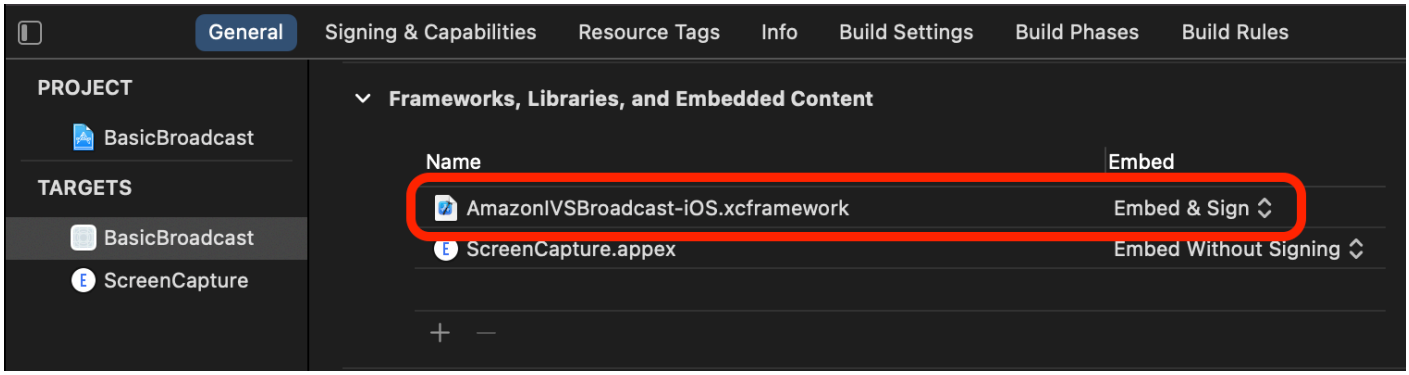
1. Laden Sie die Datei Package.swift von https://broadcast.live-video.net/1.40.0/Package.swift herunter.
2. Erstellen Sie in Ihrem Projekt ein neues Verzeichnis mit dem Namen AmazonIVSBroadcast und fügen Sie es der Versionskontrolle hinzu.
3. Platzieren Sie die heruntergeladene Datei Package.swift im neuen Verzeichnis.

4. Gehen Sie in Xcode zu Datei > Paketabhängigkeiten hinzufügen und wählen Sie Lokal hinzufügen …
5. Navigieren Sie zu dem von Ihnen erstellten AmazonIVSBroadcast-Verzeichnis, wählen Sie es aus und wählen Sie Paket hinzufügen aus.
6. Wenn Sie aufgefordert werden, Paketprodukte für AmazonIVSBroadcast auszuwählen, wählen Sie AmazonIVSBroadcastStages als Ihr Paketprodukt aus, indem Sie Ihr Anwendungsziel im Abschnitt Zum Ziel hinzufügen festlegen.
7. Wählen Sie Paket hinzufügen aus.

Wichtig: Das IVS-Echtzeit-Streaming-Broadcast-SDK beinhaltet alle Feature des IVS-Streaming-Broadcast-SDK mit niedriger Latenz. Es ist nicht möglich, beide SDKs in dasselbe Projekt zu integrieren.

Manuelles Installieren der Framework

1. Laden Sie die neueste Version von https://broadcast.live-video.net/1.40.0/AmazonIVSBroadcast-Stages.xcframework.zip herunter.
2. Extrahieren Sie den Inhalt des Archivs. `AmazonIVSBroadcast.xcframework` enthält das SDK für Gerät und Simulator.
3. Betten Sie `AmazonIVSBroadcast.xcframework` ein, indem Sie es in den Abschnitt Frameworks, Bibliotheken und eingebettete Inhalte auf der Registerkarte Allgemein für Ihr Anwendungsziel ziehen.



Berechtigungen anfordern

Ihre App muss die Berechtigung für den Zugriff auf die Kamera und das Mikrofon des Benutzers anfordern. (Dies ist nicht spezifisch für Amazon IVS; es ist für jede Anwendung erforderlich, die Zugriff auf Kameras und Mikrofone benötigt.)

Hier prüfen wir, ob der Benutzer bereits Berechtigungen erteilt hat und wenn nicht, fragen wir nach ihnen:

```
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.
}
```

Sie müssen dies sowohl für `.video`- als auch für `.audio`-Medientypen tun, wenn Sie auf Kameras bzw. Mikrofone zugreifen möchten.

Sie müssen außerdem Einträge für `NSCameraUsageDescription` und `NSMicrophoneUsageDescription` zu Ihrem `Info.plist` hinzufügen. Andernfalls stürzt Ihre App ab, wenn Sie versuchen, Berechtigungen anzufordern.

Deaktivieren des Idle-Timers der Anwendung

Dies ist zwar optional, wird aber empfohlen. Es verhindert, dass Ihr Gerät in den Ruhezustand versetzt, während Sie das Broadcast-SDK verwenden, was die Übertragung unterbrechen würde.

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

Veröffentlichen und Abonnieren mit dem IVS iOS Broadcast SDK | Streaming in Echtzeit

Dieses Dokument führt Sie durch die Schritte zum Veröffentlichen und Abonnieren einer Stage mit dem iOS Broadcast SDK von IVS-Streaming in Echtzeit.

Konzepte

Drei Kernkonzepte liegen der Echtzeit-Funktionalität zugrunde: Stage, Strategie und Renderer. Das Designziel besteht in der Minimierung der Menge an clientseitiger Logik, die für die Entwicklung eines funktionierenden Produkts erforderlich ist.

Stage

Die Klasse `IVSStage` ist der Hauptinteraktionspunkt zwischen der Hostanwendung und dem SDK. Die Klasse stellt die Stage selbst dar und dient dazu, der Stage beizutreten und sie zu verlassen. Für das Erstellen einer Stage oder das Beitreten ist eine gültige, noch nicht abgelaufene Token-Zeichenfolge aus der Steuerebene erforderlich (dargestellt als `token`). Einer Stage beizutreten und sie zu verlassen, ist ganz einfach.

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

try stage.join()

stage.leave()
```

In der Klasse `IVSStage` erfolgt auch das Anhängen des `IVSStageRenderer` und `IVSErrorDelegate`:

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

Strategie

Über das Protokoll `IVSStageStrategy` kann die Hostanwendung dem SDK den gewünschten Status der Stage mitteilen. Drei Funktionen müssen implementiert werden: `shouldSubscribeToParticipant`, `shouldPublishParticipant` und `streamsToPublishForParticipant`. Alle werden im Folgenden behandelt.

Abonnieren von Teilnehmern

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

Wenn ein Remote-Teilnehmer einer Stage beitritt, fragt das SDK die Hostanwendung nach dessen gewünschtem Abonnementstatus. Die Optionen lauten `.none`, `.audioOnly` und `.audioVideo`.

Wenn ein Wert für diese Funktion zurückgegeben wird, muss sich die Hostanwendung nicht um den Veröffentlichungs-, den aktuellen Abonnement- oder den Verbindungsstatus des Stage kümmern. Bei Rückgabe von `.audioVideo` wartet das SDK mit dem Abonnieren, bis der Remote-Teilnehmer etwas veröffentlicht. Außerdem aktualisiert das SDK die Hostanwendung während des gesamten Prozesses über den Renderer.

Hier folgt ein Beispiel für eine Implementierung:

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

Hierbei handelt es sich um die vollständige Implementierung dieser Funktion für eine Hostanwendung, bei der sich alle Teilnehmer stets gegenseitig sehen sollen; z. B. eine Video-Chat-Anwendung.

Weitergehende Implementierungen sind ebenfalls möglich. Nutzen Sie die Eigenschaft `attributes` für `IVSParticipantInfo`, um Teilnehmer anhand der vom Server bereitgestellten Attribute selektiv zu abonnieren:

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

Hiermit kann eine Stage erstellt werden, auf der Moderatoren alle Gäste überwachen können, ohne selbst gesehen oder gehört zu werden. Die Hostanwendung könnte eine zusätzliche Geschäftslogik nutzen, damit Moderatoren sich gegenseitig sehen können, für Gäste aber unsichtbar bleiben.

Konfiguration für das Abonnieren von Teilnehmern

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

Wenn ein Remote-Teilnehmer abonniert wird (siehe Teilnehmer abonnieren), fragt das SDK die Host-Anwendung nach einer benutzerdefinierten Abonnementkonfiguration für diesen Teilnehmer ab. Diese Konfiguration ist optional und ermöglicht es der Hostanwendung, bestimmte Aspekte des Subscriber-Verhaltens zu steuern. Informationen darüber, was konfiguriert werden kann, finden Sie unter SubscribeConfiguration in der SDK-Referenzdokumentation.

Hier folgt ein Beispiel für eine Implementierung:

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

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

    return config
}
```

Diese Implementierung aktualisiert die Mindestverzögerung für den Jitter-Buffer für alle abonnierten Teilnehmer auf die Voreinstellung `MEDIUM`.

Wie bei `shouldSubscribeToParticipant` sind auch hier weitergehende Implementierungen möglich. Die `ParticipantInfo`-Angaben können verwendet werden, um die Abonnementkonfiguration für bestimmte Teilnehmer selektiv zu aktualisieren.

Wir empfehlen die Verwendung der Standardverhaltensweisen. Geben Sie die benutzerdefinierte Konfiguration nur an, wenn Sie ein bestimmtes Verhalten ändern möchten.

Veröffentlichen

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

Sobald die Verbindung zur Stage hergestellt ist, überprüft das SDK per Anfrage an die Hostanwendung, ob ein bestimmter Teilnehmer etwas veröffentlichen soll. Dies wird nur bei lokalen Teilnehmern aufgerufen, die auf Grundlage des bereitgestellten Tokens zur Veröffentlichung berechtigt sind.

Hier folgt ein Beispiel für eine Implementierung:

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

```
    return true
}
```

Sie ist für eine normale Video-Chat-Anwendung gedacht, bei der Benutzer immer etwas veröffentlichen möchten. Sie können die Audio- und Videowiedergabe stummschalten und die Stummschaltung aufheben, um umgehend ausgeblendet oder gesehen/gehört zu werden. (Sie können auch „Veröffentlichen/Veröffentlichung aufheben“ verwenden, was aber viel langsamer ist. „Stummschalten/Stummschalten aufheben“ ist für Anwendungsfälle vorzuziehen, in denen eine häufige Änderung der Sichtbarkeit wünschenswert ist.)

Auswählen von Streams zur Veröffentlichung

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

Beim Veröffentlichen wird hiermit bestimmt, welche Audio- und Videostreams veröffentlicht werden sollen. Dieser Punkt wird später unter Veröffentlichen eines Medienstreams ausführlicher behandelt.

Aktualisieren der Strategie

Die Strategie soll dynamisch sein: Die von einer der oben genannten Funktionen zurückgegebenen Werte lassen sich jederzeit ändern. Wenn die Hostanwendung beispielsweise erst veröffentlichen soll, wenn der Endbenutzer auf eine Schaltfläche tippt, können Sie eine Variable aus `shouldPublishParticipant` zurückgeben (zum Beispiel `hasUserTappedPublishButton`). Wenn sich diese Variable aufgrund einer Interaktion des Endbenutzers ändert, signalisieren Sie dem SDK per Aufruf von `stage.refreshStrategy()`, dass es die Strategie nach den neuesten Werten abfragen und nur Dinge anwenden soll, die sich geändert haben. Wenn das SDK feststellt, dass sich der Wert `shouldPublishParticipant` geändert hat, startet es den Veröffentlichungsprozess. Wenn alle Funktionen bei einer SDK-Abfrage den gleichen Wert zurückgeben wie zuvor, werden mit dem Aufruf von `refreshStrategy` keine Änderungen an der Stage durchgeführt.

Ändert sich der Rückgabewert von `shouldSubscribeToParticipant` von `.audioVideo` in `.audioOnly`, wird der Videostream für alle Teilnehmer mit geänderten Rückgabewerten entfernt, sofern zuvor ein Videostream vorhanden war.

Im Allgemeinen nutzt die Stage die Strategie, um den Unterschied zwischen der vorherigen und der aktuellen Strategie am effizientesten anzuwenden. Dabei muss sich die Hostanwendung nicht um die ganzen Status kümmern, die für eine ordnungsgemäße Verwaltung erforderlich sind. Stellen Sie sich den Aufruf von `stage.refreshStrategy()` daher als einen ressourcenschonenden Vorgang vor, da nur bei einer Änderung der Strategie etwas unternommen wird.

Renderer

Das Protokoll `IVSStageRenderer` teilt der Hostanwendung den Status der Stage mit. Aktualisierungen in der Benutzeroberfläche der Hostanwendung können in der Regel vollständig über die vom Renderer bereitgestellten Ereignisse gesteuert werden. Der Renderer stellt die folgenden Funktionen bereit:

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

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

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

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

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

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

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

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

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

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

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

Es wird nicht erwartet, dass sich die vom Renderer bereitgestellten Informationen auf die Rückgabewerte der Strategie auswirken. Es wird beispielsweise nicht erwartet, dass sich der Rückgabewert von `shouldSubscribeToParticipant` beim Aufruf von

`participant:didChangePublishState` ändert. Wenn die Hostanwendung einen bestimmten Teilnehmer abonnieren möchte, muss sie unabhängig von dessen Veröffentlichungsstatus den gewünschten Abonnementtyp zurückgeben. Das SDK muss dafür sorgen, dass entsprechend dem Status der Stage und dem gewünschten Status der Strategie zum richtigen Zeitpunkt gehandelt wird.

Hinweis: Nur veröffentlichende Teilnehmer lösen `participantDidJoin` aus. Wenn Teilnehmer die Veröffentlichung beenden oder die Stagesitzung verlassen, wird `participantDidLeave` ausgelöst.

Veröffentlichen eines Medienstreams

Lokale Geräte wie eingebaute Mikrofone und Kameras werden über `IVSDeviceDiscovery` erkannt. Hier folgt ein Beispiel für die Auswahl der nach vorne gerichteten Kamera und des Standardmikrofons und deren anschließende Rückgabe als `IVSLocalStageStreams` zur Veröffentlichung durch das 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]
}
```

Anzeigen und Entfernen von Teilnehmern

Nach Abschluss von Abonnements erhalten Sie über die Funktion `didAddStreams` des Renderers eine Reihe von `IVSStageStream`-Objekten. Um Statistiken des Audiolevels zu diesem Teilnehmer

in der Vorschau anzuzeigen oder abzurufen, können Sie über den Stream auf das zugrunde liegende `IVSDevice`-Objekt zugreifen:

```
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 */
    })
}
```

Wenn ein Teilnehmer die Veröffentlichung beendet oder dessen Abonnement beendet wird, wird die Funktion `didRemoveStreams` mit den Streams aufgerufen, die entfernt wurden. Hostanwendungen sollten dies als Signal nutzen, um den Videostream des Teilnehmers aus der Ansichtshierarchie zu entfernen.

`didRemoveStreams` wird für alle Szenarien aufgerufen, in denen ein Stream entfernt werden könnte, darunter:

- Der Remote-Teilnehmer beendet die Veröffentlichung.
- Ein lokales Gerät beendet das Abonnement oder ändert das Abonnement von `.audioVideo` in `.audioOnly`.
- Der Remote-Teilnehmer verlässt die Stage.
- Der lokale Teilnehmer verlässt die Stage.

Da `didRemoveStreams` bei allen Szenarien aufgerufen wird, ist keine benutzerdefinierte Geschäftslogik erforderlich, um Teilnehmer beim remoten oder lokalen Verlassen aus der Benutzeroberfläche zu entfernen.

Stummschalten von Medienstreams und Aufheben der Stummschaltung

`IVSLocalStageStream`-Objekte verfügen über eine `setMuted`-Funktion, die das Stummschalten des Streams steuert. Diese Funktion kann für den Stream aufgerufen werden, bevor oder nachdem er von der Strategiefunktion `streamsToPublishForParticipant` zurückgegeben wird.

Wichtig: Wenn nach einem Aufruf von `refreshStrategy` eine neue `IVSLocalStageStream`-Objekt-Instance von `streamsToPublishForParticipant` zurückgegeben wird, wird der Stummschaltungsstatus des neuen Streamobjekts auf die Stage angewendet. Seien Sie vorsichtig

beim Erstellen neuer `IVSLocalStageStream`-Instances, um sicherzustellen, dass der erwartete Stummschaltungsstatus beibehalten wird.

Überwachen des Medien-Stummschaltungsstatus von Remote-Teilnehmern

Wenn ein Teilnehmer den Stummschaltungsstatus seines Video- oder Audiostreams ändert, wird die Funktion `didChangeMutedStreams` des Renderers mit einer Gruppe der Streams aufgerufen, die sich geändert haben. Verwenden Sie die Eigenschaft `isMuted` für `IVSStageStream`, um die Benutzeroberfläche entsprechend zu aktualisieren:

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

Erstellen einer Stagekonfiguration

Die Werte der Videokonfiguration einer Stage passen Sie mit `IVSLocalStageStreamVideoConfiguration` an:

```
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
```

Abrufen von WebRTC-Statistiken

Um die neuesten WebRTC-Statistiken für einen veröffentlichten oder abonnierten Stream abzurufen, verwenden Sie `requestRTCStats` für `IVSStageStream`. Nach Abschluss einer Erfassung erhalten Sie Statistiken über den `IVSStageStreamDelegate`, der für `IVSStageStream` eingestellt werden kann. Um WebRTC-Statistiken kontinuierlich zu erfassen, rufen Sie diese Funktion für einen `Timer` auf.

```
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)")
      }
   }
}
```

Abrufen von Teilnehmerattributen

Wenn Sie Attribute in der Vorgangsanfrage `CreateParticipantToken` angeben, können Sie die Attribute in den Eigenschaften von `IVSParticipantInfo` einsehen:

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

Eingebettete Nachrichten

Die `embedMessage` Methode auf IVSImageDevice ermöglicht es Ihnen, Metadaten-Nutzdaten während der Veröffentlichung direkt in Videoframes einzufügen. Dies ermöglicht framesynchronisiertes Messaging für Echtzeitanwendungen. Das Einbetten von Nachrichten ist nur verfügbar, wenn das SDK für die Veröffentlichung in Echtzeit verwendet wird (nicht für Veröffentlichungen mit niedriger Latenz).

Es kann nicht garantiert werden, dass eingebettete Nachrichten bei Abonnenten ankommen, da sie direkt in Videoframes eingebettet und über UDP übertragen werden, wodurch die Paketzustellung nicht garantiert wird. Der Verlust von Paketen während der Übertragung kann zum Verlust von Nachrichten führen, insbesondere bei schlechten Netzwerkbedingungen. Um dem entgegenzuwirken, beinhaltet die `embedMessage`-Methode einen `repeatCount`-Parameter, der die Nachricht über mehrere aufeinanderfolgende Frames dupliziert und so die Zuverlässigkeit der Zustellung erhöht. Diese Funktion ist nur für Videostreams verfügbar.

Verwenden von embedMessage

Publishing-Clients können Nachrichten-Nutzdaten mithilfe der `embedMessage`-Methode auf IVSImageDevice in ihren Videostream einbetten. Die Nutzdaten müssen größer als 0 KB und kleiner als 1 KB sein. Die Anzahl der pro Sekunde eingebetteten Nachrichten darf 10 KB pro Sekunde nicht überschreiten.

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

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

Nachrichten-Nutzdaten

Verwenden Sie `repeatCount`, um die Nachricht über mehrere Frames hinweg zu duplizieren, um die Zuverlässigkeit zu erhöhen. Dieser Wert muss zwischen 0 und 30 liegen. Empfangende Clients müssen über eine Logik verfügen, um die Nachricht zu deduplizieren.

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

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

Lesen eingebetteter Nachrichten

Informationen zum Lesen eingebetteter Nachrichten aus eingehenden Streams finden Sie weiter unten unter „Zusätzliche Erweiterungsinformationen (SEI) abrufen“.

Abrufen von SEI-Daten (Supplemental Enhancement Information)

Die NAL-Einheit für Supplemental Enhancement Information (SEI) wird verwendet, um Frame-orientierte Metadaten zusammen mit dem Video zu speichern. Subscriber können SEI-Nutzlasten von einem Publisher lesen, der H.264-Video veröffentlicht, indem sie die Eigenschaften `embeddedMessages` der `IVSImageDeviceFrame`-Objekte aus dem `IVSImageDevice` des Publishers überprüfen. Erlangen Sie dazu das `IVSImageDevice` eines Publishers und beobachten Sie dann jeden Frame über einen Callback an `setOnFrameCallback`, wie im folgenden Beispiel gezeigt:

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

let imageDevice: IVSImageDevice? = imageStream.device as? IVSImageDevice
imageDevice?.setOnFrameCallback { frame in
 for message in frame.embeddedMessages {
```

```
      if let seiMessage = message as? IVSUserDataUnregisteredSEIMessage {
          let seiMessageData = seiMessage.data
          let seiMessageUUID = seiMessage.UUID

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

Fortsetzen der Sitzung im Hintergrund

Wenn die App in den Hintergrund wechselt, können Sie weiterhin auf der Stage präsent sein, während Sie Remote-Ton hören. Es ist jedoch nicht möglich, das eigene Bild und den eigenen Ton weiterhin zu senden. Sie müssen die Implementierung Ihrer `IVSStrategy` aktualisieren, um die Veröffentlichung zu beenden und `.audioOnly` zu abonnieren (oder gegebenenfalls `.none`):

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

Anschließend rufen Sie `stage.refreshStrategy()` auf.

Mehrschichtige Kodierung mit Simulcast

Bei der mehrschichtigen Kodierung mit Simulcast handelt es sich um ein Feature für IVS-Echtzeit-Streaming, mit dessen Hilfe Publisher mehrere Videoschichten unterschiedlicher Qualität senden können. Subscriber können diese Schichten dynamisch oder manuell konfigurieren. Das Feature wird im Dokument Streaming-Optimierungen ausführlicher beschrieben.

Konfigurieren mehrschichtiger Kodierung (Publisher)

Um als Publisher die mehrschichtige Kodierung mit Simulcast zu aktivieren, fügen Sie dem `IVSLocalStageStream` bei der Instanziierung die folgende Konfiguration hinzu:

```
// Enable Simulcast
let config = IVSLocalStageStreamVideoConfiguration()
```

```
config.simulcast.enabled = true

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

// Other Stage implementation code
```

Je nach der in der Videokonfiguration eingestellten Auflösung wird eine festgelegte Anzahl von Schichten kodiert und gesendet, wie im Abschnitt Standardmäßige Schichten, Qualitäten und Bildraten von Streaming-Optimierungen definiert.

Außerdem können Sie optional einzelne Ebenen innerhalb der Simulcast-Konfiguration konfigurieren:

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

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

try config.simulcast.setLayers(layers)

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

// Other Stage implementation code
```

Alternativ können Sie eigene benutzerdefinierte Ebenenkonfigurationen für bis zu drei Ebenen erstellen. Wenn Sie ein leeres Array oder keinen Wert angeben, werden die oben beschriebenen Standardwerte verwendet. Ebenen werden mit den folgenden erforderlichen Eigenschaftssetzern beschrieben:

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

Ausgehend von den Voreinstellungen können Sie entweder einzelne Eigenschaften überschreiben oder eine völlig neue Konfiguration erstellen:

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

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

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

try config.simulcast.setLayers(layers)

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

// Other Stage implementation code
```

Informationen zu Höchstwerten, Grenzwerten und Fehlern, die bei der Konfiguration einzelner Ebenen ausgelöst werden können, finden Sie in der SDK-Referenzdokumentation.

Konfigurieren mehrschichtiger Kodierung (Subscriber)

Subscriber müssen nichts unternehmen, um die mehrschichtige Kodierung zu aktivieren. Wenn ein Publisher Simulcast-Schichten sendet, passt sich der Server standardmäßig dynamisch den Schichten an, um je nach Gerät und Netzwerkbedingungen des Subscribers die optimale Qualität auszuwählen.

Alternativ gibt es mehrere nachfolgend beschriebene Optionen, um explizite Schichten auszuwählen, die der Publisher sendet.

Option 1: Einstellung für die Qualität der Anfangsschicht

Mit der Strategie `subscribeConfigurationForParticipant` können Sie auswählen, welche Anfangsschicht Sie als Subscriber erhalten möchten:

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

    config.simulcast.initialLayerPreference = .lowestQuality
```

```
    return config
}
```

Standardmäßig wird Subscribern zunächst immer die Schicht mit der niedrigsten Qualität gesendet. Nach und nach wird die Qualität gesteigert, bis die Schicht mit der höchsten Qualität erreicht ist. Das optimiert den Bandbreitenverbrauch der Endbenutzer, verkürzt die Zeit bis zum Abspielen des Videos und verringert das anfängliche Einfrieren von Videos bei Benutzern in Netzwerken mit geringerer Bandbreite.

Folgende Optionen sind für `InitialLayerPreference` verfügbar:

- `lowestQuality` – Der Server stellt zuerst die Videoschicht mit der niedrigsten Qualität bereit. Dadurch werden der Bandbreitenverbrauch und die Zeit bis zum Abspielen von Medien optimiert. Die Qualität ist definiert als die Kombination aus Größe, Bitrate und Bildrate des Videos. Beispielsweise weisen 720p-Videos eine geringere Qualität auf als 1080p-Videos.
- `highestQuality` – Der Server stellt zuerst die Videoschicht mit der höchsten Qualität bereit. Das optimiert die Qualität, kann aber die Zeit bis zum Abspielen von Medien verlängern. Die Qualität ist definiert als die Kombination aus Größe, Bitrate und Bildrate des Videos. Beispielsweise weisen 1080p-Videos eine höhere Qualität auf als 720p-Videos.

Hinweis: Damit die anfänglichen Schichteinstellungen (der Aufruf `initialLayerPreference`) wirksam werden, muss ein neues Abonnement abgeschlossen werden, da diese Updates für das aktive Abonnement nicht gelten.

Option 2: Bevorzugte Schicht für Streams

Mit der Strategiemethode `preferredLayerForStream` können Sie eine Schicht auswählen, nachdem der Stream gestartet wurde. Diese Strategiemethode erhält die Teilnehmer- und Streaminformationen, sodass Sie eine Schicht für jeden Teilnehmer auswählen können. Das SDK ruft diese Methode als Reaktion auf bestimmte Ereignisse auf, z. B. wenn sich die Streamschichten ändern, sich der Teilnehmerstatus ändert oder die Hostanwendung die Strategie aktualisiert.

Der Strategiemodus gibt ein `IVSRemoteStageStreamLayer`-Objekt zurück, wobei es sich um Folgendes handeln kann:

- ein Schichtobjekt, z. B. eines, das von `IVSRemoteStageStream.layers` zurückgegeben wird
- null, was bedeutet, dass keine Schicht ausgewählt werden sollte und eine dynamische Anpassung bevorzugt wird

Bei der folgenden Strategie wählen die Benutzer beispielsweise immer die Videoschicht mit der niedrigsten verfügbaren Qualität aus:

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

Um die Schichtauswahl zurückzusetzen und zur dynamischen Anpassung zurückzukehren, geben Sie in der Strategie null oder undefiniert zurück. In diesem Beispiel ist `appState` eine Platzhaltervariable, die den Status der Hostanwendung darstellt.

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

Option 3: Helferobjekte für RemoteStageStream-Schicht

`IVSRemoteStageStream` weist mehrere Helferobjekte auf, mit deren Hilfe Entscheidungen über die Schichtauswahl getroffen und Endbenutzern die entsprechende Auswahl angezeigt werden kann:

- Schichtereignisse – Neben `IVSStageRenderer` verfügt der `IVSRemoteStageStreamDelegate` über Ereignisse, die Änderungen bei der Schicht- und Simulcast-Anpassung kommunizieren:
  - `func stream(_ stream: IVSRemoteStageStream, didChangeAdaption adaption: Bool)`
  - `func stream(_ stream: IVSRemoteStageStream, didChange layers: [IVSRemoteStageStreamLayer])`
  - `func stream(_ stream: IVSRemoteStageStream, didSelect layer: IVSRemoteStageStreamLayer?, reason: IVSRemoteStageStream.LayerSelectedReason)`
- Schichtmethoden – `IVSRemoteStageStream` verfügt über mehrere Helfermethoden, mit denen Informationen über den Stream und die präsentierten Schichten abgerufen werden können. Diese Methoden sind sowohl für den in der Strategie `preferredLayerForStream`

bereitgestellten Remote-Stream als auch für Remote-Streams verfügbar, die über `func stage(_ stage: IVSStage, participant: IVSParticipantInfo, didAdd streams: [IVSStageStream])` verfügbar gemacht werden.

- `stream.layers`
- `stream.selectedLayer`
- `stream.lowestQualityLayer`
- `stream.highestQualityLayer`
- `stream.layers(with: IVSRemoteStageStreamLayerConstraints)`

Einzelheiten finden Sie im Abschnitt zur Klasse `IVSRemoteStageStream` in der SDK-Referenzdokumentation. Falls als Grund für `LayerSelected` `UNAVAILABLE` zurückgegeben wird, bedeutet das, dass die angeforderte Schicht nicht ausgewählt werden konnte. Stattdessen wird eine bestmögliche Auswahl getroffen. Dabei handelt es sich in der Regel um eine Schicht mit niedrigerer Qualität, um die Stabilität des Streams zu gewährleisten.

Übertragung der Stage auf einen IVS-Kanal

Zum Übertragen einer Stage erstellen Sie eine separate `IVSBroadcastSession` und folgen Sie dann den oben beschriebenen üblichen Anweisungen für die Übertragung mit dem SDK. Die Eigenschaft `device` für `IVSStageStream` ist entweder ein `IVSImageDevice` oder ein `IVSAudioDevice`, wie im obigen Snippet veranschaulicht. Diese können mit dem `IVSBroadcastSession.mixer` verbunden werden, um die gesamte Stage in einem individuell anpassbaren Layout zu übertragen.

Optional können Sie eine Stage zusammenstellen und sie auf einen IVS-Kanal mit niedriger Latenz übertragen, um ein größeres Publikum zu erreichen. Sehen Sie Aktivierung mehrerer Hosts in einem Amazon-IVS-Stream im Benutzerhandbuch für IVS-Streaming mit niedriger Latenz.

So wählt iOS Kameraauflösung und Bildrate

Die vom Broadcast-SDK verwaltete Kamera optimiert ihre Auflösung und Bildrate (Bilder pro Sekunde oder FPS), um die Wärmeentwicklung und den Energieverbrauch zu minimieren. In diesem Abschnitt wird erläutert, wie Auflösung und Bildrate ausgewählt werden, um Hostanwendungen bei der Optimierung für ihre Anwendungsfälle zu unterstützen.

Wenn ein `IVSLocalStageStream` mit einer `IVSCamera` erstellt wird, ist die Kamera für eine Bildrate von `IVSLocalStageStreamVideoConfiguration.targetFramerate` und eine

Auflösung von `IVSLocalStageStreamVideoConfiguration.size` optimiert. Das Aufrufen von `IVSLocalStageStream.setConfiguration` aktualisiert die Kamera mit neueren Werten.

Kameravorschau

Wenn Sie eine Vorschau einer `IVSCamera` erstellen, ohne sie mit einer `IVSBroadcastSession` oder `IVSStage` zu verbinden, wird standardmäßig eine Auflösung von 1080p und eine Bildrate von 60 Bildern pro Sekunde verwendet.

Übertragen einer Stage

Bei der Verwendung einer `IVSBroadcastSession` zu Übertragung einer `IVSStage` versucht das SDK, die Kamera mit einer Auflösung und Bildrate zu optimieren, die die Kriterien beider Sitzungen erfüllen.

Wenn die Broadcast-Konfiguration beispielsweise auf eine Bildrate von 15 FPS und eine Auflösung von 1080p eingestellt ist, während die Stage eine Bildrate von 30 FPS und eine Auflösung von 720p hat, wählt das SDK eine Kamerakonfiguration mit einer Bildrate von 30 FPS und einer Auflösung von 1080p aus. Bei der `IVSBroadcastSession` fehlt jedes zweite Bild von der Kamera und die `IVSStage` skaliert das 1080p-Bild auf 720p herunter.

Wenn eine Host-Anwendung plant, sowohl eine `IVSBroadcastSession` als auch eine `IVSStage` zusammen mit einer Kamera zu verwenden, empfehlen wir, dass die Eigenschaften `targetFramerate` und `size` der jeweiligen Konfigurationen übereinstimmen. Eine Nichtübereinstimmung kann dazu führen, dass sich die Kamera während der Videoaufnahme selbst neu konfiguriert, was zu einer kurzen Verzögerung bei der Übertragung von Videos führt.

Wenn identische Werte nicht dem Anwendungsfall der Hostanwendung entsprechen, verhindert das Erstellen der Kamera mit höherer Qualität, dass sich die Kamera selbst neu konfiguriert, wenn die Sitzung mit niedrigerer Qualität hinzugefügt wird. Wenn Sie zum Beispiel mit 1080p und 30 Bildern pro Sekunde übertragen und später einer Stage beitreten, die auf 720p und 30 FPS eingestellt ist, konfiguriert sich die Kamera nicht selbst neu und die Videowiedergabe läuft ohne Unterbrechung weiter. Dies liegt daran, dass 720p kleiner oder gleich 1080p und 30 FPS kleiner oder gleich 30 FPS sind.

Beliebige Bildraten, Auflösungen und Seitenverhältnisse

Die meisten Kameras können gängige Formate wie 720p bei 30 FPS oder 1080p bei 60 FPS exakt wiedergeben. Es ist jedoch unmöglich, alle Formate exakt wiederzugeben. Das Broadcast-SDK

wählt die Kamerakonfiguration auf der Grundlage der folgenden Regeln aus (in der Reihenfolge ihrer Priorität):

1. Breite und Höhe der Auflösung sind größer oder gleich der gewünschten Auflösung, aber innerhalb dieser Beschränkung sind Breite und Höhe so klein wie möglich.
2. Die Bildrate ist größer oder gleich der gewünschten Bildrate, aber innerhalb dieser Beschränkung ist die Bildrate so niedrig wie möglich.
3. Das Seitenverhältnis entspricht dem gewünschten Seitenverhältnis.
4. Wenn es mehrere passende Formate gibt, wird das Format mit dem größten Sichtfeld verwendet.

Nachfolgend finden Sie zwei Beispiele:

- Die Host-Anwendung versucht, in 4K mit 120 FPS zu übertragen. Die ausgewählte Kamera unterstützt nur 4K bei 60 FPS oder 1080p bei 120 FPS. Das gewählte Format ist 4K bei 60 FPS, da die Auflösungsregel eine höhere Priorität hat als die Bildratenregel.
- Eine unregelmäßige Auflösung wird angefordert, 1910x1070. Die Kamera wird 1920x1080 verwenden. Vorsicht: Wenn Sie eine Auflösung wie 1921x1080 wählen, skaliert die Kamera auf die nächste verfügbare Auflösung (z. B. 2592x1944) hoch, was sich nachteilig auf die CPU- und Speicherbandbreite auswirkt.

Was ist mit Android?

Android passt seine Auflösung oder Bildrate nicht wie iOS im laufenden Betrieb an, sodass dies keine Auswirkungen auf das Android-Broadcast-SDK hat.

Bekannte Probleme und Behelfslösungen im IVS iOS Broadcast SDK | Streaming in Echtzeit

In diesem Dokument werden bekannte Probleme aufgeführt, die bei der Verwendung des iOS Broadcast SDK von Amazon-IVS-Streaming in Echtzeit auftreten können, und es werden mögliche Problemumgehungen vorgeschlagen.

- Das Ändern von Bluetooth-Audiorouten kann unvorhersehbar sein. Wenn Sie ein neues Gerät in der Mitte der Sitzung verbinden, kann iOS die Eingaberoute automatisch ändern oder nicht. Es ist auch nicht möglich, zwischen mehreren Bluetooth-Headsets zu wählen, die gleichzeitig verbunden sind. Dies geschieht sowohl bei normalen Broadcast- als auch bei Stagesitzungen.

Problemumgehung: Wenn Sie ein Bluetooth-Headset verwenden möchten, verbinden Sie es, bevor Sie den Broadcast oder die Stage starten und lassen Sie es während der gesamten Sitzung verbunden.

- Teilnehmer, die ein iPhone 14, iPhone 14 Plus, iPhone 14 Pro oder iPhone 14 Pro Max nutzen, können bei anderen Teilnehmern ein Echo verursachen.

  Problemumgehung: Teilnehmer, die die betroffenen Geräte nutzen, können Kopfhörer verwenden, um das Echo bei anderen Teilnehmern zu vermeiden.

- Wenn ein Teilnehmer mit einem Token beitritt, das von einem anderen Teilnehmer verwendet wird, wird die erste Verbindung ohne einen bestimmten Fehler getrennt.

  Problemumgehung: Keine.

- Es gibt ein seltenes Problem, bei dem der Publisher etwas veröffentlicht, der Veröffentlichungsstatus, den Subscriber erhalten, jedoch `inactive` lautet.

  Problemumgehung: Versuchen Sie, die Sitzung zu verlassen und ihr wieder beizutreten. Wenn das Problem weiterhin besteht, erstellen Sie ein neues Token für den Publisher.

- Wenn ein Teilnehmer etwas veröffentlicht oder abonniert, kann selbst bei einem stabilen Netzwerk eine Fehlermeldung mit dem Code 1400 angezeigt werden. Sie weist darauf hin, dass die Verbindung aufgrund eines Netzwerkproblems unterbrochen wurde.

  Problemumgehung: Versuchen Sie, erneut zu veröffentlichen bzw. erneut zu abonnieren.

- Während einer Stagesitzung kann zeitweise ein seltenes Problem mit Tonverzerrungen auftreten, in der Regel bei längeren Anrufen.

  Problemumgehung: Der Teilnehmer mit dem verzerrtem Ton kann die Sitzung entweder verlassen und erneut beitreten oder die Veröffentlichung des Audios aufheben und dann erneut veröffentlichen.

Fehlerbehandlung im IVS iOS Broadcast SDK | Streaming in Echtzeit

Dieser Abschnitt gibt einen Überblick über die Fehlerbedingungen, wie das iOS Broadcast SDK von IVS-Streaming in Echtzeit sie an die Anwendung meldet und wie eine Anwendung reagieren sollte, wenn diese Fehler auftreten.

Schwerwiegende und nicht schwerwiegende Fehler

Das Fehlerobjekt hat den booleschen Wert „ist fatal“. Dies ist ein Wörterbucheintrag unter `IVSBroadcastErrorIsFatalKey`, der einen booleschen Wert enthält.

Im Allgemeinen hängen schwerwiegende Fehler mit der Verbindung zum Stages-Server zusammen (entweder kann eine Verbindung nicht hergestellt werden oder sie ist verloren gegangen und kann nicht wiederhergestellt werden). Die Anwendung sollte die Stage neu erstellen und erneut beitreten, ggf. mit einem neuen Token oder wenn die Konnektivität des Geräts wiederhergestellt ist.

Fehler, die nicht schwerwiegend sind, hängen in der Regel mit dem Status „Veröffentlichen/Abonnieren“ zusammen und werden vom SDK behandelt, das den Vorgang zum Veröffentlichen/Abonnieren erneut versucht.

Sie können diese Eigenschaft überprüfen:

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

Beitrittsfehler

Fehlerhaft formatiertes Token

Dies passiert, wenn das Stage-Token falsch formatiert ist.

Das SDK löst eine Swift-Ausnahme mit dem Fehlercode = 1000 und IVSBroadcastErrorIsFatalKey = YES aus.

Aktion: Erstellen Sie ein gültiges Token und versuchen Sie erneut beizutreten.

Abgelaufenes Token

Dies passiert, wenn das Stage-Token abgelaufen ist.

Das SDK löst eine Swift-Ausnahme mit dem Fehlercode = 1001 und IVSBroadcastErrorIsFatalKey = YES aus.

Aktion: Erstellen Sie ein neues Token und versuchen Sie erneut beizutreten.

Ungültiges oder widerrufenes Token

Dies passiert, wenn das Stage-Token nicht falsch formatiert ist, sondern vom Stages-Server zurückgewiesen wird. Dies wird asynchron über den von der Anwendung bereitgestellten Stage-Renderer gemeldet.

Das SDK ruft `stage(didChange connectionState, withError error)` mit dem Fehlercode = 1026 und IVSBroadcastErrorIsFatalKey = YES auf.

Aktion: Erstellen Sie ein gültiges Token und versuchen Sie erneut beizutreten.

Netzwerkfehler beim ersten Beitritt

Dies passiert, wenn das SDK den Stages-Server nicht kontaktieren kann, um eine Verbindung herzustellen. Dies wird asynchron über den von der Anwendung bereitgestellten Stage-Renderer gemeldet.

Das SDK ruft `stage(didChange connectionState, withError error)` mit dem Fehlercode = 1300 und IVSBroadcastErrorIsFatalKey = YES auf.

Handlung: Warten Sie, bis die Konnektivität des Geräts wiederhergestellt ist, und versuchen Sie erneut, eine Verbindung herzustellen.

Netzwerkfehler, wenn bereits eine Verbindung hergestellt wurde

Wenn die Netzwerkverbindung des Geräts ausfällt, verliert das SDK möglicherweise die Verbindung zu den Stage-Servern. Dies wird asynchron über den von der Anwendung bereitgestellten Stage-Renderer gemeldet.

Das SDK ruft `stage(didChange connectionState, withError error)` mit dem Fehlercode = 1300 und dem Wert IVSBroadcastErrorIsFatalKey = YES auf.

Handlung: Warten Sie, bis die Konnektivität des Geräts wiederhergestellt ist, und versuchen Sie erneut, eine Verbindung herzustellen.

Fehler beim Veröffentlichen/Abonnieren

Anfänglich

Es gibt mehrere Arten von Fehlern:

- MultihostSessionOfferCreationFailPublish (1.020)

- MultihostSessionOfferCreationFailSubscribe (1.021)
- MultihostSessionNoIceCandidates (1.022)
- MultihostSessionStageAtCapacity (1.024)
- SignallingSessionCannotRead (1.201)
- SignallingSessionCannotSend (1.202)
- SignallingSessionBadResponse (1.203)

Diese werden asynchron über den von der Anwendung bereitgestellten Stage-Renderer gemeldet.

Das SDK wiederholt den Vorgang für eine begrenzte Anzahl von Malen. Bei Wiederholungen ist der Status „Veröffentlichen/Abonnieren“ `ATTEMPTING_PUBLISH` / `ATTEMPTING_SUBSCRIBE`. Wenn die Wiederholungsversuche erfolgreich sind, ändert sich der Status auf `PUBLISHED` / `SUBSCRIBED`.

Das SDK ruft `IVSErrorDelegate:didEmitError` mit dem entsprechenden Fehlercode und `IVSBroadcastErrorIsFatalKey == NO` auf.

Aktion: Es ist keine Aktion erforderlich, da das SDK es automatisch wiederholt. Optional kann die Anwendung die Strategie aktualisieren, um weitere Wiederholungsversuche zu erzwingen.

Bereits eingerichtet, dann gescheitert

Eine Veröffentlichung oder ein Abonnement kann nach der Einrichtung fehlschlagen, was höchstwahrscheinlich auf einen Netzwerkfehler zurückzuführen ist. Fehlercode 1400, Meldung: „Die Peer-Verbindung wurde aufgrund eines unbekannten Netzwerkfehlers unterbrochen.“

Dies wird asynchron über den von der Anwendung bereitgestellten Stage-Renderer gemeldet.

Das SDK versucht erneut, den Vorgang zu veröffentlichen/abonnieren. Bei Wiederholungen ist der Status „Veröffentlichen/Abonnieren“ `ATTEMPTING_PUBLISH` / `ATTEMPTING_SUBSCRIBE`. Wenn die Wiederholungsversuche erfolgreich sind, ändert sich der Status auf `PUBLISHED` / `SUBSCRIBED`.

Das SDK ruft `didEmitError` mit dem Fehlercode = 1400 und IVSBroadcastErrorIsFatalKey = NO auf.

Aktion: Es ist keine Aktion erforderlich, da das SDK es automatisch wiederholt. Optional kann die Anwendung die Strategie aktualisieren, um weitere Wiederholungsversuche zu erzwingen. Im Falle eines vollständigen Verbindungsverlusts ist es wahrscheinlich, dass auch die Verbindung zu Stages fehlschlägt.

IVS-Broadcast-SDK: Gemischte Geräte

Gemischte Geräte sind Audio- und Videogeräte, die mehrere Eingangsquellen auf einen einzigen Ausgang kombinieren. Geräte mischen ist ein leistungsstarkes Feature, mit dem Sie mehrere Bildschirmelemente (Video) und Audiospuren definieren und verwalten können. Sie können Video und Audio aus mehreren Quellen wie Kameras, Mikrofonen, Bildschirmaufnahmen sowie von Ihrer App generiertes Audio und Video kombinieren. Sie können mit Übergängen diese Quellen im Video, das Sie an IVS streamen, verschieben und während des Streamings Quellen hinzufügen oder entfernen.

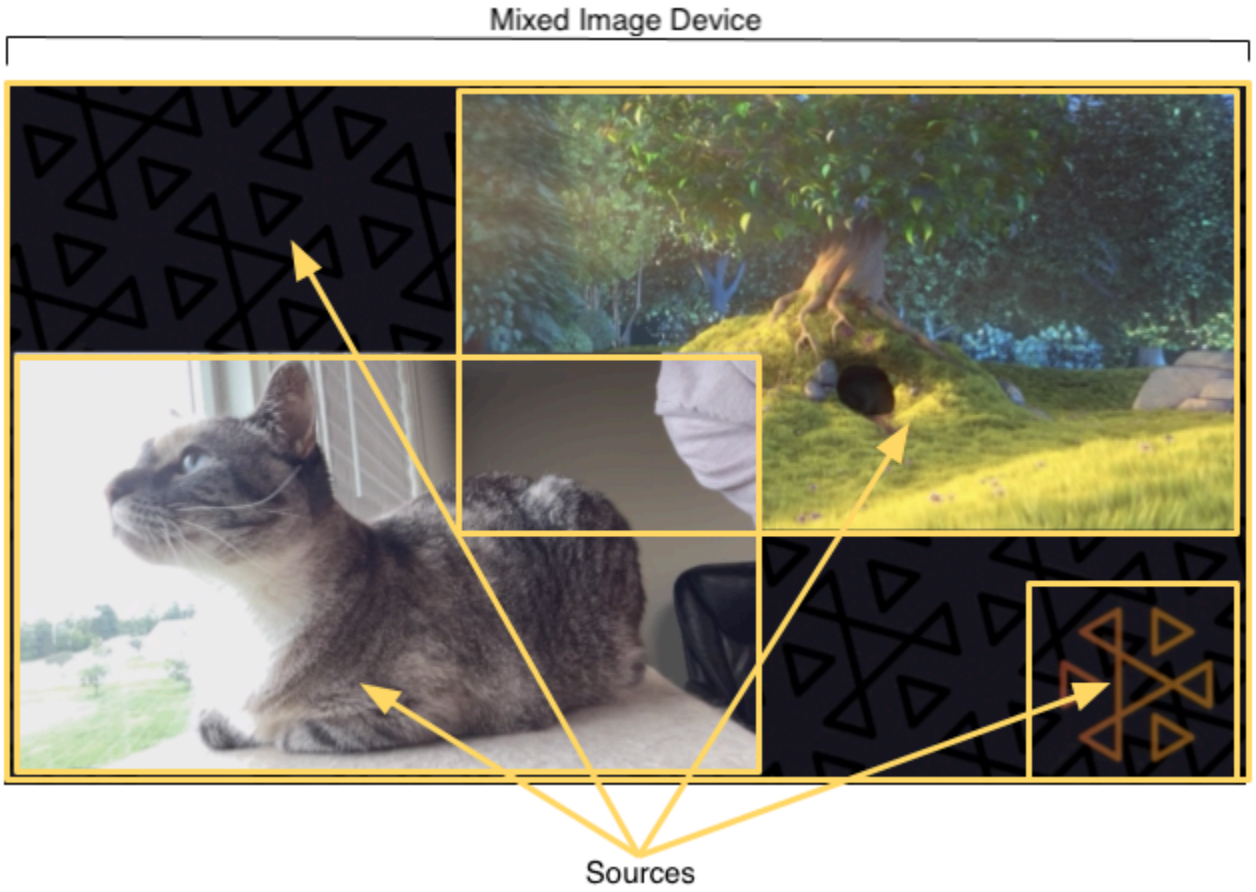
Gemischte Geräte gibt es in Bild- und Audioausführungen. Um ein Gemischtes-Bild-Gerät zu erstellen, rufen Sie auf:

`DeviceDiscovery.createMixedImageDevice()` auf Android

`IVSDeviceDiscovery.createMixedImageDevice()` auf iOS

Das zurückgegebene Gerät kann wie jedes andere Gerät an eine `BroadcastSession` (Streaming mit niedriger Latenz) oder `Stage` (Echtzeit-Streaming) angeschlossen werden.

Terminologie



Begriff	Beschreibung
Gerät	Eine Hardware- oder Softwarekomponente, die Audio- oder Bild-Eingaben produziert. Beispiele für Geräte sind Mikrofone, Kameras, Bluetooth-Headsets und virtuelle Geräte wie Bildschirmaufnahmen oder benutzerdefinierte Image-Ein gänge.
Gemischtes Gerät	Ein `Device`, das wie jedes andere `Device` an ein `BroadcastSession` angehängt werden kann, jedoch mit zusätzlichen APIs, die das Hinzufügen von `Source`-Objekten ermöglichen. Gemischte Geräte verfügen über interne Mischer, die Audio- oder Bilddateien zusammensetzen und so einen einzigen Audio- und Bildausgangsstream erzeugen. Gemischte Geräte gibt es in Bild- und Audioausführungen.

Begriff	Beschreibung
Konfiguration gemischter Geräte	Ein Konfigurationsobjekt für das gemischte Gerät. Bei gemischten Geräten mit Bildern werden dadurch Eigenschaften wie Abmessungen und Bildrate konfiguri ert. Bei gemischten Geräten mit Audiodateien wird dadurch die Kanalanzahl konfiguriert.
Quelle	Ein Container, der die Position eines visuellen Elements auf dem Bildschirm und die Eigenschaften einer Audiospur im Audiomix definiert. Ein gemischtes Gerät kann mit null oder mehr Quellen konfiguriert werden. Quellen erhalten eine Konfiguration, die sich darauf auswirkt, wie die Medien der Quelle verwendet werden. Das obige Image zeigt vier Bildquellen: • Unten links mit Kameraeingang • Oben rechts mit Filmeingabe • Unten rechts mit dem Amazon IVS-Logo • Ein Vollbild-Hintergrundbild
Konfiguration der Quelle	Ein Konfigurationsobjekt für die Quelle, welche in ein gemischtes Gerät geht. Die vollständigen Konfigurationsobjekte werden unten beschrieben.
Übergang	Um einen Slot an eine neue Position zu verschieben oder einige seiner Eigenschaften zu ändern, verwenden Sie `MixedDevice.transitionToCon figuration()` . Diese Methode verwendet: • Eine neue Quellkonfiguration, die den nächsten Status für die Quelle darstellt. • Eine Dauer, die angibt, wie lange die Animation relativ zur Zeitlinie des Videos dauern soll. Wenn die Dauer auf 0 festgelegt ist, erfolgt der Übergang im nächsten Frame, das gemischt ist. • Eine optionale Rückmeldung, die Sie informiert, wann die Animation abgeschlo ssen ist. Diese Rückmeldung kann nützlich sein, um Animationen zu verketten.

Gemischtes Audio-Gerät

Konfiguration

`MixedAudioDeviceConfiguration` auf Android

`IVSMixedAudioDeviceConfiguration` auf iOS

Name	Typ	Beschreibung
`channels`	Ganzzahl	Anzahl der Ausgangskanäle des Audiomischers. Gültige Werte: 1 und 2. 1 ist Mono-Audio; 2 Stereo-Audio. Standard: 2

Konfiguration der Quelle

`MixedAudioDeviceSourceConfiguration` auf Android

`IVSMixedAudioDeviceSourceConfiguration` auf iOS

Name	Typ	Beschreibung
`gain`	Gleitkommazahl	Audio-Gain. Dies ist ein Multiplikator, daher erhöht jeder Wert über 1 den Gain; jeder Wert unter 1 verringert ihn. Zulässige Werte: 0 bis 2. Standard: 1

Gemischtes Bild-Gerät

Konfiguration

`MixedImageDeviceConfiguration` auf Android

`IVSMixedImageDeviceConfiguration` auf iOS

Name	Typ	Beschreibung
`size`	Vec2	Größe der Video-Bildfläche.
`targetFramerate`	Ganzzahl	Anzahl der Ziel-Frames pro Sekunde für das gemischte Gerät. Im Durchschnitt sollte dieser Wert erreicht werden, aber das System kann unter bestimmten Umständen Frames auslassen (z. B. hohe CPU- oder GPU-Auslastung).

Name	Typ	Beschreibung
`transparencyEnabled`	Boolesch	Dies ermöglicht das Mischen mithilfe der `alpha`-Eigenschaft in Bildquellenkonfigurationen. Wenn Sie diese Einstellung auf `true` einstellen, erhöht sich der Speicher- und CPU-Verbrauch. Standardwert: `false`.

Konfiguration der Quelle

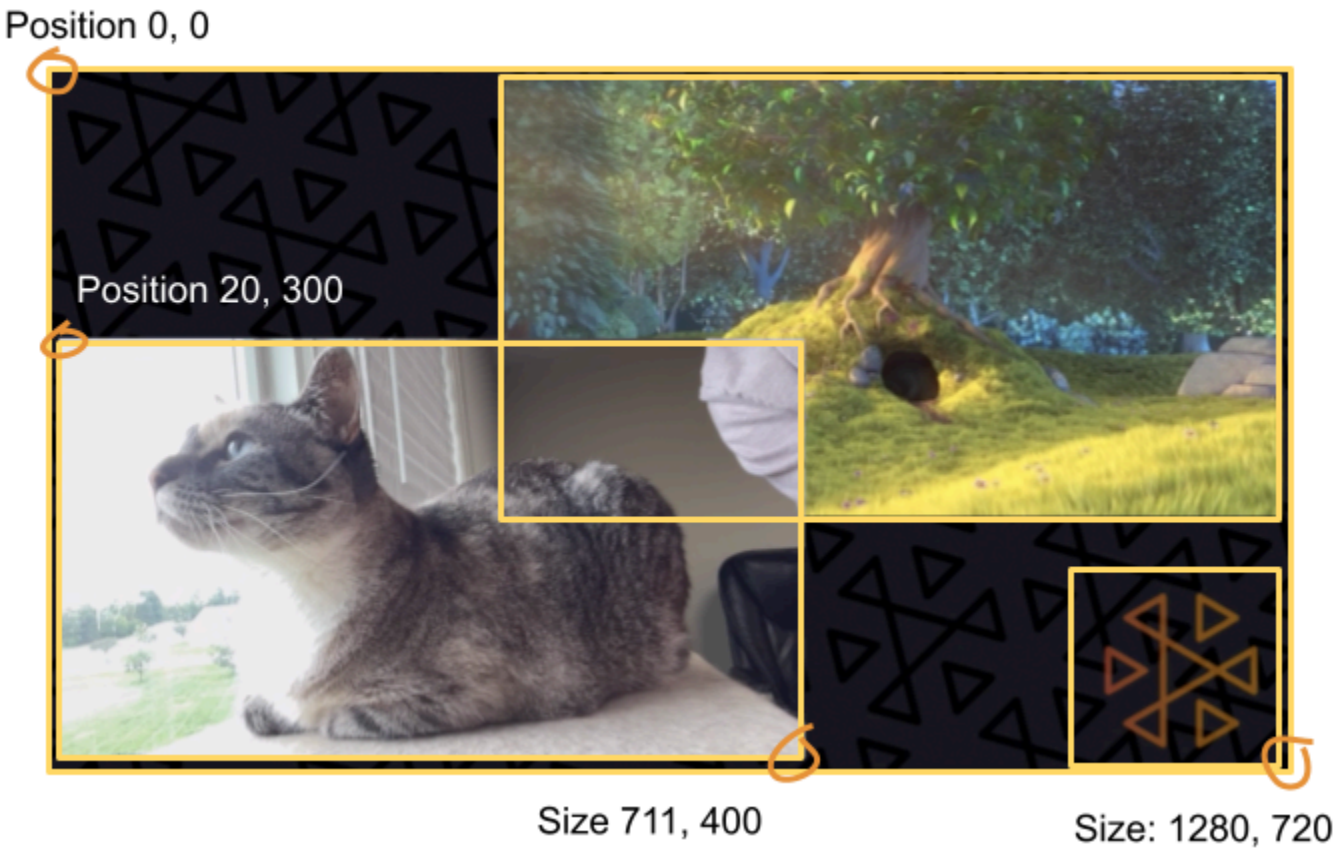
`MixedImageDeviceSourceConfiguration` auf Android

`IVSMixedImageDeviceSourceConfiguration` auf iOS

Name	Typ	Beschreibung
`alpha`	Gleitkommazahl	Alpha des Slots. Dies ist multiplikativ mit allen Alpha-Werten im Image. Gültige Werte: 0 bis 1, wobei 0 vollständig undurchsichtig und 1 vollständig transparent ist. Standard: 1
`aspect`	AspectMode	Aspekt-Ratio-Modus für jedes Image, das im Slot gerendert wird. Zulässige Werte: • `Fill` - Behält das Seitenverhältnis des Images bei, füllt jedoch den Slot aus. Das Bild wird bei Bedarf beschnitten. • `Fit` - Behält das Seitenverhältnis des Images bei, passt es jedoch an den Slot an. Der Slot kann bei Bedarf horizontale oder vertikale Randstreifen haben. Der Brief/Briefkasten befindet sich in der `fillColor` , wenn dieser Wert festgelegt wurde; andernfalls ist er transparent (was schwarz erscheinen kann, wenn die Hintergrundfarbe des Bildes schwarz ist). • `None` - Behält das Seitenverhältnis des Images nicht bei. Das Bild wird so skaliert, dass es den Abmessungen des Slots entspricht.

Name	Typ	Beschreibung
		Standardwert: `Fit`
`fillColor`	Vec4	Füllfarbe, die mit `aspect Fit` verwendet wird, wenn die Seitenverhältnisse von Slot und Bild nicht übereinst immen. Das Format ist (rot, grün, blau, alpha). Gültiger Wert (für jeden Kanal): 0 bis 1. Standard: 0.0.0.0
`position`	Vec2	Position des Slots (in Pixel) relativ zur linken oberen Ecke der Bildfläche. Der Ursprung des Slots ist ebenfalls oben links.
`size`	Vec2	Größe des Slots in Pixel. Ein Festlegen dieses Wertes setzt `matchCanvasSize` außerdem auf `false`. Standard: (0, 0); weil `matchCanvasSize` jedoch standardmäßig `true` ist, ist die gerenderte Slotgröße gleich der Bildflächengröße, nicht (0, 0).
`zIndex`	Gleitkomm azahl	Relative Reihenfolge der Slots. Slots mit höherem `zIndex`-Wert werden über Slots mit niedrigerem `zIndex`-Wert gelegt.

Erstellen und Konfigurieren eines gemischten Bildgerätes



Hier erstellen wir eine Szene, die der am Anfang dieses Handbuchs ähnelt, mit drei Bildschirmelementen:

- Slot unten links für eine Kamera.
- Slot unten rechts für eine Logo-Überlagerung.
- Slot oben rechts für einen Film.

Beachten Sie, dass der Ursprung für die Bildfläche die obere linke Ecke ist und dies für alle Slots gilt. Wenn Sie also einen Slot an (0, 0) positionieren, wird er in die obere linke Ecke gesetzt, wobei der gesamte Slot sichtbar ist.

iOS

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

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

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

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

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

Android

```
val deviceDiscovery = DeviceDiscovery(this /* context */)
val mixedImageConfig = MixedImageDeviceConfiguration().apply {
```

```
    setSize(BroadcastConfiguration.Vec2(1280f, 720f))
    setTargetFramerate(60)
    setEnableTransparency(true)
}
val mixedImageDevice = deviceDiscovery.createMixedImageDevice(mixedImageConfig)

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

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

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

Entfernen von Quellen

Um eine Quelle zu entfernen, rufen Sie `MixedDevice.remove` mit dem `Source`-Objekt auf, das Sie entfernen möchten.

Animationen mit Übergängen

Die Übergangsmethode ersetzt die Konfiguration einesr Quelle durch eine neue Konfiguration. Dieser Ersatz kann zeitlich animiert werden, indem eine Dauer von mehr als 0 Sekunden festgelegt wird.

Welche Eigenschaften können animiert werden?

Nicht alle Eigenschaften in der Slot-Struktur können animiert werden. Alle Eigenschaften, die auf Float-Typen basieren, können animiert werden; andere Eigenschaften werden entweder am Anfang oder am Ende der Animation wirksam.

Name	Kann es animiert werden?	Auswirkungspunkt
`Audio.gain`	Ja	Interpoliert
`Image.alpha`	Ja	Interpoliert
`Image.aspect`	Nein	Ende
`Image.fillColor`	Ja	Interpoliert
`Image.position`	Ja	Interpoliert
`Image.size`	Ja	Interpoliert
`Image.zIndex` Hinweis: `zIndex` verschiebt 2D-Ebenen durch den 3D-Raum, so dass der Übergang stattfindet, wenn sich die beiden Ebenen irgendwann in der Mitte der Animation kreuzen. Dies könnte berechnet werden, hängt aber des `zIndex`-Start- und -Endwerte s ab. Kombinieren Sie dies für einen reibungsloseren Übergang mit `alpha`.	Ja	Unbekannt

Einfache Beispiele:

Im Folgenden finden Sie Beispiele für eine Vollbild-Kameraübernahme unter Verwendung der oben unter Erstellen und Konfigurieren eines gemischten Bildgeräts definierten Konfiguration. Dies wird 0,5 Sekunden lang animiert.

iOS

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

Android

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

Spiegelung der Übertragung

Um ein angeschlossenes Bildgerät in der Übertragung in diese Richtung zu spiegeln ...	Verwenden Sie einen negativen Wert für …
Horizontal	Die Breite des Slots
Vertikal	Die Höhe des Slots

Um ein angeschlossenes Bildgerät in der Übertragung in diese Richtung zu spiegeln ...	Verwenden Sie einen negativen Wert für …
Sowohl horizontal als auch vertikal	Die Slot-Breite und -Höhe

Die Position muss um denselben Wert angepasst werden, damit der Slot beim Spiegeln in die richtige Position gebracht wird.

Im Folgenden finden Sie Beispiele für die horizontale und vertikale Spiegelung der Übertragung.

iOS

Horizontale Spiegelung:

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

Vertikale Spiegelung:

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

Android

Horizontale Spiegelung:

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

Vertikale Spiegelung:

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

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

Hinweis: Diese Spiegelung unterscheidet sich von der `setMirrored`-Methode in `ImagePreviewView` (Android) und `IVSImagePreviewView` (iOS). Diese Methode wirkt sich nur auf die lokale Vorschauansicht auf dem Gerät aus und hat keine Auswirkungen auf die Übertragung.

IVS-Broadcast-SDK: Token-Austausch | Echtzeit-Streaming

Der Token-Austausch ermöglicht es Ihnen, die Funktionen für Teilnehmer-Tokens zu aktualisieren oder herabzustufen und Token-Attribute innerhalb des Mobile Broadcast SDK zu aktualisieren, ohne dass die Teilnehmer erneut eine Verbindung herstellen müssen. Dies ist nützlich für Szenarien wie Co-Hosting, bei denen Teilnehmer zunächst nur über Subscribe-Funktionen verfügen und später Veröffentlichungsfunktionen benötigen.

Einschränkungen:

- Der Token-Austausch funktioniert nur mit Token, die auf Ihrem Server mit einem Schlüsselpaar erstellt wurden. Er funktioniert nicht mit Token, die über die CreateParticipantToken-API erstellt wurden.
- Wenn Sie den Token-Austausch verwenden, um Attribute zu ändern, die serverseitige Zusammensetzungslayouts steuern (wie featuredParticipantAttribute und participantOrderAttribute), wird das Layout einer aktiven Zusammensetzung erst aktualisiert, wenn der Teilnehmer wieder eine Verbindung herstellt.

Teilnehmer-Token

Der Token-Austausch ist unkompliziert: Rufen Sie die `exchangeToken`-API für das `Stage`/`IVSStage`-Objekt auf und stellen Sie das neue Token bereit. Wenn sich die `capabilities` des neuen Tokens von denen des vorherigen Tokens unterscheiden, werden die Fähigkeiten des neuen Tokens sofort bewertet. Wenn das vorherige Token beispielsweise nicht über die Fähigkeit `publish` verfügte, das neue Token jedoch schon, werden die Stage-Strategy-Funktionen für die Veröffentlichung aufgerufen, sodass die Hostanwendung entscheiden kann, ob sie mit der neuen Funktion sofort veröffentlichen oder warten möchte. Das Gleiche gilt für entfernte Funktionen: Wenn das vorherige Token über die Fähigkeit `publish` verfügte und das neue Token nicht, macht der

Teilnehmer die Veröffentlichung sofort rückgängig, ohne die Funktionen der Phasenstrategie für die Veröffentlichung aufzurufen.

Beim Token-Austausch müssen das vorherige und das neue Token dieselben Werte für die folgenden Nutzdaten-Felder haben:

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

Diese Felder sind unveränderlich. Der Austausch eines Tokens, das ein unveränderliches Feld ändert, führt dazu, dass das SDK den Austausch sofort ablehnt.

Die übrigen Felder können geändert werden, darunter:

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

iOS

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

Android

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

```
stage.exchangeToken(newToken)
```

Empfangen von Updates

Eine Funktion in `StageRenderer`/`IVSStageRenderer` empfängt Updates über bereits veröffentlichte Remote-Teilnehmer, die ihre Token austauschen, um ihre `userId` oder `attributes` zu aktualisieren. Fernteilnehmer, die noch nicht veröffentlichen, erhalten ihre `userId` und `attributes` aktualisiert und über die vorhandenen `onParticipantJoined`/`participantDidJoin`-Renderer-Funktionen angezeigt, wenn sie schließlich veröffentlichen.

iOS

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

Android

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

Sichtbarkeitsstatus

Wenn ein Teilnehmer ein Token austauscht, um seine `userId` oder `attributes` zu aktualisieren, hängt die Sichtbarkeit dieser Änderungen von seinem aktuellen Veröffentlichungsstatus ab:

- Wenn der Teilnehmer nicht veröffentlicht: Die Aktualisierung wird im Hintergrund verarbeitet. Wenn er irgendwann veröffentlicht, erhalten alle SDKs die aktualisierte `userId` und `attributes` als Teil der ersten Veröffentlichung.

- Falls der Teilnehmer bereits veröffentlicht: Das Update wird sofort übertragen. Allerdings erhalten nur mobile SDKs v1.37.0+ die Benachrichtigung. Teilnehmer mit dem Web-SDK, älteren mobilen SDKs und der serverseitigen Zusammensetzung sehen die Änderung erst, wenn der Teilnehmer die Veröffentlichung rückgängig macht und sie erneut veröffentlicht.

In dieser Tabelle wird die Unterstützungsmatrix verdeutlicht:

Teilnehmerstatus	Beobachter: Mobile SDK 1.37.0+	Beobachter: Ältere mobile SDKs, Web-SDK, serversei tige Zusammensetzung
Wird nicht veröffentlicht (startet dann)	# Sichtbar (bei Veröffent lichung über eine Veranstal tung, der ein Teilnehmer beigetreten ist)	# Sichtbar (bei Veröffent lichung über eine Veranstal tung, der ein Teilnehmer beigetreten ist)
Bereits veröffentlicht (wird nie erneut veröffentlicht)	# Sichtbar (sofort über ein durch Metadaten der Teilnehmer aktualisiertes Ereignis)	# Nicht sichtbar
Bereits veröffentlicht (Veröffen tlichung rückgängig gemacht und erneut veröffentlicht)	# Sichtbar (sofort über ein durch Metadaten der Teilnehmer aktualisiertes Ereignis)	## Letztendlich sichtbar (bei erneuten Veröffentlichung über eine Veranstaltung, der ein Teilnehmer beigetreten ist)

IVS-Broadcast-SDK: Benutzerdefinierte Image-Quellen | Echtzeit-Streaming

Benutzerdefinierte Bildeingabequellen ermöglichen es einer Anwendung, eine eigene Bildeingabe für das Broadcast-SDK bereitzustellen, anstatt sich auf die voreingestellten Kameras oder die Bildschirmfreigabe zu beschränken. Eine benutzerdefinierte Bildquelle kann so einfach sein wie ein halbtransparentes Wasserzeichen oder eine statische „Bin gleich zurück“-Szene, oder es kann der Anwendung ermöglichen, zusätzliche benutzerdefinierte Verarbeitungen wie das Hinzufügen von Schönheitsfiltern zur Kamera durchzuführen.

Wenn Sie eine benutzerdefinierte Image-Eingangsquelle zur benutzerdefinierten Steuerung der Kamera verwenden (z. B. die Verwendung von Schönheitsfilter-Bibliotheken, die Kamerazugriff erfordern), ist das Broadcast-SDK nicht mehr für die Verwaltung der Kamera verantwortlich. Stattdessen ist die Anwendung dafür verantwortlich, den Lebenszyklus der Kamera korrekt zu handhaben. Lesen Sie die offizielle Plattformdokumentation darüber, wie Ihre Anwendung die Kamera verwalten soll.

Android

Erstellen Sie nach dem Erstellen einer `DeviceDiscovery`-Sitzung eine Bildeingabequelle:

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

Diese Methode gibt ein `CustomImageSource` zurück, welche eine Image-Quelle ist, die von einem Standard-Android-Surface unterstützt wird. Die Unterklasse `SurfaceSource` kann in der Größe geändert und gedreht werden. Sie können auch ein `ImagePreviewView` erstellen, um eine Vorschau seines Inhalts anzuzeigen.

So rufen Sie das zugrundeliegende ab `Surface`:

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

Dieses `Surface` kann als Ausgabepuffer für Image-Produzenten wie Camera2, OpenGL ES und andere Bibliotheken verwendet werden. Der einfachste Anwendungsfall ist das direkte Zeichnen einer statischen Bitmap oder Farbe in den Canvas des Surface. Viele Bibliotheken (wie Schönheitsfilter-Bibliotheken) bieten jedoch eine Methode, mit der eine Anwendung ein externes `Surface` zum Rendern angeben kann. Sie können eine solche Methode verwenden, um dieses `Surface` an die Filterbibliothek zu übergeben, die es der Bibliothek ermöglicht, verarbeitete Frames für das Streamen der Broadcast-Sitzung auszugeben.

Dieses `CustomImageSource` kann in einen `LocalStageStream` verpackt und von der `StageStrategy` zurückgegeben werden, um es in einer `Stage` zu veröffentlichen.

iOS

Erstellen Sie nach dem Erstellen einer `DeviceDiscovery`-Sitzung eine Bildeingabequelle:

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

Diese Methode gibt eine `IVSCustomImageSource` zurück, welche eine Image-Quelle ist, die es der Anwendung ermöglicht, `CMSampleBuffers` manuell abzusenden. Informationen zu unterstützten Pixelformaten finden Sie in der iOS-Broadcast-SDK-Referenz; ein Link zur aktuellsten Version befindet sich in den Versionshinweisen zu Amazon IVS für die neueste Broadcast-SDK-Version.

Die an die benutzerdefinierte Quelle gesendeten Proben werden auf die Stage gestreamt:

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

Verwenden Sie diese Methode zum Streamen von Videos in einem Rückruf. Wenn Sie beispielsweise die Kamera verwenden, kann die Anwendung jedes Mal, wenn ein neuer Beispielpuffer von einer `AVCaptureSession` erhalten wird, den Beispielpuffer an die benutzerdefinierte Image-Quelle weiterleiten. Falls gewünscht, kann die Anwendung eine weitere Verarbeitung (wie einen Schönheitsfilter) anwenden, bevor sie das Beispiel an die benutzerdefinierte Image-Quelle absendet.

Dieses `IVSCustomImageSource` kann in einen `IVSLocalStageStream` verpackt und von der `IVSStageStrategy` zurückgegeben werden, um es in einer `Stage` zu veröffentlichen.

IVS-Broadcast-SDK: Benutzerdefinierte Audio-Quellen | Echtzeit-Streaming

Hinweis: Diese Anleitung führt Sie durch die Schritte zum Einstieg in das Android Broadcast-SDK für IVS-Streaming. Informationen für die iOS- und Web-SDKs werden zukünftig veröffentlicht.

Benutzerdefinierte Audioeingangsquellen ermöglichen es einer Anwendung, ihre eigenen Audioeingänge an das Broadcast-SDK zu liefern, anstatt auf das integrierte Mikrofon des Geräts beschränkt zu sein. Eine benutzerdefinierte Audioquelle ermöglicht es Anwendungen, verarbeitetes Audio mit Effekten zu streamen, mehrere Audiostreams zu mischen oder in Audioverarbeitungsbibliotheken von Drittanbietern zu integrieren.

Beachten Sie, dass das Broadcast-SDK nicht mehr für die Verwaltung der Kamera verantwortlich ist, wenn Sie eine benutzerdefinierte Bildeingabequelle zur benutzerdefinierten Steuerung der Kamera verwenden. Stattdessen ist Ihre Anwendung für die Erfassung, Verarbeitung und Übertragung von Audiodaten an die benutzerdefinierte Quelle verantwortlich.

Der Arbeitsablauf für benutzerdefinierte Audioquellen folgt diesen Schritten:

1. Audioeingang – Erstellen Sie eine benutzerdefinierte Audioquelle mit einem bestimmten Audioformat (Samplerate, Kanäle, Format).

2. Ihre Verarbeitung – Erfassen oder generieren Sie Audiodaten aus Ihrer Audioverarbeitungspipeline.
3. Benutzerdefinierte Audioquelle – Senden Sie Audiopuffer an die benutzerdefinierte Quelle mithilfe von`appendBuffer()`.
4. Stage – In `LocalStageStream` abschließen und über Ihre `StageStrategy` in der Stage veröffentlichen.
5. Teilnehmer – Die Teilnehmer der Phase erhalten das bearbeitete Audio in Echtzeit.

Android

Erstellen einer benutzerdefinierten Audioquelle

Erstellen Sie nach dem Erstellen einer `DeviceDiscovery`-Sitzung eine Audioeingabequelle:

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

// Create custom audio source with specific format
CustomAudioSource customAudioSource = deviceDiscovery.createAudioInputSource(
   2,  // Number of channels (1 = mono, 2 = stereo)
   BroadcastConfiguration.AudioSampleRate.RATE_48000,  // Sample rate
   AudioDevice.Format.INT16  // Audio format (16-bit PCM)
);
```

Diese Methode gibt eine `CustomAudioSource` zurück, die unformatierte PCM-Audiodaten akzeptiert. Die benutzerdefinierte Audioquelle muss mit demselben Audioformat konfiguriert sein, das Ihre Audioverarbeitungspipeline erzeugt.

Unterstützte Formate

Parameter	Optionen	Beschreibung
Kanäle	1 (Mono), 2 (Stereo)	Die Anzahl der Audiokanäle.
Abtastrate	RATE_16000, RATE_44100, RATE_48000	Audio-Abtastrate in Hz. Für hohe Qualität werden 48 kHz empfohlen.
Format	INT16, FLOAT32	Format der Hörprobe. INT16 ist 16-Bit-Festkomma-PCM, FLOAT32 ist 32-Bit-Gl

Parameter	Optionen	Beschreibung
		eitkomma-PCM. Es sind sowohl verschachtelte als auch planare Formate verfügbar.

Audioeinstellungen

Verwenden Sie die Methode `appendBuffer()`, um Audiodaten an die benutzerdefinierte Quelle zu senden:

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

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

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

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

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

Wichtige Überlegungen:

- Audiodaten müssen in dem Format vorliegen, das bei der Erstellung der benutzerdefinierten Quelle angegeben wurde.
- Die Zeitstempel sollten monoton ansteigend sein und von Ihrer Audioquelle bereitgestellt werden, um eine reibungslose Audiowiedergabe zu gewährleisten.

- Reichen Sie regelmäßig Audiodateien ein, um Lücken im Stream zu vermeiden.
- Die Methode gibt die Anzahl der verarbeiteten Samples zurück (0 bedeutet Fehler).

In einer Stage veröffentlichen

Die `CustomAudioSource` in einem `AudioLocalStageStream` abschließen und von Ihrer `StageStrategy` zurücksenden:

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

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

   @Override
   public boolean shouldPublishFromParticipant(
         @NonNull Stage stage,
         @NonNull ParticipantInfo participantInfo) {
      return true;  // Control when to publish
   }

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

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

Vollständiges Beispiel: Integration der Audioverarbeitung

Hier ist ein vollständiges Beispiel, das die Integration mit einem Audioverarbeitungs-SDK zeigt:

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

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

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

      // Initialize IVS components
      initializeIVSStage();

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

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

      // Create custom audio source (48kHz stereo, 16-bit)
      customAudioSource = deviceDiscovery.createAudioInputSource(
         2,  // Stereo
         BroadcastConfiguration.AudioSampleRate.RATE_48000,
         AudioDevice.Format.INT16
      );

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

      // Create stage with strategy
      Strategy strategy = new Strategy() {
         @NonNull
         @Override
         public List<LocalStageStream> stageStreamsToPublishForParticipant(
               @NonNull Stage stage,
```

```
            @NonNull ParticipantInfo participantInfo) {
        return Collections.singletonList(audioStream);
      }

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

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

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

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

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

private void submitAudioToStage(byte[] audioData, long timestamp) {
    try {
      // Allocate direct buffer
```

```
            ByteBuffer buffer = ByteBuffer.allocateDirect(audioData.length);
            buffer.put(audioData);

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

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

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

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

Bewährte Methoden

Audioformatkonsistenz

Stellen Sie sicher, dass das von Ihnen eingereichte Audioformat dem Format entspricht, das bei der Erstellung der benutzerdefinierten Quelle angegeben wurde:

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

// Your audio data must be:
// - 2 channels (stereo)
```

```
// - 48000 Hz sample rate
// - 16-bit interleaved PCM format
```

Pufferverwaltung

Verwenden Sie direkte `ByteBuffers` und verwenden Sie sie erneut, um die Garbage Collection zu minimieren:

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

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

Timing und Synchronisation

Für eine reibungslose Audiowiedergabe müssen Sie die von Ihrer Audioquelle bereitgestellten Zeitstempel verwenden. Wenn Ihre Audioquelle keinen eigenen Zeitstempel hat, erstellen Sie Ihren eigenen Epochenzeitstempel und berechnen Sie die Dauer zwischen den einzelnen Samples manuell anhand der Anzahl der Frames und der Framegröße.

```
// "audioFrameTimestamp" should be generated by your audio source
// Consult your audio source’s documentation for information on how to get this
long timestamp = audioFrameTimestamp;
```

Fehlerbehandlung

Überprüfen Sie immer den Rückgabewert von `appendBuffer()`:

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

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

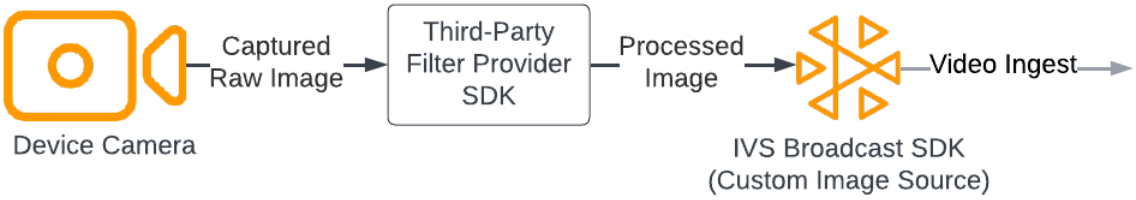
IVS Broadcast SDK: Kamerafilter von Drittanbietern | Echtzeit-Streaming

In diesem Handbuch wird davon ausgegangen, dass Sie bereits mit benutzerdefinierten Bild-Quellen vertraut sind und das Broadcast-SDK zu IVS-Echtzeit-Streaming in Ihre Anwendung integrieren.

Kamerafilter ermöglichen es Livestream-Erstellern, ihr Gesichts- oder Hintergrundbild zu verbessern oder zu ändern. Dies kann möglicherweise das Engagement der Zuschauer steigern, Zuschauer anlocken und das Live-Streaming-Erlebnis verbessern.

Integration von Kamerafiltern von Drittanbietern

Sie können Kamera-Filter-SDKs von Drittanbietern in das IVS-Broadcast-SDK integrieren, indem Sie die Ausgabe des Filter-SDKs einer benutzerdefinierten Bildeingabequelle zuführen. Mit einer benutzerdefinierten Bildeingabequelle kann eine Anwendung ihre eigene Bildereingabe für das Broadcast SDK bereitstellen. Das SDK eines Drittanbieters von Filtern kann möglicherweise den Lebenszyklus der Kamera verwalten, um Bilder von der Kamera zu verarbeiten, einen Filtereffekt anzuwenden und in einem Format auszugeben, das an eine benutzerdefinierte Bildquelle übergeben werden kann.



Informationen zu integrierten Methoden zum Konvertieren eines Kamerarahmens mit angewendetem Filtereffekt in ein Format, das an eine benutzerdefinierte Bildeingabequelle übergeben werden kann, finden Sie in der Dokumentation Ihres Drittanbieters von Filtern. Der Prozess variiert, je nachdem, welche Version des IVS Broadcast SDK verwendet wird:

- Web – Der Filteranbieter muss in der Lage sein, seine Ausgabe auf einem Bildflächenelement zu rendern. Anschließend kann die Methode captureStream verwendet werden, um einen MediaStream des Bildflächeninhalts zurückzugeben. Der MediaStream kann dann in eine Instance eines LocalStageStream umgewandelt und auf einer Stage veröffentlicht werden.
- Android – Das SDK des Filteranbieters kann entweder einen Frame auf einem vom IVS-Broadcast-SDK bereitgestellten Android `Surface` rendern oder den Frame in eine Bitmap konvertieren. Wenn Sie eine Bitmap verwenden, kann diese dann durch Entsperren und Schreiben in eine Bildfläche

in der zugrunde liegenden `Surface` gerendert werden, das von der benutzerdefinierten Bildquelle bereitgestellt wird.

- iOS – Das SDK eines Drittanbieter für Filter muss einen Kamerarahmen mit einem als `CMSampleBuffer` angewendeten Filtereffekt bereitstellen. Informationen dazu, wie Sie nach der Verarbeitung eines Kamerabilds ein `CMSampleBuffer` als endgültige Ausgabe erhalten, finden Sie in der SDK-Dokumentation Ihres Drittanbieters für Filter.

Verwendung von BytePlus mit dem IVS Broadcast SDK

In diesem Dokument wird erklärt, wie Sie das BytePlus Effects SDK mit dem IVS Broadcast SDK verwenden.

Android

Installieren und Einrichten des BytePlus-Effects-SDK

Einzelheiten zur Installation, Initialisierung und Einrichtung des BytePlus-Effects-SDK finden Sie im Android-Zugriffsleitfaden von BytePlus.

Einrichten der benutzerdefinierten Bildquelle

Führen Sie nach der Initialisierung des SDK verarbeitete Kamerarahmen mit einem Filtereffekt ein, der auf eine benutzerdefinierte Bildeingabequelle angewendet wird. Erstellen Sie dazu eine Instance eines `DeviceDiscovery`-Objekts sowie eine benutzerdefinierte Bildquelle. Beachten Sie, dass das Broadcast-SDK nicht mehr für die Verwaltung der Kamera verantwortlich ist, wenn Sie eine benutzerdefinierte Bildeingabequelle zur benutzerdefinierten Steuerung der Kamera verwenden. Stattdessen ist die Anwendung dafür verantwortlich, den Lebenszyklus der Kamera korrekt zu handhaben.

Java

```
var deviceDiscovery = DeviceDiscovery(applicationContext)
var customSource = deviceDiscovery.createImageInputSource( BroadcastConfiguration.Vec2(
720F, 1280F
))
var surface: Surface = customSource.inputSurface
var filterStream = ImageLocalStageStream(customSource)
```

Konvertieren der Ausgabe in ein Bitmap und Zuführen zu einer benutzerdefinierten Bildeingabequelle

Damit Kamerarahmen, auf die ein vom BytePlus-Effects-SDK angewendeter Filtereffekt angewendet wurde, direkt an das IVS-Broadcast-SDK weitergeleitet werden können, konvertieren Sie die Texturausgabe des BytePlus-Effects-SDK in eine Bitmap. Bei der Verarbeitung eines Bildes wird die `onDrawFrame()`-Methode vom SDK aufgerufen. Die `onDrawFrame()`-Methode ist eine öffentliche Methode der GLSurfaceView.Renderer-Schnittstelle von Android. In der von BytePlus bereitgestellten Android-Beispielanwendung wird diese Methode bei jedem Kamerarahmen aufgerufen. Diese gibt eine Textur aus. Gleichzeitig können Sie die `onDrawFrame()`-Methode mit Logik ergänzen, um diese Textur in eine Bitmap umzuwandeln und sie einer benutzerdefinierten Bildeingabequelle zuzuführen. Verwenden Sie für diese Konvertierung, wie im folgenden Codebeispiel gezeigt, die vom BytePlus-SDK bereitgestellte `transferTextureToBitmap`-Methode. Diese Methode wird von der com.bytedance.labcv.core.util.ImageUtil-Bibliothek aus dem BytePlus Effects SDK bereitgestellt, wie im folgenden Codebeispiel gezeigt. Anschließend können Sie auf das zugrunde liegende Android `Surface` des `CustomImageSource` rendern, indem Sie die resultierende Bitmap auf die Bildfläche einer Oberfläche schreiben. Viele aufeinanderfolgende Aufrufe von `onDrawFrame()` führen zu einer Folge von Bitmaps und erzeugen bei der Kombination einen Videostream.

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);
```

Verwendung von DeepAR mit dem IVS Broadcast SDK

In diesem Dokument wird erklärt, wie Sie das DeepAR SDK mit dem IVS Broadcast SDK verwenden.

Android

Einzelheiten zur Integration des DeepAR-SDK in das Android-IVS-Broadcast-SDK finden Sie im Android-Integrationshandbuch von DeepAR.

iOS

Einzelheiten zur Integration des DeepAR-SDK in das iOS IVS-Broadcast-SDK finden Sie im iOS-Integrationshandbuch von DeepAR.

Verwendung von Snap mit dem IVS Broadcast SDK

In diesem Dokument wird erklärt, wie Sie das Camera Kit SDK von Snap mit dem IVS Broadcast SDK verwenden.

Web

In diesem Abschnitt wird davon ausgegangen, dass Sie bereits mit dem Veröffentlichen und Abonnieren von Videos mithilfe des Web-Broadcast-SDK vertraut sind.

Um das Camera-Kit-SDK von Snap mit dem Web-Broadcast-SDK des IVS-Echtzeit-Streaming zu integrieren, müssen Sie Folgendes tun:

1. Installieren Sie das Camera-Kit-SDK und das Webpack. (In unserem Beispiel wird Webpack als Bundler verwendet, Sie können jedoch jeden Bundler Ihrer Wahl verwenden.)
2. Erstellen von `index.html`.
3. Fügen Sie Einrichtungselemente hinzu.
4. Erstellen von `index.css`.
5. Zeigen Sie Teilnehmer an und legen Sie sie fest.
6. Zeigen Sie die angeschlossenen Kameras und Mikrofone an.
7. Erstellen Sie eine Camera-Kit-Sitzung.
8. Rufen Sie Objektive ab und füllen Sie die Objektivauswahl.
9. Rendern Sie die Ausgabe einer Camera-Kit-Sitzung in einer Bildfläche.
10. Erstellen Sie eine Funktion, um das Dropdown-Menü „Objektiv“ zu füllen.

11. Stellen Sie Camera Kit eine Medienquelle zum Rendern und Veröffentlichen eines `LocalStageStream` zur Verfügung.
12. Erstellen von `package.json`.
13. Erstellen Sie eine Webpack-Konfigurationsdatei.
14. Richten Sie einen HTTPS-Server ein und testen Sie ihn.

Jeder dieser Schritte wird nachfolgend beschrieben.

Installieren des Camera-Kit-SDK und des Webpacks

In diesem Beispiel verwenden wir Webpack als Bundler, Sie können jedoch auch einen beliebigen anderen Bundler nutzen.

```
npm i @snap/camera-kit webpack webpack-cli
```

index.html erstellen

Erstellen Sie als Nächstes das HTML-Boilerplate und importieren Sie das Web-Broadcast-SDK als Skript-Tag. Stellen Sie im folgenden Code sicher, dass Sie `<SDK version>` durch die von Ihnen verwendete Broadcast-SDK-Version ersetzen.

HTML

```
<!--
/*! Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. SPDX-License-
Identifier: Apache-2.0 */
-->
<!DOCTYPE html>
<html lang="en">

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

  <title>Amazon IVS Real-Time Streaming Web Sample (HTML and JavaScript)</title>

  <!-- Fonts and Styling -->
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?
family=Roboto:300,300italic,700,700italic" />
```

```
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/normalize/8.0.1/
normalize.css" />
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/milligram/1.4.1/
milligram.css" />
  <link rel="stylesheet" href="./index.css" />

  <!-- Stages in Broadcast SDK -->
  <script src="https://web-broadcast.live-video.net/<SDK version>/amazon-ivs-web-
broadcast.js"></script>
</head>

<body>
  <!-- Introduction -->
  <header>
    <h1>Amazon IVS Real-Time Streaming Web Sample (HTML and JavaScript)</h1>

    <p>This sample is used to demonstrate basic HTML / JS usage. <b><a href="https://
docs.aws.amazon.com/ivs/latest/LowLatencyUserGuide/multiple-hosts.html">Use the AWS
 CLI</a></b> to create a <b>Stage</b> and a corresponding <b>ParticipantToken</b>.
 Multiple participants can load this page and put in their own tokens. You can <b><a
 href="https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-guides/stages#glossary"
 target="_blank">read more about stages in our public docs.</a></b></p>
  </header>
  <hr />

  <!-- Setup Controls -->

  <!-- Display Local Participants -->

  <!-- Lens Selector -->

  <!-- Display Remote Participants -->

  <!-- Load All Desired Scripts -->
```

Einrichtungselemente hinzufügen

Erstellen Sie den HTML-Code für die Auswahl einer Kamera, eines Mikrofons und eines Objektivs sowie für die Angabe eines Teilnehmer-Tokens:

HTML

```
<!-- Setup Controls -->
```

```
  <div class="row">
    <div class="column">
      <label for="video-devices">Select Camera</label>
      <select disabled id="video-devices">
        <option selected disabled>Choose Option</option>
      </select>
    </div>
    <div class="column">
      <label for="audio-devices">Select Microphone</label>
      <select disabled id="audio-devices">
        <option selected disabled>Choose Option</option>
      </select>
    </div>
    <div class="column">
      <label for="token">Participant Token</label>
      <input type="text" id="token" name="token" />
    </div>
    <div class="column" style="display: flex; margin-top: 1.5rem">
      <button class="button" style="margin: auto; width: 100%" id="join-button">Join
 Stage</button>
    </div>
    <div class="column" style="display: flex; margin-top: 1.5rem">
      <button class="button" style="margin: auto; width: 100%" id="leave-button">Leave
 Stage</button>
    </div>
  </div>
```

Fügen Sie darunter zusätzlichen HTML-Code hinzu, um Kamera-Feeds von lokalen und entfernten Teilnehmern anzuzeigen:

HTML

```
 <!-- Local Participant -->
<div class="row local-container">
    <canvas id="canvas"></canvas>

    <div class="column" id="local-media"></div>
    <div class="static-controls hidden" id="local-controls">
      <button class="button" id="mic-control">Mute Mic</button>
      <button class="button" id="camera-control">Mute Camera</button>
    </div>
  </div>
```

```
  <hr style="margin-top: 5rem"/>

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

Laden Sie zusätzliche Logik, einschließlich Hilfsmethoden zum Einrichten der Kamera und der gebündelten JavaScript-Datei. (Später in diesem Abschnitt werden Sie diese JavaScript-Dateien erstellen und sie in einer einzigen Datei bündeln, damit Sie Camera Kit als Modul importieren können. Die gebündelte JavaScript-Datei enthält die Logik, mit der Sie Camera Kit festlegen können, ein Objektiv anwenden und den Kamera-Feed mit einem Objektiv in einer Stage veröffentlichen). Fügen Sie schließende Tags für die Elemente `body` und `html` hinzu, um die Erstellung von `index.html` abzuschließen.

HTML

```
<!-- Load all Desired Scripts -->
  <script src="./helpers.js"></script>
  <script src="./media-devices.js"></script>
  <!-- <script type="module" src="./stages-simple.js"></script> -->
  <script src="./dist/bundle.js"></script>
</body>
</html>
```

Erstellen einer Datei vom Typ „index.css“

Erstellen Sie eine CSS-Quelldatei, um die Seite zu formatieren. Wir erläutern diesen Code nicht näher, sondern konzentrieren uns auf die Logik für die Verwaltung einer Stage und die Integration in das Camera-Kit-SDK von Snap.

CSS

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

html,
body {
  margin: 2rem;
  box-sizing: border-box;
```

```
  height: 100vh;
  max-height: 100vh;
  display: flex;
  flex-direction: column;
}

hr {
  margin: 1rem 0;
}

table {
  display: table;
}

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

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

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

.content {
  flex: 1 0 auto;
}

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

.local-container {
  position: relative;
}
```

```
.static-controls {
  position: absolute;
  margin-left: auto;
  margin-right: auto;
  left: 0;
  right: 0;
  bottom: -4rem;
  text-align: center;
}

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

.hidden {
  display: none;
}

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

video {
  border: 0.5rem solid #555;
  border-radius: 0.5rem;
}
.placeholder {
  background-color: #333333;
  display: flex;
  text-align: center;
  margin-bottom: 1rem;
}
.placeholder span {
  margin: auto;
  color: white;
}
#local-media {
  display: inline-block;
  width: 100vw;
}
```

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

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

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

Teilnehmer anzeigen und einrichten

Erstellen Sie als Nächstes `helpers.js`, das Hilfsmethoden zum Anzeigen und Einrichten von Teilnehmern enthält:

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;
}
```

Angeschlossene Kameras und Mikrofone anzeigen

Erstellen Sie als Nächstes `media-devices.js`, das Hilfsmethoden zum Anzeigen von Kameras und Mikrofonen enthält, die mit Ihrem Gerät verbunden sind:

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,
    },
  });
}
```

Erstellen einer Camera-Kit-Sitzung

Erstellen Sie `stages.js`, das die Logik zum Anwenden eines Objektivs auf den Kamera-Feed und zum Veröffentlichen des Feeds in einer Stage enthält. Es wird empfohlen, den folgenden Codeblock zu kopieren und in `stages.js` einzufügen. Sie können den Code dann Stück für Stück überprüfen, um zu verstehen, was in den folgenden Abschnitten geschieht.

JavaScript

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

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

import {
  bootstrapCameraKit,
  createMediaStreamSource,
  Transform2D,
} from '@snap/camera-kit';

let cameraButton = document.getElementById('camera-control');
```

```
let micButton = document.getElementById('mic-control');
let joinButton = document.getElementById('join-button');
let leaveButton = document.getElementById('leave-button');

let controls = document.getElementById('local-controls');
let videoDevicesList = document.getElementById('video-devices');
let audioDevicesList = document.getElementById('audio-devices');

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

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

const liveRenderTarget = document.getElementById('canvas');

const init = async () => {
  await initializeDeviceSelect();

  const cameraKit = await bootstrapCameraKit({
    apiToken: 'INSERT_YOUR_API_TOKEN_HERE',
  });

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

  availableLenses = lenses;
  populateLensSelector(lenses);

  const snapStream = liveRenderTarget.captureStream();

  lensSelector.addEventListener('change', handleLensChange);
  lensSelector.disabled = true;
  cameraButton.addEventListener('click', () => {
    const isMuted = !cameraStageStream.isMuted;
```

```
    cameraStageStream.setMuted(isMuted);
    cameraButton.innerText = isMuted ? 'Show Camera' : 'Hide Camera';
  });

  micButton.addEventListener('click', () => {
    const isMuted = !micStageStream.isMuted;
    micStageStream.setMuted(isMuted);
    micButton.innerText = isMuted ? 'Unmute Mic' : 'Mute Mic';
  });

  joinButton.addEventListener('click', () => {
    joinStage(session, snapStream);
  });

  leaveButton.addEventListener('click', () => {
    leaveStage();
  });
};

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

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

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

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

```
const joinStage = async (session, snapStream) => {
  if (connected || joining) {
    return;
  }
  joining = true;

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

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

  // Retrieve the User Media currently set on the page
  localCamera = await getCamera(videoDevicesList.value);
  localMic = await getMic(audioDevicesList.value);
  await setCameraKitSource(session, localCamera);

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

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

  stage = new Stage(token, strategy);

  // Other available events:
  // https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-guides/stages#events
  stage.on(StageEvents.STAGE_CONNECTION_STATE_CHANGED, (state) => {
    connected = state === ConnectionState.CONNECTED;

    if (connected) {
      joining = false;
```

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

stage.on(StageEvents.STAGE_PARTICIPANT_JOINED, (participant) => {
  console.log('Participant Joined:', participant);
});

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

    let streamsToDisplay = streams;

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

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

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

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

```
  }
};

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

  joining = false;
  connected = false;

  cameraButton.innerText = 'Hide Camera';
  micButton.innerText = 'Mute Mic';
  controls.classList.add('hidden');
};

init();
```

Im ersten Teil dieser Datei wird das Broadcast-SDK und das Camera-Kit-Web-SDK importiert und die Variablen, die mit jedem SDK verwenden werden, initialisiert. Es wird eine Camera-Kit-Sitzung erstellt , durch Aufrufen von `createSession` nach dem Bootstrapping das Camera-Kit-Web-SDK. Beachten Sie, dass ein Elementobjekt einer Bildfläche an eine Sitzung übergeben wird. Dies weist Camera Kit an, in dieser Bildfläche zu rendern.

JavaScript

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

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

import {
  bootstrapCameraKit,
  createMediaStreamSource,
  Transform2D,
} from '@snap/camera-kit';

let cameraButton = document.getElementById('camera-control');
```

```
let micButton = document.getElementById('mic-control');
let joinButton = document.getElementById('join-button');
let leaveButton = document.getElementById('leave-button');

let controls = document.getElementById('local-controls');
let videoDevicesList = document.getElementById('video-devices');
let audioDevicesList = document.getElementById('audio-devices');

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

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

const liveRenderTarget = document.getElementById('canvas');

const init = async () => {
  await initializeDeviceSelect();

  const cameraKit = await bootstrapCameraKit({
    apiToken: 'INSERT_YOUR_API_TOKEN_HERE',
  });

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

Abrufen von Objektiven und Auffüllen der Objektivauswahl

Um Ihre Objektive abzurufen, ersetzen Sie den Platzhalter für die Objektivgruppen-ID durch Ihre eigene ID, die Sie im Camera-Kit-Entwicklerportal finden. Füllen Sie das Dropdown-Menü für die Objektivauswahl mithilfe der Funktion `populateLensSelector()` aus, die wir später erstellen.

JavaScript

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

```
  ]);

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

Rendern der Ausgabe einer Camera-Kit-Sitzung in einer Bildfläche

Verwenden Sie die Methode captureStream, um einen `MediaStream` des Bildflächeninhalts zurückzugeben. Die Bildfläche enthält einen Videostream des Kamera-Feeds mit angewendetem Objektiv. Fügen Sie außerdem Ereignis-Listener für Schaltflächen zum Stummschalten von Kamera und Mikrofon sowie Ereignis-Listener für das Betreten und Verlassen einer Stage hinzu. Im Ereignis-Listener für den Beitritt zu einer Stage übergeben wir eine Camera-Kit-Sitzung und das `MediaStream` aus der Bildfläche, damit es in einer Stage veröffentlicht werden kann.

JavaScript

```
const snapStream = liveRenderTarget.captureStream();

  lensSelector.addEventListener('change', handleLensChange);
  lensSelector.disabled = true;
  cameraButton.addEventListener('click', () => {
    const isMuted = !cameraStageStream.isMuted;
    cameraStageStream.setMuted(isMuted);
    cameraButton.innerText = isMuted ? 'Show Camera' : 'Hide Camera';
  });

  micButton.addEventListener('click', () => {
    const isMuted = !micStageStream.isMuted;
    micStageStream.setMuted(isMuted);
    micButton.innerText = isMuted ? 'Unmute Mic' : 'Mute Mic';
  });

  joinButton.addEventListener('click', () => {
    joinStage(session, snapStream);
  });

  leaveButton.addEventListener('click', () => {
    leaveStage();
  });
};
```

Erstellen einer Funktion zum Auffüllen des Dropdown-Menüs „Objektiv“

Erstellen Sie die folgende Funktion, um die Objektivauswahl mit den zuvor abgerufenen Objektiven zu füllen. Die Objektivauswahl ist ein Benutzeroberflächenelement auf der Seite, mit dem Sie aus einer Liste von Objektiven auswählen können, die auf den Kamera-Feed angewendet werden sollen. Erstellen Sie außerdem die Rückruffunktion `handleLensChange`, um das angegebene Objektiv anzuwenden, wenn es im Dropdown-Menü Objektiv ausgewählt wird.

JavaScript

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

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

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

Stellen Sie dem Camera Kit eine Medienquelle zum Rendern zur Verfügung und veröffentlichen Sie einen LocalStageStream

Um einen Videostream mit einem angewandten Objektiv zu veröffentlichen, rufen Sie eine Funktion mit dem Namen `setCameraKitSource` auf, um das zuvor von der Bildfläche erfasste `MediaStream` zu übergeben. Der `MediaStream` von der Bildfläche tut im Moment nichts, weil wir unseren lokalen Kamera-Feed noch nicht integriert haben. Wir können unseren lokalen Kamera-Feed integrieren, indem wir die Hilfsmethode `getCamera` aufrufen und sie `localCamera` zuweisen. Wir können dann unseren lokalen Kamera-Feed (über `localCamera`) und das Sitzungsobjekt an `setCameraKitSource` übergeben. Die `setCameraKitSource`-Funktion konvertiert unseren lokalen Kamera-Feed in eine Medienquelle für CameraKit, indem sie `createMediaStreamSource` aufruft. Die Medienquelle für `CameraKit` wird dann umgewandelt, um die nach vorne gerichtete

Kamera zu spiegeln. Der Objektiveffekt wird dann auf die Medienquelle angewendet und durch Aufrufen von `session.play()` auf die Ausgabe-Bildfläche gerendert.

Nachdem das Objektiv nun auf das von der Bildfläche erfasste `MediaStream` angewendet wurde, können wir es in einer Stage veröffentlichen. Wir machen das, indem wir einen `LocalStageStream` mit den Videospuren des `MediaStream` erstellen. Eine Instance von `LocalStageStream` kann dann zur Veröffentlichung an eine `StageStrategy` übergeben werden.

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;
    },
  };
```

Der verbleibende Code unten dient der Erstellung und Verwaltung der Stage:

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();
```

Erstellen einer Datei vom Typ „package.json“

Erstellen Sie `package.json` und fügen Sie die folgende JSON-Konfiguration hinzu. Diese Datei definiert die Abhängigkeiten und enthält einen Skriptbefehl zum Bündeln des Codes.

JSON-Konfiguration

```
{
  "dependencies": {
    "@snap/camera-kit": "^0.10.0"
  },
  "name": "ivs-stages-with-snap-camerakit",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "build": "webpack"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "devDependencies": {
    "webpack": "^5.95.0",
    "webpack-cli": "^5.1.4"
  }
}
```

Erstellen einer Webpack-Konfigurationsdatei

Erstellen Sie `webpack.config.js` und fügen Sie den folgenden Code hinzu. Dadurch wird der bisher erstellte Code gebündelt, sodass wir die Importanweisung zur Verwendung von Camera Kit verwenden können.

JavaScript

```
const path = require('path');
module.exports = {
  entry: ['./stage.js'],
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
};
```

Führen Sie schließlich `npm run build` aus, um Ihr JavaScript nach den Vorgaben in der Webpack-Konfigurationsdatei zu bündeln. Zu Testzwecken können Sie dann HTML und JavaScript über den lokalen Computer bereitstellen. In diesem Beispiel verwenden wir das Python-Modul `http.server`.

Einrichten eines HTTPS-Servers und Ausführen von Tests

Zum Testen des Codes müssen Sie einen HTTPS-Server einrichten. Wenn Sie einen HTTPS-Server für die lokale Entwicklung verwenden und die Integration Ihrer Web-App mit dem Camera-Kit-SDK von Snap testen, können Sie CORS-Probleme (Cross-Origin Resource Sharing) vermeiden.

Öffnen Sie ein Terminal und navigieren Sie zu dem Verzeichnis, in dem Sie den gesamten Code bis zu diesem Zeitpunkt erstellt haben. Führen Sie den folgenden Befehl aus, um ein selbstsigniertes SSL-/TLS-Zertifikat und einen privaten Schlüssel zu generieren:

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

Dadurch werden zwei Dateien erstellt: `key.pem` (der private Schlüssel) und `cert.pem` (das selbstsignierte Zertifikat). Erstellen Sie eine neue Python-Datei mit dem Namen `https_server.py` und fügen Sie den folgenden Code hinzu:

Python

```
import http.server
import ssl

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

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

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

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

Öffnen Sie ein Terminal, navigieren Sie zu dem Verzeichnis, in dem Sie die Datei `https_server.py` erstellt haben, und führen Sie den folgenden Befehl aus:

```
python3 https_server.py
```

Dadurch wird der HTTPS-Server unter „https://localhost:4443“ gestartet und Dateien werden aus dem aktuellen Verzeichnis bereitgestellt. Stellen Sie sicher, dass sich die Dateien `cert.pem` und `key.pem` im selben Verzeichnis wie die Datei `https_server.py` befinden.

Öffnen Sie Ihren Browser und navigieren Sie zu „https://localhost:4443“. Da es sich um ein selbstsigniertes SSL-/TLS-Zertifikat handelt, wird es von Ihrem Webbrowser nicht als vertrauenswürdig eingestuft, sodass Sie eine Warnung erhalten. Sie können die Warnung umgehen, da dieser Schritt nur zu Testzwecken dient. Sie sollten dann sehen, wie der AR-Effekt für das zuvor angegebene Snap-Objektiv auf Ihren Kamera-Feed auf dem Bildschirm angewendet wird.

Beachten Sie, dass dieses Setup, das die integrierten Python-Module `http.server` und `ssl` verwendet, für lokale Entwicklungs- und Testzwecke geeignet ist, für eine Produktionsumgebung jedoch nicht empfohlen wird. Das in diesem Setup verwendete selbstsignierte SSL-/TLS-Zertifikat wird von Webbrowsern und anderen Clients nicht als vertrauenswürdig eingestuft. Das bedeutet, dass Benutzern beim Zugriff auf den Server Sicherheitswarnungen angezeigt werden. In diesem Beispiel verwenden Sie zwar die integrierten Python-Module „http.server“ und „ssl“, Sie können aber auch eine andere HTTPS-Serverlösung nutzen.

Android

Um das Camera-Kit-SDK von Snap mit dem IVS-Android-Broadcast-SDK zu integrieren, müssen Sie das Camera-Kit-SDK installieren, eine Camera-Kit-Sitzung initialisieren, ein Objektiv anwenden und die Ausgabe der Camera-Kit-Sitzung an die benutzerdefinierte Bildeingabequelle weiterleiten.

Um das Camera Kit SDK zu installieren, fügen Sie Folgendes zur Datei Ihres `build.gradle`-Moduls hinzu. Ersetzen Sie `$cameraKitVersion` durch die neueste Version des Camera-Kit-SDK.

Java

```
implementation "com.snap.camerakit:camerakit:$cameraKitVersion"
```

Initialisieren und beziehen Sie eine `cameraKitSession`. Camera Kit bietet auch einen praktischen Wrapper für die APIs des CameraX von Android, sodass Sie keine komplizierte Logik schreiben müssen, um CameraX mit Camera Kit zu verwenden. Sie können das `CameraXImageProcessorSource`-Objekt als Quelle für ImageProcessor verwenden, wodurch Sie Streaming-Frames für die Kameravorschau starten können.

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();
    }
```

Objektive abrufen und anwenden

Sie können Objektive und ihre Reihenfolge im Karussell im Entwickler-Portal des Camera Kit konfigurieren:

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

Senden Sie zur Übertragung verarbeitete Bilder an die zugrunde liegende `Surface` einer benutzerdefinierten Bildquelle. Verwenden Sie ein `DeviceDiscovery`-Objekt und erstellen Sie eine `CustomImageSource`, um ein `SurfaceSource` zurückzugeben. Anschließend können Sie die Ausgabe einer `CameraKit`-Sitzung in der `Surface` rendern die von der `SurfaceSource` bereitgestellt wird.

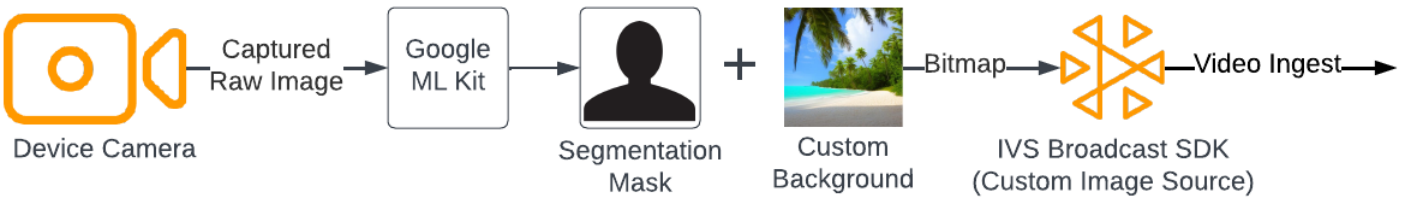
Java

```
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
```

Verwenden von Ersetzen des Hintergrunds mit dem IVS Broadcast SDK

Beim Ersetzen des Hintergrunds handelt es sich um eine Art Kamerafilter, mit dem Livestream-Ersteller ihren Hintergrund ändern können. Wie im folgenden Diagramm dargestellt, umfasst das Ersetzen Ihres Hintergrunds Folgendes:

1. Abrufen eines Kamerabildes aus dem Live-Kamera-Feed.
2. Segmentierung in Vordergrund- und Hintergrundkomponenten mithilfe des Google-ML-Sets.
3. Kombinieren der resultierenden Segmentierungsmaske mit einem benutzerdefinierten Hintergrundbild.
4. Weitergabe an eine benutzerdefinierte Bildquelle zur Übertragung.



Web

In diesem Abschnitt wird davon ausgegangen, dass Sie bereits mit dem Veröffentlichen und Abonnieren von Videos mithilfe des Web-Broadcast-SDK vertraut sind.

Um den Hintergrund eines Live-Streams durch ein benutzerdefiniertes Bild zu ersetzen, verwenden Sie das Selfie-Segmentierungsmodell mit MediaPipe Image Segmenter. Hierbei handelt es sich um ein Machine Learning-Modell, das identifiziert, welche Pixel im Video-Frame im Vorder- oder Hintergrund liegen. Anschließend können Sie die Ergebnisse des Modells verwenden, um den Hintergrund eines Live-Streams zu ersetzen, indem Sie Vordergrundpixel aus dem Video-Feed in ein benutzerdefiniertes Bild kopieren, das den neuen Hintergrund darstellt.

Um die Ersetzung des Hintergrunds in das Web-Broadcast-SDK für IVS-Echtzeit-Streaming zu integrieren, müssen Sie Folgendes tun:

1. Installieren Sie MediaPipe und Webpack. (In unserem Beispiel wird Webpack als Bundler verwendet, Sie können jedoch jeden Bundler Ihrer Wahl verwenden.)
2. Erstellen von `index.html`.
3. Fügen Sie Medienelemente hinzu.
4. Fügen Sie ein Skript-Tag hinzu.
5. Erstellen von `app.js`.
6. Lädt ein benutzerdefiniertes Hintergrundbild.
7. Erstellen Sie eine Instance von `ImageSegmenter`.
8. Rendern Sie den Video-Feed in eine Bildfläche.
9. Erstellen Sie eine Logik zum Ersetzen des Hintergrunds.
10. Erstellen Sie eine Webpack-Konfigurationsdatei.
11. Bündeln Sie Ihre JavaScript-Datei.

MediaPipe und Webpack installieren

Installieren Sie zunächst die npm-Pakete `@mediapipe/tasks-vision` und `webpack`. Das folgende Beispiel verwendet Webpack als JavaScript-Bundler. Sie können bei Bedarf einen anderen Bundler verwenden.

JavaScript

```
npm i @mediapipe/tasks-vision webpack webpack-cli
```

Stellen Sie sicher, dass Sie auch Ihr `package.json` aktualisieren, um `webpack` als Entwickler-Skript anzugeben:

JavaScript

```
"scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "build": "webpack"
  },
```

index.html erstellen

Erstellen Sie als Nächstes das HTML-Boilerplate und importieren Sie das Web-Broadcast-SDK als Skript-Tag. Stellen Sie im folgenden Code sicher, dass Sie `<SDK version>` durch die von Ihnen verwendete Broadcast-SDK-Version ersetzen.

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>
```

Medienelemente hinzufügen

Fügen Sie als Nächstes ein Videoelement und zwei Bildflächen-Elemente innerhalb des Tags des Hauptteils hinzu. Das Videoelement enthält Ihren Live-Kamera-Feed und wird als Eingabe für den MediaPipe Image Segmenter verwendet. Das erste Bildflächenelement wird verwendet, um eine Vorschau des zu übertragenden Feeds zu rendern. Das zweite Bildflächenelement wird zum Rendern des benutzerdefinierten Bildes verwendet, das als Hintergrund verwendet wird. Da die zweite Bildfläche mit dem benutzerdefinierten Bild nur als Quelle zum programmgesteuerten Kopieren von Pixeln von dort auf die endgültige Bildfläche verwendet wird, ist sie nicht sichtbar.

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>
```

Hinzufügen eines Skript-Tags

Fügen Sie ein Skript-Tag hinzu, um eine gebündelte JavaScript-Datei zu laden, die den Code für die Ersetzung des Hintergrunds enthält, und veröffentlichen Sie sie in einer Stage:

```
<script src="./dist/bundle.js"></script>
```

Erstellen von app.js

Erstellen Sie als Nächstes eine JavaScript-Datei, um die Elementobjekte für die Bildfläche und Videoelemente abzurufen, die auf der HTML-Seite erstellt wurden. Importieren Sie die Module

`ImageSegmenter` und `FilesetResolver`. Das Modul `ImageSegmenter` wird zur Durchführung der Segmentierungsaufgabe verwendet.

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";
```

Erstellen Sie als Nächstes eine Funktion mit dem Namen `init()`, um den MediaStream von der Kamera des Benutzers abzurufen und jedes Mal eine Rückruffunktion aufzurufen, wenn ein Kamerarahmen vollständig geladen ist. Fügen Sie Ereignis-Listener für die Schaltflächen zum Beitreten und Verlassen einer Stage hinzu.

Beachten Sie, dass wir beim Beitritt zu einer Stage eine Variable mit dem Namen `segmentationStream` übergeben. Hierbei handelt es sich um einen Video-Stream, der von einem Bildflächenelement erfasst wird und ein Vordergrundbild enthält, das dem benutzerdefinierten Bild, das den Hintergrund darstellt, überlagert ist. Später wird dieser benutzerdefinierte Stream zum Erstellen einer Instance eines `LocalStageStream` verwendet, die in einer Stage veröffentlicht werden kann.

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();
  });
};
```

Ein benutzerdefiniertes Hintergrundbild laden

Fügen Sie unten in der `init`-Funktion Code hinzu, um eine Funktion mit dem Namen `initBackgroundCanvas` aufzurufen, die ein benutzerdefiniertes Bild aus einer lokalen Datei lädt und es in einer Bildfläche rendert. Diese Funktion wird im nächsten Schritt definiert. Ordnen Sie das von der Kamera des Benutzers abgerufene `MediaStream` dem Videoobjekt zu. Später wird dieses Videoobjekt an den Image Segmenter übergeben. Legen Sie außerdem eine Funktion mit dem Namen `renderVideoToCanvas` als Rückruffunktion fest, die immer dann aufgerufen wird, wenn ein Videobild vollständig geladen wurde. Diese Funktion wird in einem späteren Schritt definiert.

JavaScript

```
initBackgroundCanvas();

  video.srcObject = localCamera;
  video.addEventListener("loadeddata", renderVideoToCanvas);
```

Wir implementieren die `initBackgroundCanvas`-Funktion, die ein Bild aus einer lokalen Datei lädt. In diesem Beispiel wird das Bild von einem Strandes als benutzerdefinierter Hintergrund verwendet. Die Bildfläche mit dem benutzerdefinierten Bild wird nicht angezeigt, da Sie sie mit den Vordergrundpixeln aus dem Bildflächenelement mit dem Kamera-Feed zusammenführen.

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);
  };
};
```

Erstellen einer Instance von ImageSegmenter

Erstellen Sie als Nächstes eine Instance von `ImageSegmenter`, die das Bild segmentiert und das Ergebnis als Maske zurückgibt. Verwenden Sie beim Erstellen einer Instance eines `ImageSegmenter` das Selfie-Segmentierungsmodell.

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,
  });
};
```

Rendern des Video-Feeds auf eine Bildfläche

Erstellen Sie als Nächstes die Funktion, die den Video-Feed auf das andere Bildflächenelement rendert. Das Video-Feed muss auf einer Bildfläche gerendert werden, damit mithilfe der Bildflächen-2D-API die Vordergrundpixel daraus extrahiert werden können. Dabei übergeben wir auch einen Videoframe an unsere `ImageSegmenter`-Instance und verwenden dabei die Methode segmentforVideo, um den Vordergrund vom Hintergrund im Videoframe zu segmentieren. Wenn die Methode segmentforVideo zurückgegeben wird, ruft sie die benutzerdefinierte Rückruffunktion `replaceBackground` auf, um den Hintergrund zu ersetzen.

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);
};
```

Logik zum Ersetzen des Hintergrunds erstellen

Erstellen Sie die `replaceBackground`-Funktion, die das benutzerdefinierte Hintergrundbild mit dem Vordergrund aus dem Kamera-Feed zusammenführt, um den Hintergrund zu ersetzen. Die Funktion ruft zunächst die zugrunde liegenden Pixeldaten des benutzerdefinierten Hintergrundbilds und des Video-Feeds von den beiden zuvor erstellten Bildflächenelementen ab. Anschließend durchläuft es die von `ImageSegmenter` bereitgestellte Maske, die angibt, welche Pixel im Vordergrund stehen. Beim Durchlaufen der Maske kopiert es selektiv Pixel, die den Kamera-Feed des Benutzers enthalten, in die entsprechenden Hintergrund-Pixeldaten. Sobald das erledigt ist, konvertiert es die endgültigen Pixeldaten, wobei der Vordergrund in den Hintergrund kopiert wird, und auf eine Bildfläche dargestellt wird.

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);
}
```

Als Referenz finden Sie hier die vollständige `app.js`-Datei, die die gesamte obige Logik enthält:

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();
```

Erstellen einer Webpack-Konfigurationsdatei

Fügen Sie diese Konfiguration zu Ihrer Webpack-Konfigurationsdatei hinzu, um `app.js` zu bündeln, damit die Importaufrufe funktionieren:

JavaScript

```
const path = require("path");
module.exports = {
  entry: ["./app.js"],
  output: {
    filename: "bundle.js",
    path: path.resolve(__dirname, "dist"),
  },
```

```
};
```

Bündeln Ihrer JavaScript-Dateien

```
npm run build
```

Starten Sie einen einfachen HTTP-Server aus dem Verzeichnis, das `index.html` enthält, und öffnen Sie `localhost:8000`, um das Ergebnis anzuzeigen:

```
python3 -m http.server -d ./
```

Android

Um den Hintergrund in Ihrem Livestream zu ersetzen, können Sie die Selfie-Segmentierungs-API von Google ML Kit verwenden. Die Selfie-Segmentierungs-API akzeptiert ein Kamerabild als Eingabe und gibt eine Maske zurück. Diese liefert für jedes Pixel des Bildes einen Konfidenzwert, der angibt, ob es sich im Vordergrund oder im Hintergrund befand. Basierend auf dem Konfidenzwert können Sie dann die entsprechende Pixelfarbe entweder aus dem Hintergrundbild oder dem Vordergrundbild abrufen. Dieser Vorgang wird fortgesetzt, bis alle Konfidenzwerte in der Maske überprüft wurden. Das Ergebnis ist ein neues Array von Pixelfarben, das Vordergrundpixel kombiniert mit Pixeln aus dem Hintergrundbild enthält.

Um die Ersetzung im Hintergrund mit dem Android-Broadcast-SDK für ISV-Echtzeit-Streaming zu integrieren, müssen Sie wie folgt vorgehen:

1. Installieren Sie die CameraX-Bibliotheken und das Google ML-Kit.
2. Initialisieren Sie Boilerplate-Variablen.
3. Erstellen Sie eine benutzerdefinierte Bildquelle.
4. Verwalten Sie Kamerarahmen.
5. Übergeben Sie Kamerarahmen an Google ML Kit.
6. Überlagern Sie den Vordergrund des Kamerarahmens mit Ihrem benutzerdefinierten Hintergrund.
7. Führen Sie das neue Bild einer benutzerdefinierten Bildquelle zu.

CameraX-Bibliotheken und Google ML Kit installieren

Verwenden Sie die CameraX-Bibliothek von Android, um Bilder aus dem Live-Kamera-Feed zu extrahieren. Um die CameraX-Bibliothek und das Google ML-Kit zu installieren, fügen Sie

Folgendes zur `build.gradle`-Datei Ihres Moduls hinzu. Ersetzen Sie `${camerax_version}` und `${google_ml_kit_version}` durch die neueste Version der CameraX- bzw. Google-ML-Kit-Bibliotheken.

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}"
```

Importieren Sie die folgenden Bibliotheken:

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
```

Initialisieren von Boilerplate-Variablen

Initialisieren Sie eine Instance von `ImageAnalysis` und eine Instance eines `ExecutorService`:

Java

```
private lateinit var binding: ActivityMainBinding
private lateinit var cameraExecutor: ExecutorService
private var analysisUseCase: ImageAnalysis? = null
```

Initialisieren Sie eine Segmenter-Instance im STREAM_MODE:

Java

```
private val options =
        SelfieSegmenterOptions.Builder()
            .setDetectorMode(SelfieSegmenterOptions.STREAM_MODE)
            .build()

private val segmenter = Segmentation.getClient(options)
```

Erstellen einer benutzerdefinierten Image-Quelle

Erstellen Sie in der onCreate-Methode Ihrer Aktivität eine Instance eines DeviceDiscovery-Objekts sowie eine benutzerdefinierte Bildquelle. Die von der benutzerdefinierten Bildquelle bereitgestellte Surface erhält das endgültige Bild, wobei der Vordergrund über einem benutzerdefinierten Hintergrundbild liegt. Anschließend erstellen Sie mithilfe der benutzerdefinierten Bildquelle eine Instance von einem ImageLocalStageStream. Die Instance von einem ImageLocalStageStream (in diesem Beispiel filterStream benannt) kann dann in einer Stage veröffentlicht werden. Anweisungen zum Einrichten einer Stage finden Sie im Handbuch des IVS-Android-Broadcast-SDK. Erstellen Sie abschließend auch einen Thread, der zur Verwaltung der Kamera verwendet wird.

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()
```

Verwalten von Kamerarahmen

Erstellen Sie als Nächstes eine Funktion zum Initialisieren der Kamera. Diese Funktion nutzt die CameraX-Bibliothek, um Bilder aus dem Live-Kamera-Feed zu extrahieren. Zunächst erstellen Sie eine Instance eines ProcessCameraProvider mit dem Namen cameraProviderFuture. Dieses Objekt stellt ein zukünftiges Ergebnis der Beschaffung eines Kameraanbieters dar. Anschließend laden Sie ein Bild aus Ihrem Projekt als Bitmap. In diesem Beispiel wird ein Bild von einem Strand als Hintergrund verwendet, es kann sich aber auch um ein beliebiges Bild handeln.

Anschließend fügen Sie einen Listener zu cameraProviderFuture hinzu. Dieser Listener wird benachrichtigt, wenn die Kamera verfügbar wird oder wenn beim Abrufen eines Kameraanbieters ein Fehler auftritt.

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))
    }
```

Erstellen Sie im Listener `ImageAnalysis.Builder`, um über den Live-Kamera-Feed auf jedes einzelne Bild zuzugreifen. Legen Sie die Gegendruckstrategie auf `STRATEGY_KEEP_ONLY_LATEST` fest. Dadurch wird sichergestellt, dass jeweils nur ein Kamerarahmen zur Verarbeitung geliefert wird. Konvertieren Sie jedes einzelne Kamerarahmen in eine Bitmap, sodass Sie dessen Pixel extrahieren und später mit dem benutzerdefinierten Hintergrundbild kombinieren können.

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())
```

Übergabe von Kamerarahmen an Google ML Kit

Erstellen Sie als Nächstes ein `InputImage` und übergeben Sie es zur Verarbeitung an die Segmenter-Instance. Ein `InputImage` kann aus einem `ImageProxy` erstellt werden, der von der Instance von `ImageAnalysis` bereitgestellt wird. Sobald dem Segmenter ein `InputImage` zur Verfügung gestellt wird, gibt er eine Maske mit Konfidenzwerten zurück, die die Wahrscheinlichkeit angeben, dass sich ein Pixel im Vordergrund oder Hintergrund befindet. Diese Maske bietet auch Breiten- und Höheneigenschaften, mit denen Sie ein neues Array erstellen, das die Hintergrundpixel des zuvor geladenen benutzerdefinierten Hintergrundbilds enthält.

Java

```
if (mediaImage != null) {
        val inputImage =
            InputImage.fromMediaImag


segmenter.process(inputImage)
    .addOnSuccessListener { segmentationMask ->
```

```
        val mask = segmentationMask.buffer
        val maskWidth = segmentationMask.width
        val maskHeight = segmentationMask.height
        val backgroundPixels = IntArray(maskWidth * maskHeight)
        bgBitmap.getPixels(backgroundPixels, 0, maskWidth, 0, 0, maskWidth, maskHeight)
```

Überlagern des Vordergrunds des Kamerarahmens mit Ihrem benutzerdefinierten Hintergrund

Mit der Maske, die die Konfidenzwerte enthält, dem Kamerarahmen als Bitmap und den Farbpixeln aus dem benutzerdefinierten Hintergrundbild haben Sie alles, was Sie brauchen, um den Vordergrund über Ihren benutzerdefinierten Hintergrund zu legen. Die `overlayForeground`-Funktion wird dann mit folgenden Parametern aufgerufen:

Java

```
resultBitmap = overlayForeground(mask, maskWidth, maskHeight, inputBitmap,
 backgroundPixels)
```

Diese Funktion durchläuft die Maske und prüft die Konfidenzwerte, um festzustellen, ob die entsprechende Pixelfarbe aus dem Hintergrundbild oder dem Kamerarahmen abgerufen werden soll. Wenn der Konfidenzwert angibt, dass sich ein Pixel in der Maske höchstwahrscheinlich im Hintergrund befindet, erhält er die entsprechende Pixelfarbe aus dem Hintergrundbild. Andernfalls wird die entsprechende Pixelfarbe vom Kamerarahmen abgerufen, um den Vordergrund zu erstellen. Sobald die Funktion die Iteration durch die Maske abgeschlossen hat, wird unter Verwendung des neuen Arrays von Farbpixeln eine neue Bitmap erstellt und zurückgegeben. Diese neue Bitmap enthält den Vordergrund, der sich mit dem benutzerdefinierten Hintergrund überlagert.

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
        )
    }
```

Das neue Bild einer benutzerdefinierten Bildquelle zuführen

Anschließend können Sie die neue Bitmap in die `Surface` schreiben, das von einer benutzerdefinierten Bildquelle bereitgestellt wird. Dadurch wird es in Ihre Stage übertragen.

Java

```
resultBitmap = overlayForeground(mask, inputBitmap, mutableBitmap, bgBitmap)
canvas = surface.lockCanvas(null);
canvas.drawBitmap(resultBitmap, 0f, 0f, null)
```

Hier ist die vollständige Funktion zum Abrufen der Kamerarahmen, zum Übergeben an Segmenter und zum Überlagern mit dem Hintergrund:

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))
    }
```

IVS-Broadcast-SDK: Mobile Audiomodi | Echtzeit-Streaming

Die Audioqualität ist ein wichtiger Bestandteil jedes Medienerlebnisses im echten Team, und es gibt keine einheitliche Audiokonfiguration, die für jeden Anwendungsfall am besten funktioniert. Um sicherzustellen, dass Ihre Benutzer beim Anhören eines IVS-Echtzeit-Streams über das beste Erlebnis verfügen, bieten unsere mobilen SDKs mehrere voreingestellte Audiokonfigurationen sowie bei Bedarf leistungsstärkere Anpassungen.

Einführung

Die IVS-SDKs für mobile Übertragungen stellen eine `StageAudioManager`-Klasse bereit. Diese Klasse ist als zentraler Ansprechpartner für die Steuerung der zugrunde liegenden Audiomodi auf beiden Plattformen konzipiert. Unter Android steuert dies den AudioManager, einschließlich Audiomodus, Audioquelle, Inhaltstyp, Nutzung und Kommunikationsgeräte. Unter iOS steuert es die Anwendung AVAudioSession und ob VoiceProcessing aktiviert ist.

Wichtig: Interagieren Sie nicht mit `AVAudioSession` oder `AudioManager` direkt, während das IVS-Echtzeit-Broadcast-SDK aktiv ist. Dies kann dazu führen, dass das Audio verloren geht oder Audio vom falschen Gerät aufgenommen oder wiedergegeben wird.

Bevor Sie Ihr erstes `DeviceDiscovery`- oder `Stage`-Objekt erstellen, muss die `StageAudioManager`-Klasse konfiguriert werden.

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
```

Wenn nichts auf `StageAudioManager` festgelegt wird, bevor eine `DeviceDiscovery`- oder `Stage`-Instance initialisiert wird, wird automatisch die Voreinstellung `VideoChat` angewendet.

Voreinstellungen für den Audio-Modus

Das Echtzeit-Broadcast-SDK bietet drei Voreinstellungen, die jeweils auf gängige Anwendungsfälle zugeschnitten sind, wie unten beschrieben. Für jede Voreinstellung werden fünf Hauptkategorien behandelt, die die Voreinstellungen voneinander unterscheiden.

Die Kategorie Lautstärkeregler bezieht sich auf die Art der Lautstärke (Medienlautstärke oder Anruflautstärke), die über die physischen Lautstärkeregler am Gerät verwendet oder geändert wird. Beachten Sie, dass dies Auswirkungen auf die Lautstärke hat, wenn Sie den Audio-Modus wechseln. Beispiel: Angenommen, die Lautstärke des Geräts ist auf den maximalen Wert eingestellt, während Sie die Voreinstellung „Videochat“ verwenden. Das Umschalten auf die Voreinstellung „Nur Abonnieren“ bewirkt eine andere Lautstärke als die des Betriebssystems, was zu einer erheblichen Änderung der Lautstärke auf dem Gerät führen kann.

Video-Chat

Dies ist die Standardvoreinstellung, die für den Fall konzipiert ist, dass das lokale Gerät ein Echtzeitgespräch mit anderen Teilnehmern führen soll.

Bekanntes Problem unter iOS: Wenn Sie diese Voreinstellung verwenden und kein Mikrofon anschließen, wird der Ton über den Ohrhörer und nicht über den Gerätelautsprecher wiedergegeben. Verwenden Sie diese Voreinstellung nur in Kombination mit einem Mikrofon.

Kategorie	Android	iOS
Echounterdrückung	Aktiviert	Aktiviert
Lautstärkenregler	Anruflautstärke	Anruflautstärke
Auswahl des Mikrofons	Je nach Betriebssystem begrenzt. USB-Mikrofone sind möglicher weise nicht verfügbar.	Je nach Betriebssystem begrenzt. USB- und Bluetooth-Mikrofon e sind möglicherweise nicht verfügbar. Bluetooth-Headsets, die sowohl die Ein- als auch die Ausgabe gemeinsam verarbeiten, sollten funktionieren; z. B. AirPods.
Audioausgabe	Jedes Ausgabegerät sollte funktionieren.	Je nach Betriebssystem begrenzt. Kabelgebundene Headsets sind möglicherweise nicht verfügbar.
Audioqualität	Mittel/Niedrig. Es hört sich wie ein Telefonanruf an, nicht wie die Medienwiedergabe.	Mittel/Niedrig. Es hört sich wie ein Telefonanruf an, nicht wie die Medienwiedergabe.

Nur Abonnement

Diese Voreinstellung ist für den Fall konzipiert, dass Sie andere Veröffentlichungsteilnehmer abonnieren, aber nicht selbst veröffentlichen möchten. Der Schwerpunkt liegt auf der Audioqualität und der Unterstützung aller verfügbaren Ausgabegeräte.

Kategorie	Android	iOS
Echounterdrückung	Disabled	Disabled
Lautstärkenregler	Medienlautstärke	Medienlautstärke
Auswahl des Mikrofons	Nicht verfügbar, diese Voreinstellung ist nicht für die Veröffentlichung konzipiert.	Nicht verfügbar, diese Voreinstellung ist nicht für die Veröffentlichung konzipiert.
Audioausgabe	Jedes Ausgabegerät sollte funktionieren.	Jedes Ausgabegerät sollte funktionieren.
Audioqualität	Hoch. Jeder Medientyp sollte klar zu hören sein, auch Musik.	Hoch. Jeder Medientyp sollte klar zu hören sein, auch Musik.

Studio

Diese Voreinstellung ist für qualitativ hochwertige Abonnements bei gleichzeitiger Beibehaltung der Möglichkeit zur Veröffentlichung konzipiert. Es erfordert, dass die Aufnahme- und Wiedergabe-Hardware eine Echounterdrückung ermöglicht. Ein Anwendungsfall hierfür wäre die Verwendung eines USB-Mikrofons und eines kabelgebundenen Headsets. Das SDK sorgt für die höchste Audioqualität und verlässt sich dabei auf die physische Trennung dieser Geräte, um Echos zu vermeiden.

Kategorie	Android	iOS
Echounterdrückung	Die Plattform-Echounterdrückung ist deaktiviert, doch wenn `StageAudioConfiguration.enableEchoCancellation` wahr ist, kann die Software-Echounterdrückung trotzdem erfolgen.	Disabled
Lautstärkenregler	Medienlautstärke in den meisten Fällen. Anruflautstärke, wenn ein	Medienlautstärke

Kategorie	Android	iOS
	Bluetooth-Mikrofon angeschlossen ist.	
Auswahl des Mikrofons	Jedes Mikrofon sollte funktionieren.	Jedes Mikrofon sollte funktionieren.
Audioausgabe	Jedes Ausgabegerät sollte funktionieren.	Jedes Ausgabegerät sollte funktionieren.
Audioqualität	Hoch. Beide Seiten sollten in der Lage sein, Musik zu senden und sie auf der anderen Seite klar zu hören. Wenn ein Bluetooth-Headset angeschlossen ist, nimmt die Audioqualität ab, da der Bluetooth-SCO-Modus aktiviert ist.	Hoch. Beide Seiten sollten in der Lage sein, Musik zu senden und sie auf der anderen Seite klar zu hören. Wenn ein Bluetooth-Headset angeschlossen ist, kann es je nach Headset aufgrund der Aktivierung des Bluetooth SCO-Modus zu einer Verschlechterung der Audioqualität kommen.

Fortschrittliche Anwendungsfälle

Über die Voreinstellungen hinaus ermöglichen sowohl die Echtzeit-Streaming-Broadcast-SDKs für iOS als auch Android die Konfiguration der zugrunde liegenden Audiomodi der Plattform:

- Legen Sie unter Android AudioSource, Usage und ContentType fest.
- Verwenden Sie unter iOS AVAudioSession.Category, AVAudioSession.CategoryOptions, AVAudioSession.Mode und die Möglichkeit, die Sprachverarbeitung während der Veröffentlichung zu aktivieren oder zu deaktivieren.

Hinweis: Bei Verwendung dieser Audio-SDK-Methoden ist es möglich, dass die zugrunde liegende Audiositzung falsch konfiguriert wird. Wenn Sie z. B. die Option `.allowBluetooth` unter iOS in Kombination mit der Kategorie `.playback` verwenden, entsteht eine ungültige Audiokonfiguration und das SDK kann kein Audio aufnehmen oder wiedergeben. Diese Methoden sollten nur verwendet

werden, wenn eine Anwendung spezifische Anforderungen an eine Audiositzung hat, die validiert wurden.

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
```

iOS-Echounterdrückung

Die Echounterdrückung unter iOS kann auch unabhängig über `IVSStageAudioManager` mit der zugehörigen Methode `echoCancellationEnabled` gesteuert werden. Diese Methode steuert, ob die Sprachverarbeitung auf den Ein- und Ausgangsknoten des vom SDK verwendeten zugrunde liegenden `AVAudioEngine` aktiviert ist. Es ist wichtig, die Auswirkungen einer manuellen Änderung dieser Eigenschaft zu verstehen:

- Die `AVAudioEngine`-Eigenschaft wird nur berücksichtigt, wenn das Mikrofon des SDK aktiv ist. Dies ist aufgrund der iOS-Anforderung erforderlich, dass die Sprachverarbeitung auf den Ein- und

Ausgangsknoten gleichzeitig aktiviert sein muss. Normalerweise erfolgt dies mithilfe des Mikrofons, das von `IVSDeviceDiscovery` zurückgegeben wurde, um einen zu veröffentlichenden `IVSLocalStageStream` zu erstellen. Alternativ kann das Mikrofon aktiviert werden, ohne dass es für die Veröffentlichung verwendet wird, indem ein `IVSAudioDeviceStatsCallback` an das Mikrofon selbst angehängt wird. Dieser alternative Ansatz ist nützlich, wenn Sie Echounterdrückung benötigen und statt des Mikrofons des IVS-SDK ein benutzerdefiniertes Mikrofon mit Audioquelle verwenden.

- Zum Aktivieren der `AVAudioEngine`-Eigenschaft ist der Modus `.videoChat` oder `.voiceChat` erforderlich. Das Anfordern eines anderen Modus führt dazu, dass das zugrunde liegende Audio-Framework von iOS das SDK bekämpft, was zu Audioverlust führt.
- Durch die Aktivierung von `AVAudioEngine` wird die Option `.allowBluetooth` automatisch aktiviert.

Das Verhalten kann je nach Gerät und iOS-Version unterschiedlich sein.

Benutzerdefinierte iOS-Audioquellen

Benutzerdefinierte Audioquellen können mit dem SDK mithilfe von `IVSDeviceDiscovery.createAudioSource` verwendet werden. Bei der Verbindung mit einer Bühne verwaltet das IVS-Broadcast-SDK für Echtzeit-Streaming auch dann eine interne `AVAudioEngine`-Instance für die Audiowiedergabe, wenn das Mikrofon des SDK nicht verwendet wird. Daher müssen die an `IVSStageAudioManager` übergebenen Werte mit dem von der benutzerdefinierten Audioquelle bereitgestellten Audio kompatibel sein.

Wenn die benutzerdefinierte Audioquelle, die für die Veröffentlichung verwendet wird, zwar über das Mikrofon aufzeichnet, aber von der Hostanwendung verwaltet wird, funktioniert das oben genannte SDK zur Echounterdrückung nur, wenn das vom SDK verwaltete Mikrofon aktiviert ist. Informationen zur Umgehung dieser Anforderung finden Sie unter iOS-Echounterdrückung.

Veröffentlichen mit Bluetooth unter Android

Das SDK kehrt automatisch zur Voreinstellung `VIDEO_CHAT` unter Android zurück, wenn die folgenden Bedingungen erfüllt sind:

- Die zugewiesene Konfiguration verwendet den Nutzungswert `VOICE_COMMUNICATION` nicht.
- Ein Bluetooth-Mikrofon ist mit dem Gerät verbunden.
- Der lokale Teilnehmer veröffentlicht in einer Stufe.

Hierbei handelt es sich um eine Einschränkung des Android-Betriebssystems hinsichtlich der Verwendung von Bluetooth-Headsets zur Audioaufzeichnung.

Integration mit anderen SDKs

Da sowohl iOS als auch Android nur einen aktiven Audiomodus pro Anwendung unterstützen, kommt es häufig zu Konflikten, wenn Ihre Anwendung mehrere SDKs verwendet, die die Steuerung des Audiomodus erfordern. Wenn Sie auf solche Konflikte stoßen, können Sie einige gängige Lösungsstrategien ausprobieren, die nachfolgend erläutert werden.

Werte im Audiomodus anpassen

Mithilfe der erweiterten Audiokonfigurationsoptionen des IVS-SDK oder der Funktionalität des anderen SDK können Sie die beiden SDKs auf die zugrunde liegenden Werte ausrichten.

Agora

iOS

Wenn Sie unter iOS dem Agora SDK mitteilen, dass die `AVAudioSession` aktiv bleiben soll, wird verhindert, dass es deaktiviert wird, während das Broadcast-SDK für IVS-Echtzeit-Streaming es verwendet.

```
myRtcEngine.SetParameters("{\"che.audio.keep.audiosession\":true}");
```

Android

Vermeiden Sie den Aufruf von `setEnableSpeakerphone` in `RtcEngine` und rufen Sie `enableLocalAudio(false)` während der Veröffentlichung mit dem Broadcast-SDK für IVS-Echtzeit-Streaming auf. Sie können `enableLocalAudio(true)` erneut aufrufen, wenn das IVS-SDK keine Veröffentlichung durchführt.

Amazon EventBridge mit IVS-Echtzeit-Streaming verwenden

Sie können Amazon EventBridge verwenden, um Ihre Amazon Interactive Video Service (IVS) Streams zu überwachen.

Amazon IVS sendet Änderungsereignisse zum Status Ihrer Streams an Amazon EventBridge. Alle bereitgestellten Ereignisse sind gültig. Allerdings werden Ereignisse auf Best-Effort-Basis gesendet, was bedeutet, dass keine Garantie für Folgendes besteht:

- Ereignisse werden übermittelt – Ein festgelegtes Ereignis kann auftreten (z. B. ein Stream startet), aber es ist möglich, dass Amazon IVS kein entsprechendes Ereignis an EventBridge sendet. Amazon IVS versucht, Ereignisse mehrere Stunden vor dem Aufgeben zu liefern.
- Ereignisse, die geliefert werden, kommen in einem bestimmten Zeitrahmen an – Sie können Ereignisse erhalten, die bis zu ein paar Stunden alt sind.
- Ereignisse werden in der richtigen Reihenfolge geliefert – Ereignisse können ungeordnet sein, insbesondere wenn sie innerhalb kurzer Zeit zueinander gesendet werden. Beispielsweise könnte „Teilnehmer nicht veröffentlicht“ vor „Teilnehmer veröffentlicht“ angezeigt werden.

Obwohl es selten ist, dass Ereignisse fehlen, spät oder nicht in richtiger Reihenfolge sind, sollten Sie diese Möglichkeiten berücksichtigen, wenn Sie geschäftskritische Programme schreiben, die von der Reihenfolge oder dem Vorhandensein von Benachrichtigungsereignissen abhängen.

Sie können EventBridge für jedes der folgenden Ereignisse erstellen.

Ereignistyp	Veranstaltung	Gesendet, wenn ...
Status der IVS-Zusammensetzung	Zielfehler	Der Versuch, Daten an ein Ziel auszugeben, ist fehlgeschlagen (z. B. der S3-Bucket wurde nicht gefunden, der Zugriff auf den S3-Bucket wurde verweigert oder der Stream ist bereits für ein RTMP-Ziel vorhanden).
Status der IVS-Zusammensetzung	Zielstart	Die Ausgabe an ein Ziel wurde erfolgreich gestartet.

Ereignistyp	Veranstaltung	Gesendet, wenn ...
Status der IVS-Zusammensetzung	Zielende	Die Ausgabe an ein Ziel ist abgeschlossen.
Status der IVS-Zusammensetzung	Wiederverbindung zum Ziel	Die Ausgabe an ein Ziel wurde unterbrochen. Es wird versucht, die Verbindung wiederherzustellen.
Status der IVS-Zusammensetzung	Beginn der Sitzung	Es wurde eine Sitzung zur Zusammensetzung erstellt. Dieses Ereignis wird ausgelöst, wenn eine Zusammensetzungsprozess-Pipeline erfolgreich initialisiert wurde. Zu diesem Zeitpunkt hat die Zusammensetzungs-Pipeline erfolgreich eine Stufe abonniert, empfängt Medien und kann Videos erstellen.
Status der IVS-Zusammensetzung	Ende der Sitzung	Eine Zusammensetzungs-Sitzung ist abgeschlossen.
Status der IVS-Zusammensetzung	Sitzungsfehler	Eine Zusammensetzungspipeline ist aufgrund des Löschens einer Stufe, des Fehlschlagens eines oder mehrerer Ausgaben oder eines anderen internen Fehlers fehlgeschlagen.
Statusänderung der IVS-Teilnehmeraufzeichnung	Aufzeichnungsstart	Ein Publisher hat eine Verbindung zur Bühne hergestellt und wird gerade in S3 aufgezeichnet.
Statusänderung der IVS-Teilnehmeraufzeichnung	Aufzeichnungsende	Ein Publisher hat die Verbindung zur Bühne getrennt und alle verbleibenden Dateien wurden in S3 geschrieben.

Ereignistyp	Veranstaltung	Gesendet, wenn ...
Statusänderung der IVS-Teilnehmeraufzeichnung	Fehler beim Aufzeichnen	Ein Publisher stellt eine Verbindung zur Stufe her, doch die Aufzeichnung kann aufgrund von Fehlern nicht gestartet werden (z. B. wenn ein S3-Bucket nicht gefunden wird oder darauf nicht zugegriffen werden kann). Der Live-Stream dieses Publishers wird nicht aufgezeichnet.
Statusänderung der IVS-Teilnehmeraufzeichnung	Fehler bei Aufnahmeende	Die Aufzeichnung wird aufgrund von Fehlern, die während der Aufzeichnung auftreten (z. B. wenn ein S3-Bucket nicht gefunden wird oder darauf nicht zugegriffen werden kann), mit einem Fehler beendet. Einige Objekte können weiterhin in den konfigurierten Speicherort geschrieben werden.
IVS-Bühne-Aktualisierung	Teilnehmer Published	Ein Teilnehmer beginnt, zu einer Bühne zu veröffentlichen.
IVS-Bühne-Aktualisierung	Teilnehmer Unveröffentlicht	Ein Teilnehmer hat aufgehört, zu einer Bühne zu veröffentlichen.
IVS-Bühne-Aktualisierung	Fehler bei Teilnehmerveröffentlichung	Der Versuch eines Teilnehmers, auf einer Bühne zu veröffentlichen, ist fehlgeschlagen.
IVS-Bühne-Aktualisierung	Start der Teilnehmerreplikation	Eine Teilnehmerreplikation startet.

Ereignistyp	Veranstaltung	Gesendet, wenn ...
IVS-Bühne-Aktualisierung	Ende der Teilnehmerreplikation	Eine Teilnehmerreplikation endet. Eine Replikation kann aufgrund des API-Vorgangs StopParticipantReplication enden, wenn der Publisher die Veröffentlichung beendet hat oder wenn er die Veröffentlichung beendet hat und das Zeitfenster für die Wiederverbindung abgelaufen ist.
IVS-Bühne-Aktualisierung	Token ausgetauscht	Ein vorhandenes Teilnehmer-Token wird gegen ein neues ausgetauscht. Dieser Austausch führt zu verbesserten oder herabgestuften Token-Funktionen und/oder aktualisierten Token-Attributen.

Erstellen von Amazon EventBridge Regeln für Amazon IVS

Sie können eine Regel erstellen, die bei einem von Amazon IVS ausgegebenen Ereignis ausgelöst wird. Folgen Sie den Schritten in Erstellen Sie eine Regel in Amazon EventBridge im Benutzerhandbuch für Amazon EventBridge. Wählen Sie bei der Auswahl eines Services Interactive Video Service (IVS).

Beispiele: Status der Zusammensetzung

Zielfehler: Dieses Ereignis wird übermittelt, wenn beim Versuch, Daten an ein Ziel auszugeben, ein Fehler aufgetreten ist (z. B. der S3-Bucket wurde nicht gefunden, der Zugriff auf den S3-Bucket wurde verweigert oder der Stream für ein RTMP-Ziel ist bereits vorhanden).

```
{
   "version": "0",
   "id": "01234567-0123-0123-0123-012345678901",
   "detail-type": "IVS Composition State Change",
   "source": "aws.ivs",
   "account": "aws_account_id",
```

```
  "time": "2017-06-12T10:23:43Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:ivs:us-east-1:aws_account_id:composition/123456789012"
  ],
  "detail": {
    "event_name": "Destination Failure",
    "stage_arn": "<stage-arn>",
    "id": "<Destination-id>",
    "error_code": "e.g., AccessDeniedException",
    "reason": "e.g., Access denied to S3 bucket. Please verify your bucket policy"
  }
}
```

In der folgenden Tabelle sind die `error_code`- und `reason`-Werte für Ereignisse im Zusammenhang mit Zielfehlern sowie Hinweise zur Fehlerbehebung aufgeführt:

error_code	Grund	Anleitung zur Fehlerbehebung
ResourceNotFoundException	S3-Bucket nicht gefunden. Stellen Sie sicher, dass Ihr Bucket vorhanden ist.	Stellen Sie sicher, dass Ihr S3-Bucket vorhanden ist und sich in der richtigen Region befindet.
AccessDeniedException	Zugriff auf den S3-Bucket verweigert. Überprüfen Sie Ihre Bucket-Richtlinie.	Stellen Sie sicher, dass Ihre S3-Bucket-Richtlinie dem IVS-Service die erforderlichen Berechtigungen gewährt.
ConflictException	Stream ist bereits vorhanden	Stellen Sie sicher, dass im gleichen RTMP-Zielkanal keine andere Übertragung aktiv ist.
InternalServerException	Interner Service-Fehler	Wiederholen Sie den Vorgang. Wenn das Problem weiterhin besteht,

error_code	Grund	Anleitung zur Fehlerbehebung
		wenden Sie sich an den AWS Support.

Zielstart: Dieses Ereignis wird gesendet, wenn die Ausgabe an ein Ziel erfolgreich gestartet wurde.

```
{
   "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>",
   }
}
```

Zielende: Dieses Ereignis wird gesendet, wenn die Ausgabe an ein Ziel abgeschlossen ist.

```
{
   "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>",
```

```
    }
}
```

Wiederverbindung zum Ziel: Dieses Ereignis wird gesendet, wenn die Ausgabe an ein Ziel unterbrochen wurde und versucht wird, die Verbindung wiederherzustellen.

```
{
   "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>",
   }
}
```

Sitzungsstart: Dieses Ereignis wird gesendet, wenn eine Zusammensetzungssitzung erstellt wurde. Dieses Ereignis wird ausgelöst, wenn eine Zusammensetzungsprozess-Pipeline erfolgreich initialisiert wurde. Zu diesem Zeitpunkt hat die Zusammensetzungs-Pipeline erfolgreich eine Stufe abonniert, empfängt Medien und kann Videos erstellen.

```
{
   "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>"
   }
}
```

Sitzungsende: Dieses Ereignis wird gesendet, wenn eine Zusammensetzungssitzung abgeschlossen ist und alle Ressourcen gelöscht wurden.

```
{
   "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>"
   }
}
```

Sitzungsfehler: Dieses Ereignis wird übermittelt, wenn eine Zusammensetzungspipeline aufgrund des Löschens einer Stufe, des Fehlschlagens eines oder mehrerer Ausgänge oder eines anderen internen Fehlers fehlgeschlagen ist.

```
{
   "version": "0",
   "id": "01234567-0123-0123-0123-012345678901",
   "detail-type": "IVS Composition State Change",
   "source": "aws.ivs",
   "account": "aws_account_id",
   "time": "2017-06-12T10:23:43Z",
   "region": "us-east-1",
   "resources": [
     "arn:aws:ivs:us-east-1:aws_account_id:composition/123456789012"
   ],
   "detail": {
     "event_name": "Session Failure",
     "stage_arn": "<stage-arn>",
```

```
    "error_code": "e.g., DestinationFailure",
    "reason": "e.g. One or more outputs failed"
  }
}
```

In der folgenden Tabelle sind die `error_code`- und `reason`-Werte für Ereignisse im Zusammenhang mit Sitzungsfehlern sowie Hinweise zur Fehlerbehebung aufgeführt:

error_code	Grund	Anleitung zur Fehlerbehebung
StageDeleted	Stufe wurde gelöscht	Stellen Sie vor Beginn der Zusammensetzung sicher, dass die Stufe vorhanden ist.
DestinationFailure	Eine oder mehrere Ausgaben sind fehlgeschlagen	Überprüfen Sie die Fehler der einzelnen Ziele.
InternalServerException	Interner Service-Fehler	Wiederholen Sie den Vorgang. Wenn das Problem weiterhin besteht, wenden Sie sich an den AWS Support.

Beispiele: Statusänderung der Aufzeichnung einzelner Teilnehmer

Aufzeichnungsstart: Dieses Ereignis wird gesendet, wenn ein Publisher eine Verbindung zur Bühne hergestellt hat und in S3 aufgezeichnet wird.

```
{
  "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"
   }
}
```

Aufzeichnungsende: Dieses Ereignis wird gesendet, wenn ein Publisher die Verbindung zur Bühne getrennt hat und alle verbleibenden Dateien in S3 geschrieben wurden.

```
{
   "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
   }
}
```

Fehler beim Aufzeichnungsstart: Dieses Ereignis wird übermittelt, wenn ein Publisher eine Verbindung zur Stufe herstellt, die Aufzeichnung jedoch aufgrund von Fehlern nicht gestartet werden kann (z. B. wenn ein S3-Bucket nicht gefunden wird oder darauf nicht zugegriffen werden kann). Der Livestream des Publishers wird nicht aufgezeichnet.

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

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

In der folgenden Tabelle sind die `error_code`- und `reason`-Werte für Ereignisse für Fehler beim Aufzeichnungsstart sowie Hinweise zur Fehlerbehebung aufgeführt:

error_code	Grund	Anleitung zur Fehlerbehebung
ResourceNotFoundException	S3-Bucket nicht gefunden. Stellen Sie sicher, dass Ihr Bucket vorhanden ist.	Stellen Sie sicher, dass Ihr S3-Bucket vorhanden ist und sich in der richtigen Region befindet.
AccessDeniedException	Zugriff auf den S3-Bucket verweigert. Überprüfen Sie Ihre Bucket-Richtlinie.	Stellen Sie sicher, dass Ihre S3-Bucket-Richtlinie dem IVS-Service die erforderlichen Berechtigungen gewährt.
ValidationException	Video-Codec für die Aufzeichnung nicht unterstützt	Stellen Sie sicher, dass der Publisher einen unterstützten Video-Codec verwendet.

error_code	Grund	Anleitung zur Fehlerbehebung
InternalServerException	Interner Service-Fehler	Wiederholen Sie den Vorgang. Wenn das Problem weiterhin besteht, wenden Sie sich an den AWS Support.

Fehler beim Aufzeichnungsende: Dieses Ereignis wird übermittelt, wenn die Aufzeichnung aufgrund von Fehlern, die während der Aufzeichnung aufgetreten sind, fehlschlägt (z. B. wenn ein S3-Bucket nicht gefunden wird oder darauf nicht zugegriffen werden kann). Einige Objekte können weiterhin in den konfigurierten Speicherort geschrieben werden.

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

In der folgenden Tabelle sind die `error_code`- und `reason`-Werte für Ereignisse für Fehler beim Aufzeichnungsendes sowie Hinweise zur Fehlerbehebung aufgeführt:

error_code	Grund	Anleitung zur Fehlerbehebung
ResourceNotFoundException	S3-Bucket nicht gefunden. Stellen Sie sicher, dass Ihr Bucket vorhanden ist.	Stellen Sie sicher, dass Ihr S3-Bucket vorhanden ist und sich in der richtigen Region befindet.
AccessDeniedException	Zugriff auf den S3-Bucket verweigert. Überprüfen Sie Ihre Bucket-Richtlinie.	Stellen Sie sicher, dass Ihre S3-Bucket-Richtlinie dem IVS-Service die erforderlichen Berechtigungen gewährt.
InternalServerException	Interner Service-Fehler	Wiederholen Sie den Vorgang. Wenn das Problem weiterhin besteht, wenden Sie sich an den AWS Support.

Beachten Sie, dass IVS versucht, im selben S3-Präfix wie bei der vorherigen Sitzung aufzuzeichnen, wenn die Zusammenführung von Aufzeichnungen einzelner Teilnehmer aktiviert ist und ein Bühnen-Publisher die Verbindung zu einer Bühne trennt und dann wieder herstellt. Folglich kann die Komponente `session_id` von `recording_s3_key_prefix` in den obigen Beispielen einen anderen Wert aufweisen als das Feld `session_id` in `detail`. Siehe Zusammenführen fragmentierter Aufzeichnungen einzelner Teilnehmer.

Beispiele: Bühne-Aktualisierung

Zu den Bühne-Aktualisierung-Ereignissen gehören ein Ereignisname (der das Ereignis klassifiziert) und Metadaten zum Ereignis. Zu den Metadaten gehören die Teilnehmer-ID, die das Ereignis ausgelöst hat, die zugehörigen Bühnen- und Sitzungs-IDs sowie die Benutzer-ID.

Teilnehmer Veröffentlicht: Dieses Ereignis wird gesendet, wenn ein Teilnehmer beginnt, zu einer Bühne zu veröffentlichen.

```
{
    "version": "0",
    "id": "12345678-1a23-4567-a1bc-1a2b34567890",
    "detail-type": "IVS Stage Update",
    "source": "aws.ivs",
    "account": "123456789012",
    "time": "2020-06-23T20:12:36Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij"
    ],
    "detail": {
        "session_id": "st-ZyXwvu1T2s",
        "event_name": "Participant Published",
        "event_time": "2025-11-18T16:40:32Z",
        "user_id": "Your User Id",
        "participant_id": "xYz1c2d3e4f",
        "replica": true,
        "source_stage_arn": "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij",
        "source_session_id": "st-sdfdfdfgdfgh"
    }
}
```


Teilnehmer Unveröffentlicht: Dieses Ereignis wird gesendet, wenn ein Teilnehmer damit aufgehört hat, zu einer Bühne zu veröffentlichen.


```
{
    "version": "0",
    "id": "12345678-1a23-4567-a1bc-1a2b34567890",
    "detail-type": "IVS Stage Update",
    "source": "aws.ivs",
    "account": "123456789012",
    "time": "2020-06-23T20:12:36Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij"
    ],
    "detail": {
        "session_id": "st-ZyXwvu1T2s",
        "event_name": "Participant Unpublished",
        "event_time": "2025-11-18T16:40:32Z",
        "user_id": "Your User Id",
        "participant_id": "xYz1c2d3e4f",
```

```
        "replica": true,
        "source_stage_arn": "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij",
        "source_session_id": "st-sdfdfdfgdfgh"
    }
}
```

Fehler bei Teilnehmerveröffentlichung: Dieses Ereignis wird gesendet, wenn der Versuch eines Teilnehmers, auf einer Bühne zu veröffentlichen, fehlgeschlagen ist.

```
{
    "version": "0",
    "id": "12345678-1a23-4567-a1bc-1a2b34567890",
    "detail-type": "IVS Stage Update",
    "source": "aws.ivs",
    "account": "123456789012",
    "time": "2020-06-23T20:12:36Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij"
    ],
    "detail": {
        "session_id": "st-ZyXwvu1T2s",
        "event_name": "Participant Publish Error",
        "event_time": "2024-08-13T14:38:17.089061676Z",
        "user_id": "Your User Id",
        "participant_id": "xYz1c2d3e4f",
        "error_code": "BITRATE_EXCEEDED",
        "replica": true,
        "source_stage_arn": "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij",
        "source_session_id": "st-sdfdfdfgdfgh"
    }
}
```

Start der Teilnehmerreplikation: Dieses Ereignis wird gesendet, wenn eine Teilnehmerreplikation startet.

```
{
    "version": "0",
    "id": "12345678-1a23-4567-a1bc-1a2b34567890",
    "detail-type": "IVS Stage Update",
    "source": "aws.ivs",
    "account": "123456789012",
```

```
    "time": "2020-06-23T20:12:36Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij"
    ],
    "detail": {
        "session_id": "st-ZyXwvu1T2s",
        "event_name": "Participant Replication Start",
        "event_time": "2025-11-18T16:40:32Z",
        "user_id": "Your User Id",
        "participant_id": "xYz1c2d3e4f",
        "destination_stage_arn": "arn:aws:ivs:us-west-2:123456789012:stage/
XYZdef1G2hij",
        "destination_session_id":  "aBC1c2d3e4f"
    }
}
```

Ende der Teilnehmerreplikation: Dieses Ereignis wird gesendet, wenn eine Teilnehmerreplikation endet. Eine Replikation kann aufgrund des API-Vorgangs StopParticipantReplication enden, wenn der Publisher die Veröffentlichung beendet hat oder wenn er die Veröffentlichung beendet hat und das Zeitfenster für die Wiederverbindung abgelaufen ist.

```
{
    "version": "0",
    "id": "12345678-1a23-4567-a1bc-1a2b34567890",
    "detail-type": "IVS Stage Update",
    "source": "aws.ivs",
    "account": "123456789012",
    "time": "2020-06-23T20:12:36Z",
    "region": "us-west-2",
    "resources": [
        "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij"
    ],
    "detail": {
        "session_id": "st-ZyXwvu1T2s",
        "event_name": "Participant Replication End",
        "event_time": "2025-11-18T16:40:32Z",
        "user_id": "Your User Id",
        "participant_id": "xYz1c2d3e4f",
        "destination_stage_arn": "arn:aws:ivs:us-west-2:123456789012:stage/
XYZdef1G2hij",
        "destination_session_id":  "aBC1c2d3e4f"
    }
```

```
}
```

Token ausgetauscht: Dieses Ereignis wird gesendet, wenn ein vorhandenes Teilnehmer-Token gegen ein neues ausgetauscht wird, was zu verbesserten oder herabgestuften Token-Funktionen und/oder aktualisierten Token-Attributen führt.

```
{
    "version": "0",
    "id": "12345678-1a23-4567-a1bc-1a2b34567890",
    "detail-type": "IVS Stage Update",
    "source": "aws.ivs",
    "account": "123456789012",
    "time": "2020-06-23T20:12:36Z",
    "region": "us-west-2"
    "resources": [
        "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij"
    ],
    "detail": {
        "session_id": "st-ZyXwvu1T2s",
        "event_name": "Token Exchanged",
        "event_time": "2025-11-12T20:54:53Z",
        "user_id": "UpdatedUser",
        "participant_id": "xYz1c2d3e4f",
        "previous_token": {
            "capabilities": ["SUBSCRIBE"],
            "attributes": {
                "role": "viewer"
            },
            "user_id": "InitialUser",
            "expiration_time": "2025-11-12T21:54:52Z"
        },
        "new_token": {
            "capabilities": ["SUBSCRIBE", "PUBLISH"],
            "attributes": {
                "role": "moderator"
            },
            "user_id": "UpdatedUser",
            "expiration_time": "2025-11-12T22:54:52Z"
        }
    }
}
```

Serverseitige Zusammensetzung von IVS | Echtzeit-Streaming

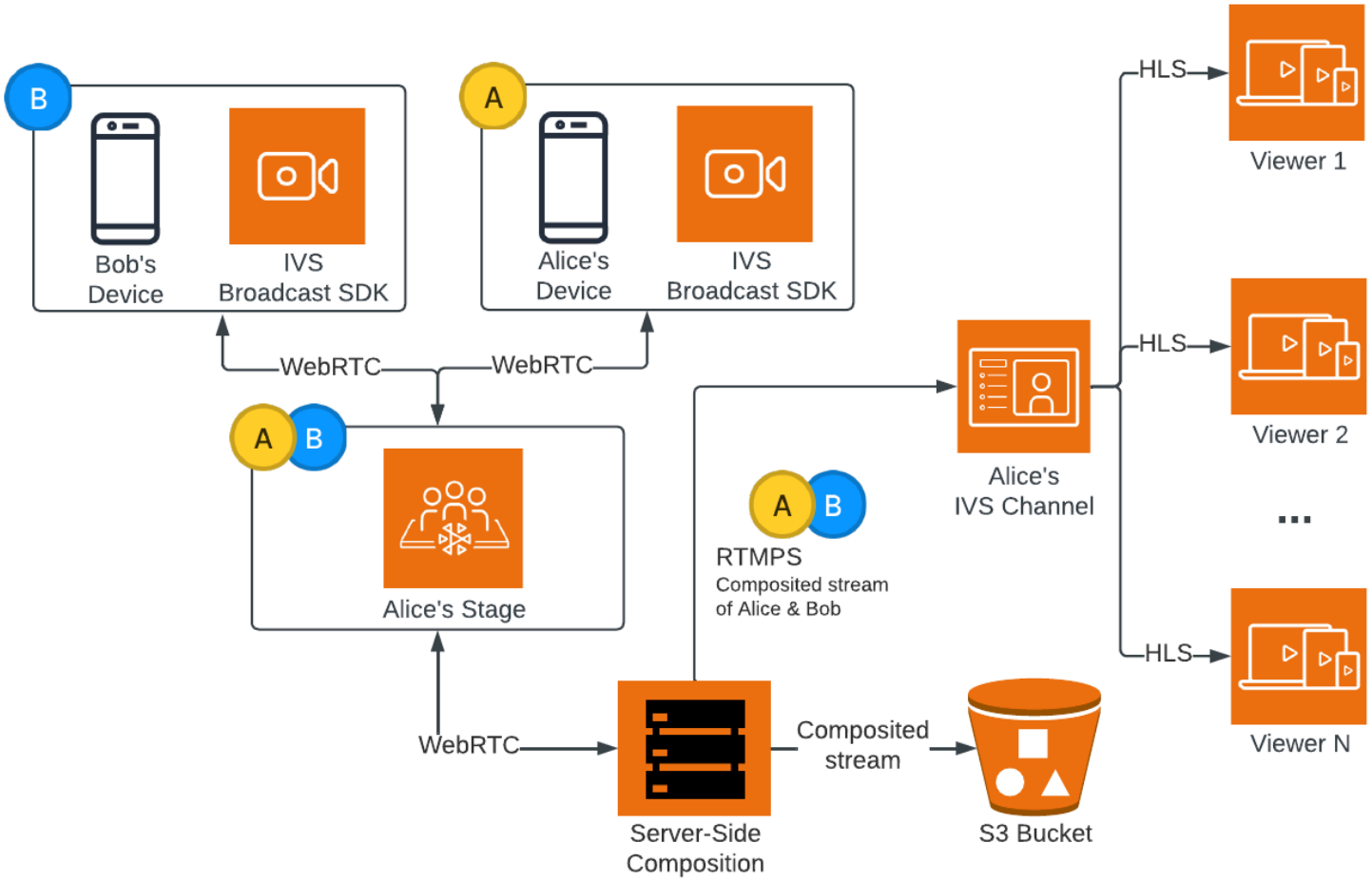
Die serverseitige Zusammensetzung verwendet einen IVS-Server, um Audio- und Videodaten von allen Teilnehmern der Stufe zu mischen und sendet dieses gemischte Video dann an einen IVS-Kanal (z. B. um ein größeres Publikum zu erreichen) oder einen S3-Bucket. Die serverseitige Zusammensetzung wird über Vorgänge der IVS-Steuerebene in der Heimatregion der Stufe aufgerufen.

Die Übertragung oder Aufzeichnung einer Stufe mittels serverseitiger Zusammensetzung bietet zahlreiche Vorteile und ist daher eine attraktive Wahl für Benutzer, die effiziente und zuverlässige cloudbasierte Video-Workflows suchen.

Themen

Überblick über IVS Serverseitige Zusammensetzung

Dieses Diagramm veranschaulicht, wie serverseitige Zusammensetzung funktioniert:



Vorteile

Im Vergleich zur clientseitigen Zusammensetzung bietet die serverseitige Zusammensetzung die folgenden Vorteile:

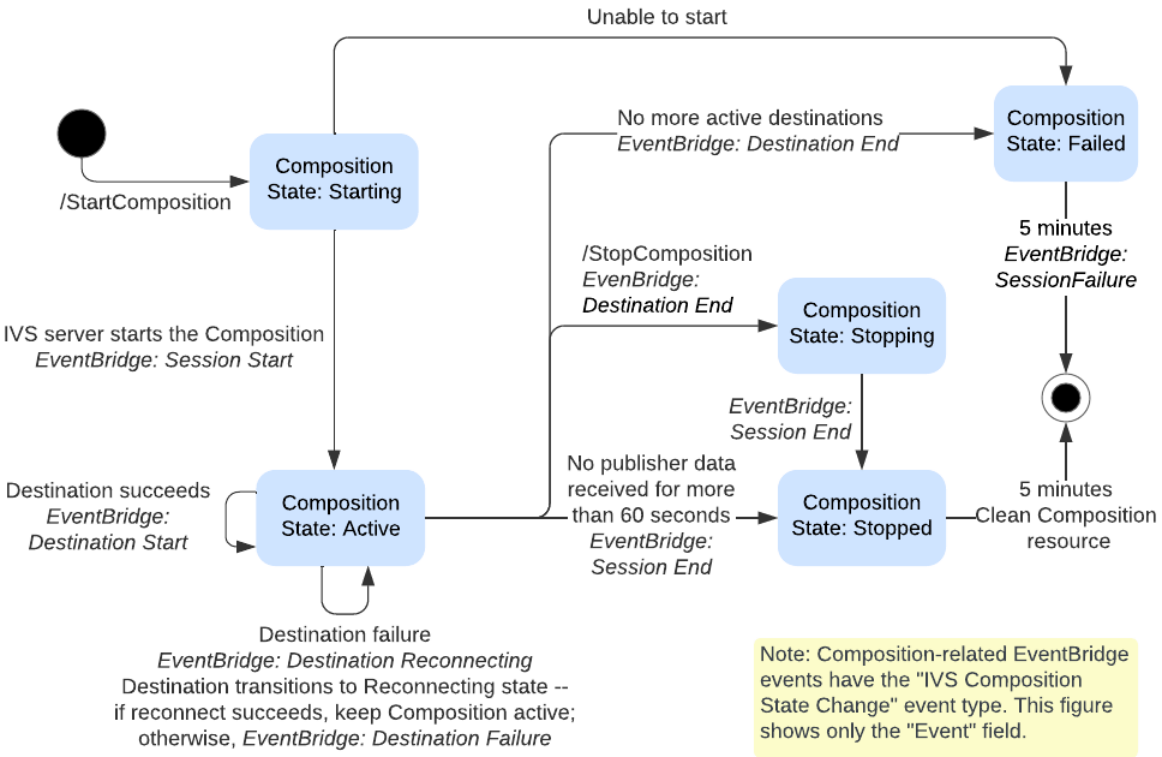
- Reduzierte Client-Last – Bei der serverseitigen Zusammensetzung wird die Last der Verarbeitung und Kombination von Audio- und Videoquellen von einzelnen Client-Geräten auf den Server selbst verlagert. Durch die serverseitige Zusammensetzung entfällt die Anforderung, dass Client-Geräte ihre CPU- und Netzwerkressourcen für die Zusammensetzung der Anzeige und deren Übertragung an IVS verwenden. Dies bedeutet, dass Zuschauer die Übertragung ansehen können, ohne dass ihre Geräte ressourcenintensive Aufgaben erledigen müssen, was zu einer verbesserten Akkulaufzeit und einem flüssigeren Seherlebnis führen kann.
- Gleichbleibende Qualität – Die serverseitige Zusammensetzung ermöglicht eine präzise Kontrolle über die Qualität, Auflösung und Bitrate des endgültigen Streams. Dies gewährleistet ein einheitliches Seherlebnis für alle Zuschauer, unabhängig von den Fähigkeiten ihrer einzelnen Geräte.

- Ausfallsicherheit – Durch die Zentralisierung des Zusammensetzungsprozesses auf dem Server wird die Übertragung stabiler. Selbst wenn ein Publisher-Gerät technischen Einschränkungen oder Schwankungen unterliegt, kann sich der Server anpassen und allen Zuschauermitgliedern einen reibungsloseren Stream bieten.
- Bandbreiteneffizienz – Da der Server die Zusammensetzung übernimmt, müssen Stufen-Publisher keine zusätzliche Bandbreite für die Übertragung des Videos an IVS aufwenden.

Um eine Stufe an einen IVS-Kanal zu übertragen, können Sie alternativ die Zusammensetzung clientseitig durchführen; Weitere Informationen finden Sie unter Aktivierung mehrerer Hosts auf einem IVS-Stream im Benutzerhandbuch zum IVS-Streaming mit niedriger Latenz.

Lebenszyklus einer Zusammensetzung

Verwenden Sie das folgende Diagramm, um sich mit den Statusübergänge einer Zusammensetzung vertraut zu machen:



Auf einer hohen Ebene sieht der Lebenszyklus einer Zusammensetzung wie folgt aus:

1. Eine Composition-Ressource wird erstellt, wenn der Benutzer den Vorgang StartComposition aufruft.

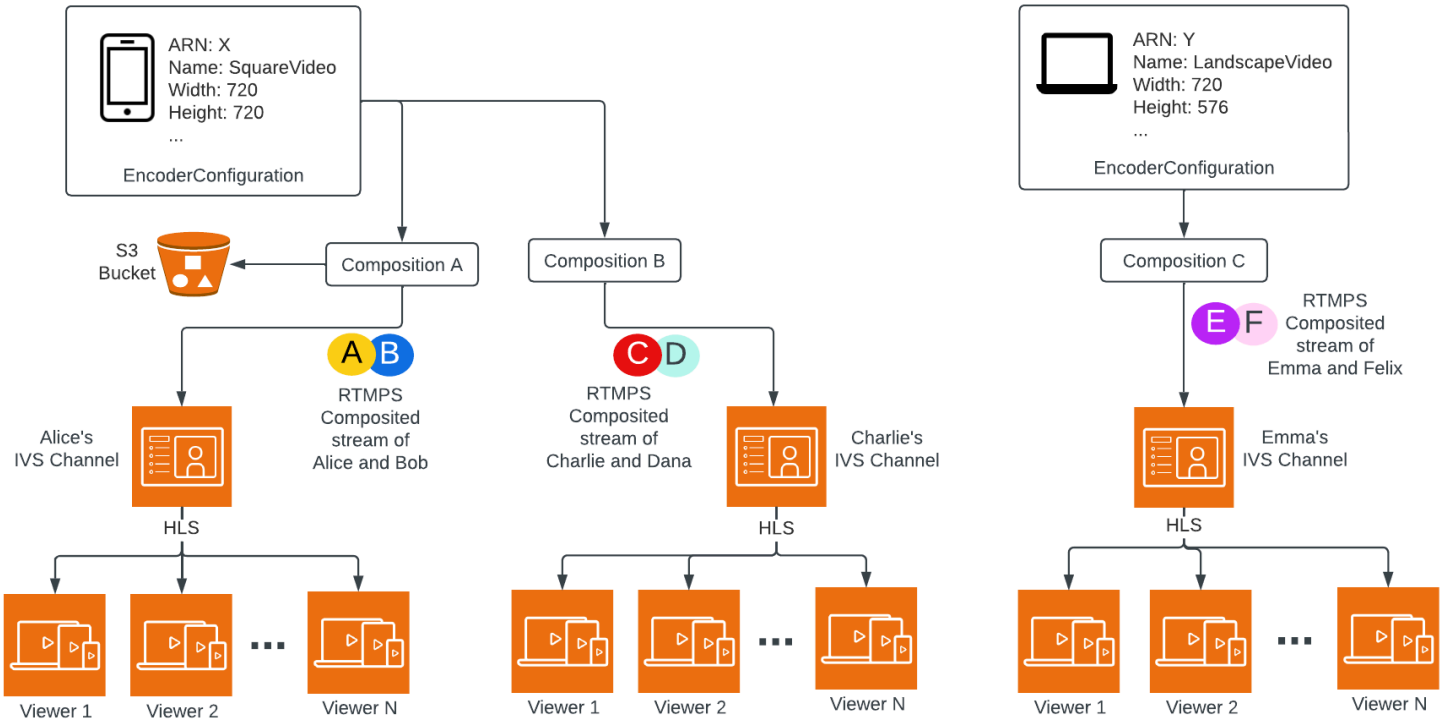
2. Sobald IVS die Zusammensetzung erfolgreich startet, wird ein EventBridge-Ereignis „Statusänderung IVS-Zusammensetzung (Sitzungsstart)“ gesendet. Einzelheiten zu Ereignissen finden Sie unter Verwenden von EventBridge mit IVS-Echtzeit-Streaming.
3. Sobald sich eine Zusammensetzung im aktiven Status befindet, kann Folgendes passieren:
   - Benutzer hält die Zusammensetzung an – Wenn der Vorgang StopComposition aufgerufen wird, initiiert IVS ein ordnungsgemäßes Herunterfahren der Zusammensetzung und sendet „Zielende“-Ereignisse, gefolgt von einem „Sitzungsende“-Ereignis.
   - Zusammensetzung wird automatisch heruntergefahren – Wenn die IVS-Stufe gelöscht wird oder 60 Sekunden lang kein Teilnehmer in der IVS-Stufe aktiv veröffentlicht, wird die Zusammensetzung automatisch abgeschlossen und EventBridge-Ereignisse werden gesendet.
   - Zielfehler – Wenn ein Ziel unerwartet ausfällt (z. B. wenn der IVS-Kanal gelöscht wird), wechselt das Ziel in den `RECONNECTING`-Status und es wird ein Ereignis „Wiederverbindung zum Ziel“ gesendet. Wenn eine Wiederherstellung nicht möglich ist, versetzt IVS das Ziel in den `FAILED`-Status und es wird ein „Zielfehler“-Ereignis gesendet. IVS hält die Zusammensetzung aktiv, wenn mindestens eines ihrer Ziele aktiv ist.
4. Sobald sich die Zusammensetzung im `STOPPED`- oder `FAILED`-Status befindet, wird sie nach fünf Minuten automatisch bereinigt. (Dann wird sie nicht mehr von ListCompositions oder GetComposition abgerufen.)

IVS-API

Die serverseitige Zusammensetzung verwendet diese wichtigen API-Elemente:

- Mit einem EncoderConfiguration-Objekt können Sie das Format des zu generierenden Videos anpassen (Höhe, Breite, Bitrate und andere Streaming-Parameter). Sie können eine EncoderConfiguration bei jedem Aufruf des Vorgangs StartComposition wiederverwenden.
- Composition-Vorgänge verfolgen die Video-Zusammensetzung und die Ausgabe über einen IVS-Kanal.
- StorageConfiguration verfolgt den S3-Bucket, in dem Zusammensetzungen aufgezeichnet werden.

Um die serverseitige Zusammensetzung zu verwenden, müssen Sie eine EncoderConfiguration erstellen und diese beim Aufruf des Vorgangs StartComposition anfügen. In diesem Beispiel wird die SquareVideo EncoderConfiguration in zwei Zusammensetzungen verwendet:



Vollständige Informationen finden Sie unter API-Referenz zu IVS-Echtzeit-Streaming.

Layouts

Der Vorgang StartComposition bietet zwei Layoutoptionen: Raster und PiP (Bild-in-Bild).

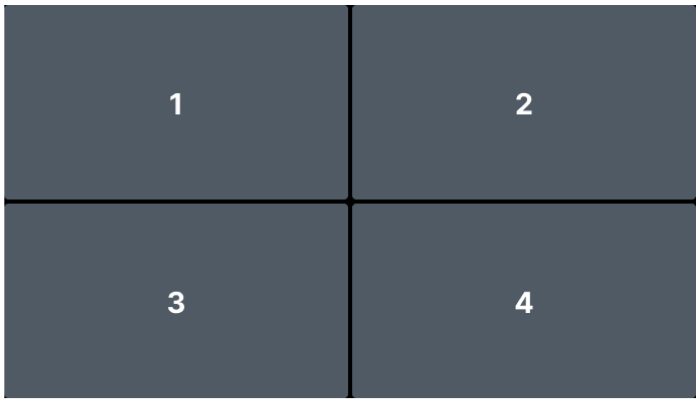
Raster-Layout

Das Raster-Layout ordnet Bühnenteilnehmer in einem Raster aus gleichgroßen Slots an. Er bietet mehrere anpassbare Eigenschaften:

- `videoAspectRatio` legt den Anzeigemodus für Teilnehmer fest, um das Seitenverhältnis von Videokacheln zu steuern.
- `videoFillMode` legt fest, wie Videoinhalte in die Teilnehmerkachel passen.
- `gridGap` gibt den Abstand zwischen Teilnehmerkacheln in Pixeln an.
- `omitStoppedVideo` ermöglicht das Ausschließen gestoppter Videostreams aus der Zusammensetzung.
- `featuredParticipantAttribute` identifiziert den hervorgehobenen Slot. Wenn diese Option aktiviert ist, wird der hervorgehobene Teilnehmer in einem größeren Slot auf dem Hauptbildschirm angezeigt, andere Teilnehmer werden darunter angezeigt.

- `participantOrderAttribute` ermöglicht eine benutzerdefinierte Teilnehmeranordnung auf Grundlage der Attributwerte in den Teilnehmer-Token. Wenn diese Option angegeben ist, werden die Teilnehmer numerisch nach ihren Attributwerten angeordnet. Bei Teilnehmern ohne das Attribut erfolgt die Anordnung wieder nach der Ankunftszeit. Das bietet eine optionale deterministische Positionierung und ermöglicht rollenbasierte Layouts.

Einzelheiten zum Raster-Layout (einschließlich gültiger Werte und Standardwerte für alle Felder) finden Sie unter dem Datentyp GridConfiguration.



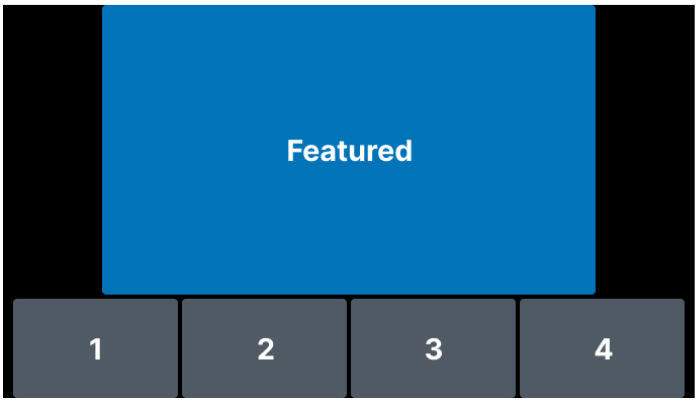


Bild-in-Bild (PiP)-Layout

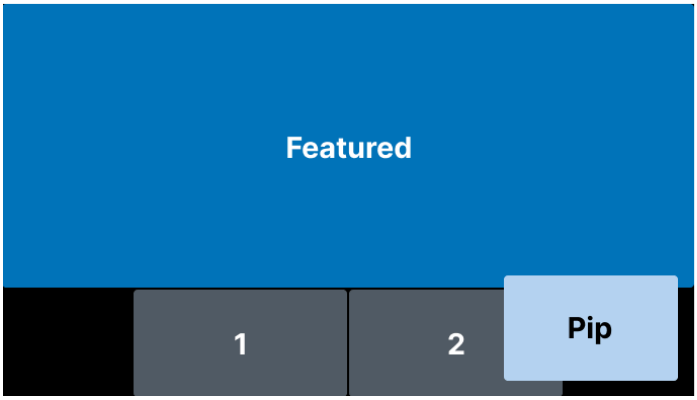
Das PiP-Layout ermöglicht die Anzeige eines Teilnehmers in einem Overlay-Fenster mit konfigurierbarer Größe, Position und Verhaltensweise. Zu den wichtigsten Eigenschaften gehören:

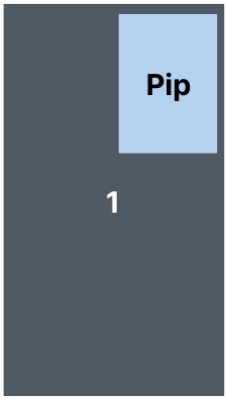
- `pipParticipantAttribute` gibt den Teilnehmer für das PiP-Fenster an.
- `pipPosition` bestimmt die Eckposition des PiP-Fensters.
- `pipWidth` und `pipHeight` konfigurieren die Breite und Höhe des PiP-Fensters.

- `pipOffset` legt die Versatzposition des PiP-Fensters in Pixeln von den nächstgelegenen Kanten fest.
- `pipBehavior` definiert das PiP-Verhalten, wenn alle anderen Teilnehmer gegangen sind.

Wie das Raster-Layout unterstützt auch das PiP-Layout `featuredParticipantAttribute`, `omitStoppedVideo`, `videoFillMode`, `gridGap` und `participantOrderAttribute` zur weiteren Anpassung der Zusammensetzung. Das `participantOrderAttribute` ermöglicht eine benutzerdefinierte Teilnehmeranordnung für die Auswahl der Teilnehmer für das PiP-Fenster und für die Positionierung der Raster-Teilnehmer auf Grundlage der Attributwerte in den Teilnehmer-Token.

Einzelheiten zum PiP-Layout (einschließlich gültiger Werte und Standardwerte für alle Felder) finden Sie unter dem Datentyp PipConfiguration.





Hinweis: Die maximale Auflösung, die von einem Stufen-Publisher bei serverseitiger Zusammensetzung unterstützt wird, beträgt 1080 p. Wenn ein Publisher Videos mit einer höheren Auflösung als 1080 p sendet, wird der Publisher als reiner Audio-Teilnehmer gerendert.

Wichtig: Stellen Sie sicher, dass Ihre Anwendung nicht von den spezifischen Features des aktuellen Layouts abhängt, z. B. von Größe und Position der Kacheln. Visuelle Verbesserungen an Layouts können jederzeit vorgenommen werden.

Erste Schritte mit IVS Serverseitige Zusammensetzung

Dieses Dokument führt Sie durch die Schritte zum Einstieg in IVS Serverseitige Zusammensetzung.

Voraussetzungen

Um die serverseitige Zusammensetzung verwenden zu können, müssen Sie über eine Stage mit aktiven Publishern verfügen und einen IVS-Kanal und/oder einen S3-Bucket als Zusammensetzungsziel verwenden.

Informationen zum Erstellen eines S3-Buckets finden Sie in der S3-Dokumentation zum Erstellen von Buckets. Der S3-Bucket muss in der gleichen AWS-Region wie die IVS-Stage erstellt werden.

Wichtig: Wenn Sie einen vorhandenen S3-Bucket nutzen, ist Folgendes zu berücksichtigen:

- Für die Einstellung Objekteigentümerschaft muss entweder Bucket-Eigentümer erzwungen oder Bucket-Eigentümer bevorzugt aktiviert sein.
- Die Standardverschlüsselung muss Serverseitige Verschlüsselung mit von Amazon S3 verwalteten Schlüsseln (SSE-S3) sein.

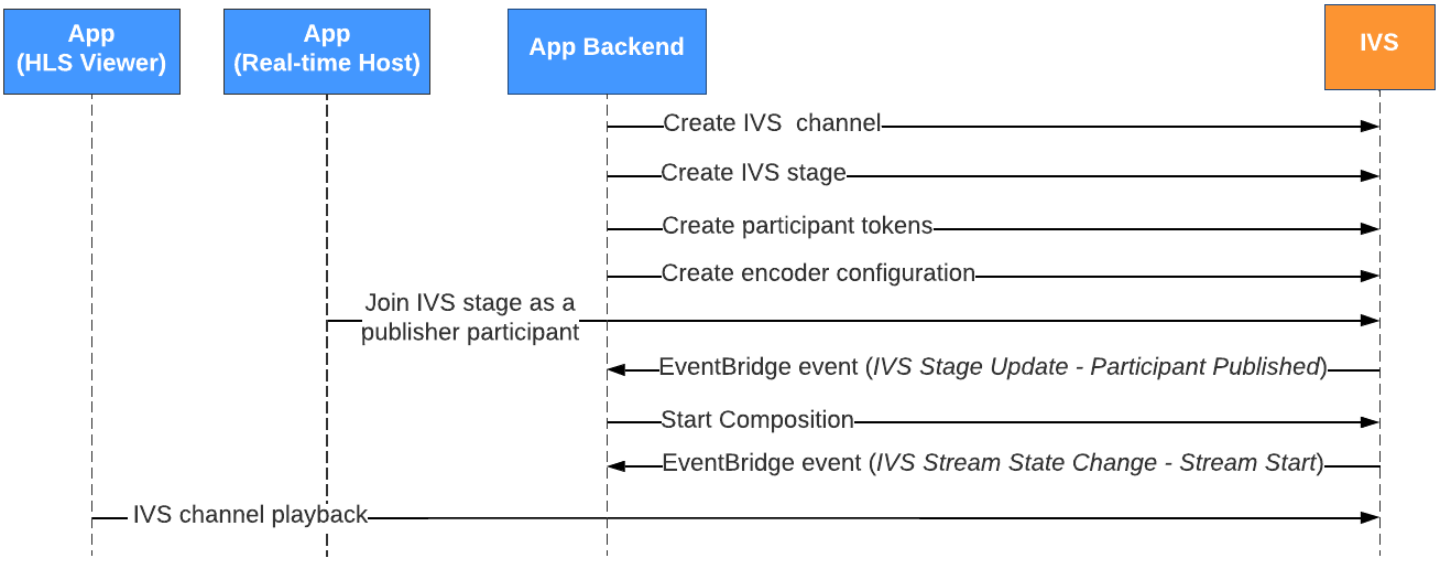
Einzelheiten finden Sie in der S3-Dokumentation zum Steuern der Eigentümerschaft von Objekten und zum Datenschutz durch Verschlüsselung.

API-Anweisungen

Nachfolgend wird ein möglicher Workflow beschrieben, der EventBridge-Ereignisse verwendet, um eine Zusammensetzung zu starten, die die Stage an einen IVS-Kanal sendet, wenn ein Teilnehmer etwas veröffentlicht. Alternativ können Sie Zusammensetzungen basierend auf Ihrer eigenen App-Logik starten und stoppen. Unter Zusammengesetzte Aufzeichnung finden Sie ein weiteres Beispiel, das die Verwendung serverseitiger Zusammensetzung zur direkten Aufzeichnung einer Stage in einem S3-Bucket demonstriert.

1. Erstellen Sie einen IVS-Kanal. Weitere Informationen finden Sie unter Erste Schritte mit Amazon-IVS-Streaming mit niedriger Latenz.

2. Erstellen Sie für jeden Publisher eine IVS-Stage und Teilnehmer-Tokens.
3. Erstellen Sie eine EncoderConfiguration.
4. Treten Sie der Stage bei und veröffentlichen Sie dort. (Weitere Informationen finden Sie in den Abschnitten „Veröffentlichen und Abonnieren“ der SDK-Anleitungen für Echtzeit-Streaming-Broadcasts: Web, Android und iOS.)
5. Wenn Sie ein vom Teilnehmer veröffentlichtes EventBridge-Ereignis erhalten, rufen Sie StartComposition mit Ihrer gewünschten Layout-Konfiguration auf.
6. Warten Sie einige Sekunden und sehen Sie sich die zusammengesetzte Ansicht in der Kanalwiedergabe an.



Hinweis: Eine Zusammensetzung wird nach 60 Sekunden Inaktivität von Publisher-Teilnehmern in der Stage automatisch heruntergefahren. An diesem Punkt wird die Zusammensetzung beendet und geht in einen `STOPPED`-Status über. Eine Zusammensetzung wird nach einigen Minuten im `STOPPED`-Status automatisch gelöscht.

CLI-Anweisungen

Die Verwendung von AWS CLI ist eine Advanced Option und erfordert, dass Sie zuerst die CLI auf Ihrem Computer herunterladen und konfigurieren. Informationen zu den ersten Schritten finden Sie im Benutzerhandbuch für die AWS-Befehlszeilenschnittstelle.

Nun können Sie mit der CLI Ressourcen erstellen und verwalten. Die Composition-Vorgänge befinden sich unter dem Namespace `ivs-realtime`.

Erstllen der EncoderConfiguration-Ressource

Eine EncoderConfiguration ist ein Objekt, mit dem Sie das Format des generierten Videos (Höhe, Breite, Bitrate und andere Streaming-Parameter) anpassen können. Sie können eine EncoderConfiguration bei jedem Aufruf des Composition-Vorgangs wiederverwenden, wie im nächsten Schritt erläutert.

Der folgende Befehl erstellt eine EncoderConfiguration-Ressource, die serverseitige Parameter für die Videozusammensetzung wie Videobitrate, Bildrate und Auflösung konfiguriert:

```
aws ivs-realtime create-encoder-configuration --name "MyEncoderConfig" --video
 "bitrate=2500000,height=720,width=1280,framerate=30"
```

Die Antwort ist:

```
{
"encoderConfiguration": {
  "arn": "arn:aws:ivs:us-east-1:927810967299:encoder-configuration/9W59OBY2M8s4",
  "name": "MyEncoderConfig",
  "tags": {},
  "video": {
  "bitrate": 2500000,
  "framerate": 30,
  "height": 720,
  "width": 1280
    }
  }
}
```

Starten einer Zusammensetzung

Erstellen Sie mithilfe des in der obigen Antwort bereitgestellten EncoderConfiguration-ARN Ihre Zusammensetzungsressource:

Beispiel für ein Raster-Layout

```
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/
DOlMW4dfMR8r", "encoderConfigurationArn": "arn:aws:ivs:us-
```

```
east-1:927810967299:encoder-configuration/9W59OBY2M8s4"}}]' --layout '{"grid":
{"participantOrderAttribute":"order","featuredParticipantAttribute":"isFeatured","videoFillMode
```

Beispiel für ein PiP-Layout

```
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/
DOlMW4dfMR8r", "encoderConfigurationArn": "arn:aws:ivs:us-
east-1:927810967299:encoder-configuration/DEkQHWPVaOwO"}}]' --layout '{"pip":
{"participantOrderAttribute":"priority","pipParticipantAttribute":"isPip","pipOffset":10,"pipPo
```

Hinweis: Mit diesem Tool können Sie die `--layout`-Konfiguration basierend auf Ihrer Layout-Auswahl einfacher generieren.

Die Antwort zeigt, dass die Zusammensetzung mit einem `STARTING`-Status erstellt wurde. Sobald die Zusammensetzung mit der Veröffentlichung der Zusammensetzung beginnt, geht der Status in `ACTIVE` über. (Sie können den Status anzeigen, indem Sie den Vorgang ListCompositions oder GetComposition aufrufen.)

Sobald eine Zusammensetzung `ACTIVE` ist, wird die zusammengesetzte Ansicht der IVS-Stage mithilfe von ListCompositions auf dem IVS-Kanal angezeigt:

```
aws ivs-realtime list-compositions
```

Die Antwort ist:

```
{
"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": {}
    }
  ]
}
```

Hinweis: Damit die Zusammensetzung aktiv bleibt, müssen die Publisher-Teilnehmer aktiv in der Stage veröffentlichen. Weitere Informationen finden Sie in den Abschnitten „Veröffentlichen und Abonnieren“ der SDK-Anleitungen für Echtzeit-Streaming-Broadcasts: Web, Android und iOS. Sie müssen für jeden Teilnehmer ein eigenes Stage-Token erstellen.

Benutzerdefinierte Teilnehmeranordnung

Mit der benutzerdefinierten Teilnehmeranordnung können Sie die Positionierung der Teilnehmer im Raster- und im PiP-Layout auf Grundlage benutzerdefinierter Attributwerte in den Teilnehmer-Token steuern. Dazu gehört auch die Positionierung der hervorgehobenen Teilnehmer und die Auswahl der Teilnehmer für das PiP-Fenster. Das bietet eine deterministische Positionierung und ermöglicht rollenbasierte Layouts.

Funktionsweise der benutzerdefinierten Anordnung

Wenn `participantOrderAttribute` in Ihrer Layoutkonfiguration angegeben ist, werden die Teilnehmer nach den folgenden Regeln angeordnet:

- Teilnehmer, deren Token das angegebene Anordnungsattribut enthalten, werden an erster Stelle positioniert und numerisch nach ihren Attributwerten angeordnet.
- Für Teilnehmer ohne das Anordnungsattribut erfolgt die Anordnung wieder nach der Ankunftszeit. Sie werden hinter den angeordneten Teilnehmern positioniert.
- Wenn mehrere Teilnehmer identische Anordnungswerte haben, werden sie auf der Bühne nach ihrer Ankunftszeit untersortiert.
- Bei der Anordnung kommt eine numerische Sortierung (keine lexikografische) zum Einsatz, sodass „10" nach „9" kommt (nicht nach „1").
- Negative Werte werden unterstützt. Sie werden vor positiven Werten positioniert.
- Nichtnumerische Werte (z. B. „abc“, „1,5") werden als ungültig behandelt. Bei diesen Teilnehmern erfolgt die Anordnung wieder nach der Ankunftszeit.

Wichtig: Die Teilnehmeranordnung wird (unabhängig davon, ob sie nach der Ankunftszeit oder benutzerdefiniert erfolgt) erst wirksam, nachdem die Zusammensetzung begonnen hat. Für

Teilnehmer, die vor Beginn der Zusammensetzung die Bühne betreten, kann die korrekte Anordnung nicht garantiert werden.

Erstellen von Token mit Anordnungsattributen

Um die benutzerdefinierte Teilnehmeranordnung zu nutzen, fügen Sie das Anordnungsattribut in die Teilnehmer-Token ein, wenn Sie diese erstellen:

```
aws ivs-realtime create-participant-token --stage-arn "arn:aws:ivs:us-
east-1:123456789012:stage/u9OiE29bT7Xp" --attributes order=1

aws ivs-realtime create-participant-token --stage-arn "arn:aws:ivs:us-
east-1:123456789012:stage/u9OiE29bT7Xp" --attributes order=2

aws ivs-realtime create-participant-token --stage-arn "arn:aws:ivs:us-
east-1:123456789012:stage/u9OiE29bT7Xp" --attributes order=3
```

Sie können das Attribut für die benutzerdefinierte Teilnehmeranordnung mit den Attributen zur Auswahl der Teilnehmer für den hervorgehobenen Slot und das PiP-Fenster kombinieren:

```
aws ivs-realtime create-participant-token --stage-arn "arn:aws:ivs:us-
east-1:123456789012:stage/u9OiE29bT7Xp" --attributes order=2,isFeatured=true

aws ivs-realtime create-participant-token --stage-arn "arn:aws:ivs:us-
east-1:123456789012:stage/u9OiE29bT7Xp" --attributes order=3,isFeatured=true

aws ivs-realtime create-participant-token --stage-arn "arn:aws:ivs:us-
east-1:123456789012:stage/u9OiE29bT7Xp" --attributes order=4,isPip=true
```

Beispielanwendungsfälle

Zu den beispielhaften Anwendungsfällen gehören:

- Konsistente Positionierung – Die Teilnehmer behalten ihre Positionen bei, wenn sie sich mit demselben Token erneut verbinden.
- Rollenbasierte Positionierung – Sie könnten beispielsweise Lehrer mit order=1 und Schüler mit order=2 angeben.
- Prioritätsbasierte Layouts – VIP-Teilnehmer mit niedrigeren Anordnungswerten werden zuerst angezeigt.

- Dynamische Layouts – Für komplexe Szenarien können Sie die benutzerdefinierte Anordnung mit `featuredParticipantAttribute` und `pipParticipantAttribute` kombinieren.
- Bühnenübergreifende Interaktionen – Wenn Sie für Szenarien wie Wettbewerbe im VS-Modus, bei denen Streamende von verschiedenen Bühnen interagieren, die Teilnehmerreplikation nutzen, können Sie die Anordnungsattribute außer Kraft setzen, um die Positionierung in der Zusammensetzung der Zielbühne festzulegen.

Hinweis: In Anwendungsfällen mit Teilnehmerreplikation können Sie die Teilnehmerattribute (einschließlich des Anordnungsattributs) beim Start einer Replikation nach Bedarf außer Kraft setzen, um das gewünschte Layout in der Zielbühne zu erreichen.

Abwärtskompatibilität

Die benutzerdefinierte Teilnehmeranordnung ist eine optionale Funktion und vollständig abwärtskompatibel. Bestehende Zusammensetzungen ohne `participantOrderAttribute` funktionieren unverändert weiter, wobei die Anordnung nach der Ankunftszeit erfolgt. Wenn `participantOrderAttribute` auf eine leere Zeichenfolge gesetzt ist, ignoriert das System die benutzerdefinierte Anordnung vollständig und kehrt zum Standardverhalten zurück.

Bildschirmfreigabe in IVS Serverseitige Zusammensetzung aktivieren

Führen Sie die folgenden Schritte aus, um ein festes Bildschirmfreigabe-Layout zu verwenden.

Erstllen der EncoderConfiguration-Ressource

Mit dem folgenden Befehl wird eine EncoderConfiguration-Ressource erstellt, die serverseitige Zusammensetzungsparameter (Video-Bitrate, Framerate und Auflösung) konfiguriert.

```
aws ivs-realtime create-encoder-configuration --name "test-ssc-with-screen-share" --
video={bitrate=2000000,framerate=30,height=720,width=1280}
```

Erstellen Sie ein Stufen-Teilnehmer-Token mit einem `screen-share`-Attribut. Da wir `screen-share` als Namen des `featured`-Slots angeben, müssen wir ein Stufen-Token erstellen, bei dem das `screen-share`-Attribut auf `true` festgelegt ist:

```
aws ivs-realtime create-participant-token --stage-arn "arn:aws:ivs:us-
east-1:123456789012:stage/u9OiE29bT7Xp" --attributes screen-share=true
```

Die Antwort ist:

```
{
   "participantToken": {
      "attributes": {
         "screen-share": "true"
      },
      "expirationTime": "2023-08-04T05:26:11+00:00",
      "participantId": "E813MFklPWLF",
      "token":
 "eyJhbGciOiJLTVMiLCJ0eXAiOiJKV1QifQ.eyJleHAiOjE2OTExMjY3NzEsImlhdCI6MTY5MTA4MzU3MSwianRpIjoiRT
   }
}
```

Starten der Zusammensetzung

Um die Zusammensetzung mit dem Feature zur Bildschirmfreigabe zu starten, wird dieses Befehl verwendet:

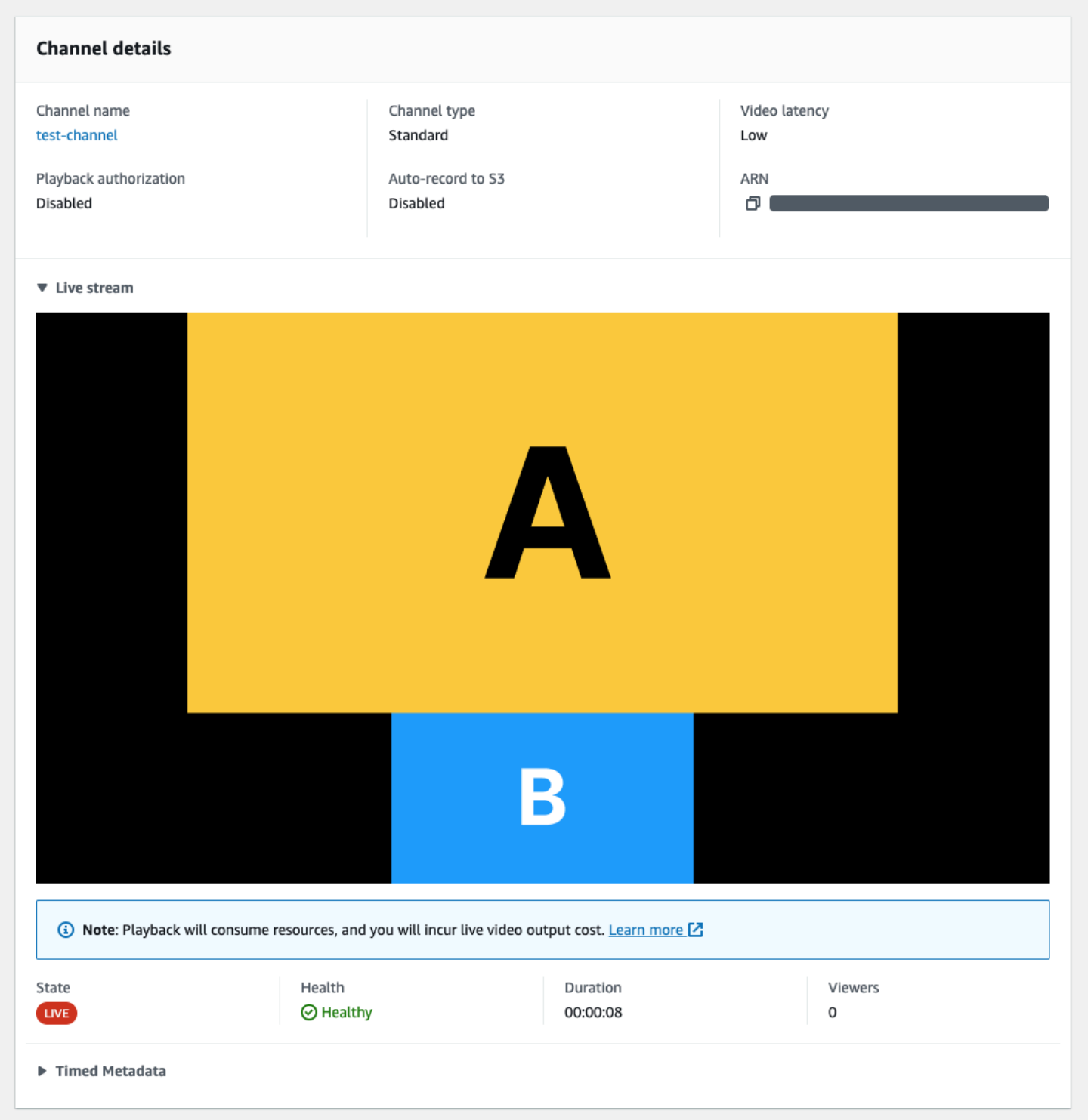
```
aws ivs-realtime start-composition --stage-arn "arn:aws:ivs:us-
east-1:927810967299:stage/8faHz1SQp0ik" --destinations  '[{"channel": {"channelArn":
 "arn:aws:ivs:us-east-1:927810967299:channel/DOlMW4dfMR8r", "encoderConfigurationArn":
 "arn:aws:ivs:us-east-1:927810967299:encoder-configuration/DEkQHWPVaOwO"}}]' --layout
 '{"grid":{"featuredParticipantAttribute":"screen-share"}}'
```

Die Antwort ist:

```
{
"composition" : {
"arn" : "arn:aws:ivs:us-east-1:927810967299:composition/B19tQcXRgtoz",
"destinations" : [ {
 "configuration" : {
 "channel" : {
    "channelArn" : "arn:aws:ivs:us-east-1:927810967299:channel/DOlMW4dfMR8r",
    "encoderConfigurationArn" : "arn:aws:ivs:us-east-1:927810967299:encoder-
configuration/DEkQHWPVaOwO"
 },
```

```
 "name" : ""
 },
 "id" : "SGmgBXTULuXv",
 "state" : "STARTING"
} ],
"layout" : {
 "grid" : {
 "featuredParticipantAttribute" : "screen-share",
 "gridGap": 2,
 "omitStoppedVideo": false,
 "videoAspectRatio": "VIDEO"
 }
},
"stageArn" : "arn:aws:ivs:us-east-1:927810967299:stage/8faHz1SQp0ik",
"startTime" : "2023-09-27T21:32:38Z",
"state" : "STARTING",
"tags" : { }
}
}
```

Wenn der Stufen-Teilnehmer `E813MFklPWLF` der Stufe beitritt, wird das Video dieses Teilnehmers im vorgestellten Slot angezeigt und alle anderen Stufen-Publishern werden unterhalb des Slots gerendert:



Anhalten der Zusammensetzung

Um eine Zusammensetzung an einem beliebigen Punkt anzuhalten, rufen Sie den Vorgang StopComposition auf:

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

Bekannte Probleme und Problemumgehungen

In diesem Abschnitt werden bekannte Probleme aufgeführt, die in IVS bei der serverseitigen Zusammensetzung auftreten können, und es werden mögliche Problemumgehungen vorgeschlagen.

- Bei einigen Kompositionen kann es nach Perioden der Stille zu einem kurzen Stottern des Audios kommen.

  Problemumgehung: Keine.

IVS-Aufzeichnung | Echtzeit-Streaming

Für das IVS-Echtzeit-Streaming gibt es zwei Aufzeichnungsoptionen:

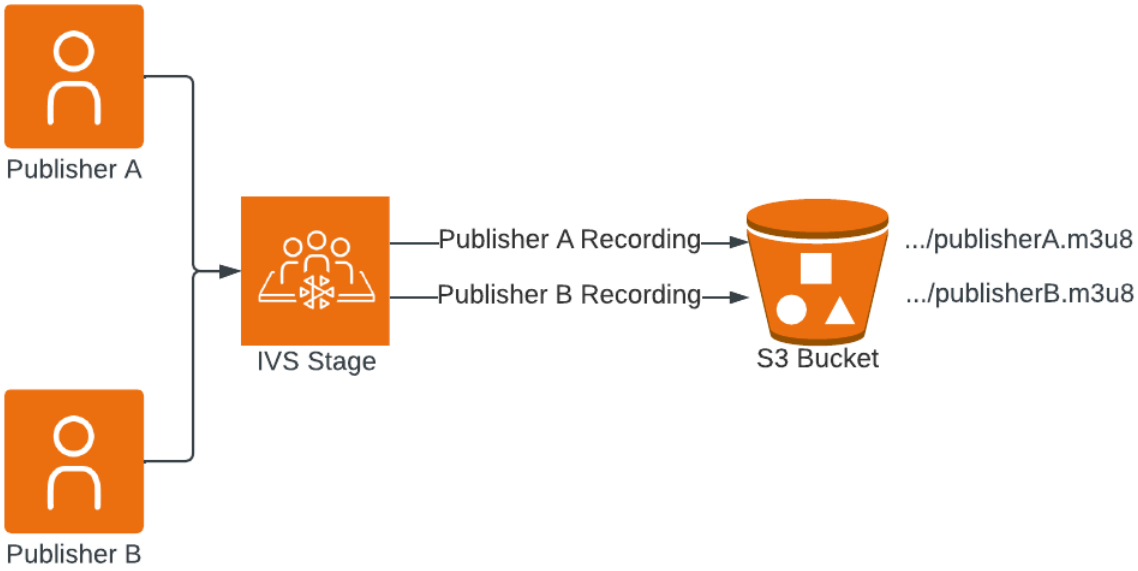
- Bei der Aufzeichnung einzelner Teilnehmer werden die Medien jedes Publishers in separaten Dateien aufgezeichnet.
- Im Gegensatz dazu fasst die zusammengesetzte Aufzeichnung Medien aller Publisher in einer einzigen Ansicht zusammen und zeichnet sie in einer Datei auf.

Für die Aufzeichnung einzelner Teilnehmer fallen keine zusätzlichen Amazon-IVS-Gebühren an, während bei der zusammengesetzten Aufzeichnung Gebühren für den Stundensatz für das codierte Video anfallen. Bei beiden Aufzeichnungsoptionen fallen standardmäßige S3-Speicher- und Anforderungskosten an. Weitere Informationen finden Sie unter Preise für Amazon IVS.

Wenn Sie eine besser anpassbare Lösung wünschen, sollten Sie das Open-Source-Projekt IVSStageSaver als Grundlage für Ihren eigenen, selbst gehosteten Aufzeichnungsdienst verwenden.

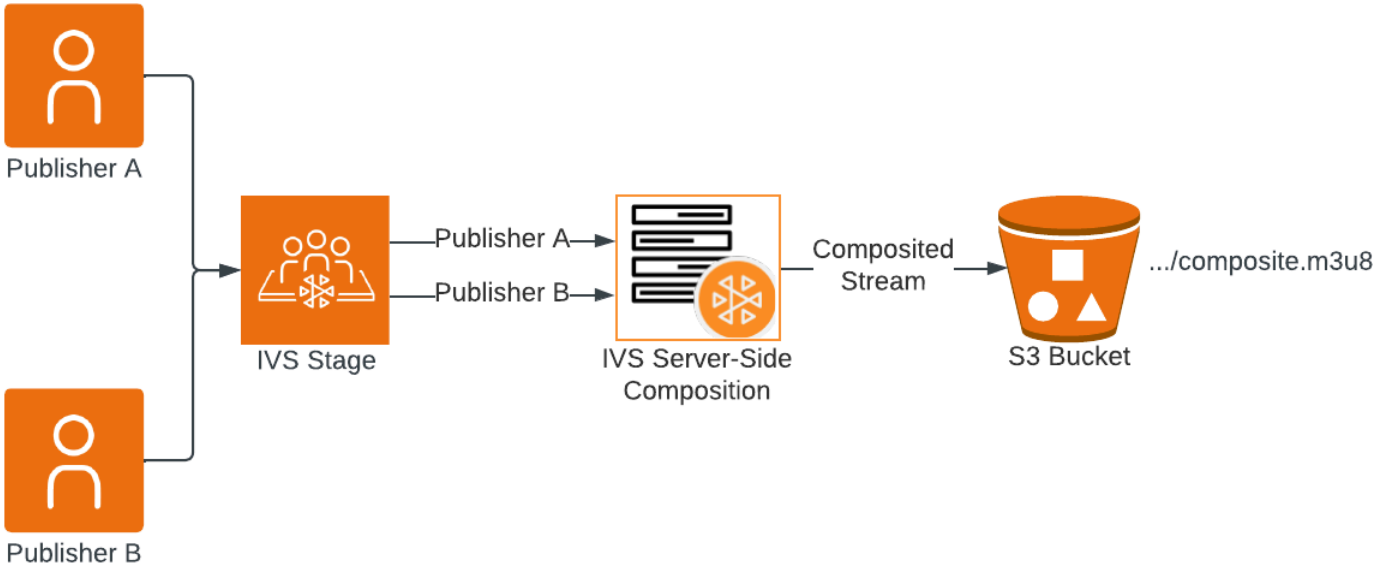
Aufzeichnung einzelner Teilnehmer

Diese Option ist ideal für Livestreams mit einem einzigen Publisher oder wenn separate Aufzeichnungen jedes Publishers benötigt werden, insbesondere zu Moderationszwecken. Weitere Informationen finden Sie unter Aufzeichnung einzelner Teilnehmer.



Zusammengesetzte Aufzeichnung

Diese Option kombiniert Medien von mehreren Publishern in einer einzigen Ansicht und zeichnet sie in einer Datei auf. Dies ist ideal für ein Video-on-Demand-Erlebnis. Weitere Informationen finden Sie unter Zusammengesetzte Aufzeichnung.



Miniaturansichten

Die Thumbnail-Aufzeichnung für IVS-Echtzeit-Streaming kann sowohl für einzelne Teilnehmeraufzeichnungen als auch für zusammengesetzte Aufzeichnungen (mehrere Teilnehmer) eingerichtet werden. Um die Aufnahme von Thumbnails zu aktivieren oder zu deaktivieren und das Intervall einzustellen, in dem Thumbnails generiert werden, gehen Sie wie folgt vor:

- Verwenden Sie die Eigenschaft `thumbnailConfiguration` für Aufzeichnungen einzelner Teilnehmer.
- Verwenden Sie die Eigenschaft `thumbnailConfigurations` für zusammengesetzte Aufzeichnungen.

Die Intervalle für Thumbnails können zwischen 1 Sekunde und 86 400 Sekunden (24 h) liegen. Details finden Sie in der API-Referenz zu Amazon-IVS-Streaming in Echtzeit.

Eine Thumbnail-Konfiguration umfasst ein `storage`-Feld, das auf `SEQUENTIAL` und/oder `LATEST` gesetzt werden kann. Das `storage`-Feld bestimmt das S3-Speicherverhalten für die Thumbnails:

- `SEQUENTIAL` speichert alle Thumbnails seriell. Dies ist die Standardeinstellung.

- `LATEST` speichert nur das neueste Thumbnail und überschreibt das vorherige.

Wenn Sie sowohl `SEQUENTIAL` als auch `LATEST` angeben, werden Thumbnails in zwei separate S3-Pfade geschrieben, einen für das sequentielle Archiv und einen für das neueste Thumbnail.

Aufzeichnung einzelner IVS-Teilnehmer | Echtzeit-Streaming

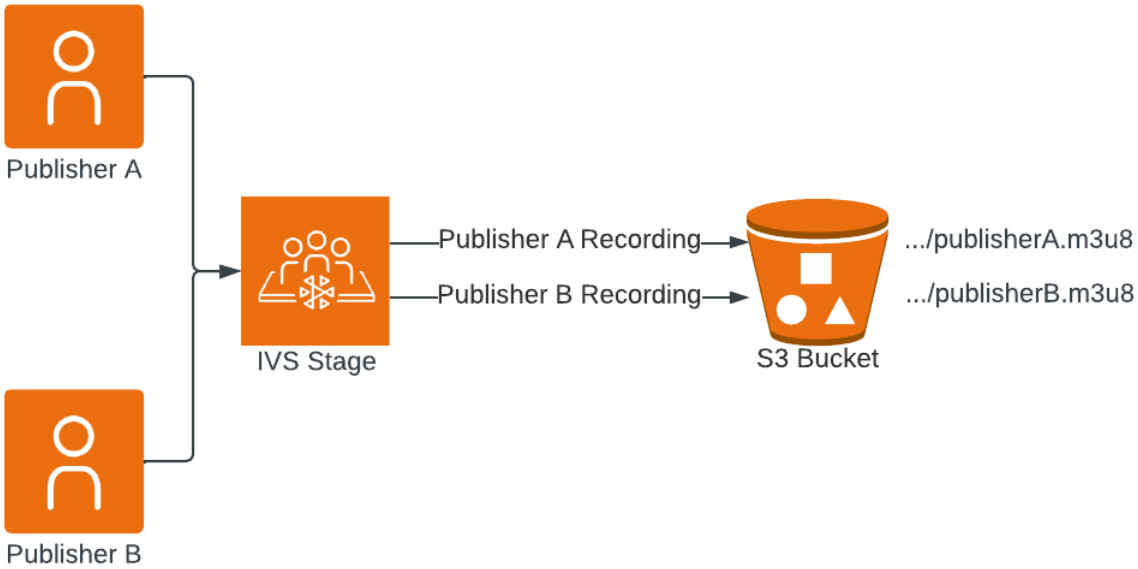
In diesem Dokument wird erklärt, wie Sie die Aufzeichnung einzelner Teilnehmer mit IVS-Streaming in Echtzeit verwenden können.

Es fallen standardmäßige S3-Speicher- und Anforderungskosten an. Für Thumbnails fallen keine zusätzlichen IVS-Gebühren an. Details finden Sie unter Preise für Amazon IVS.

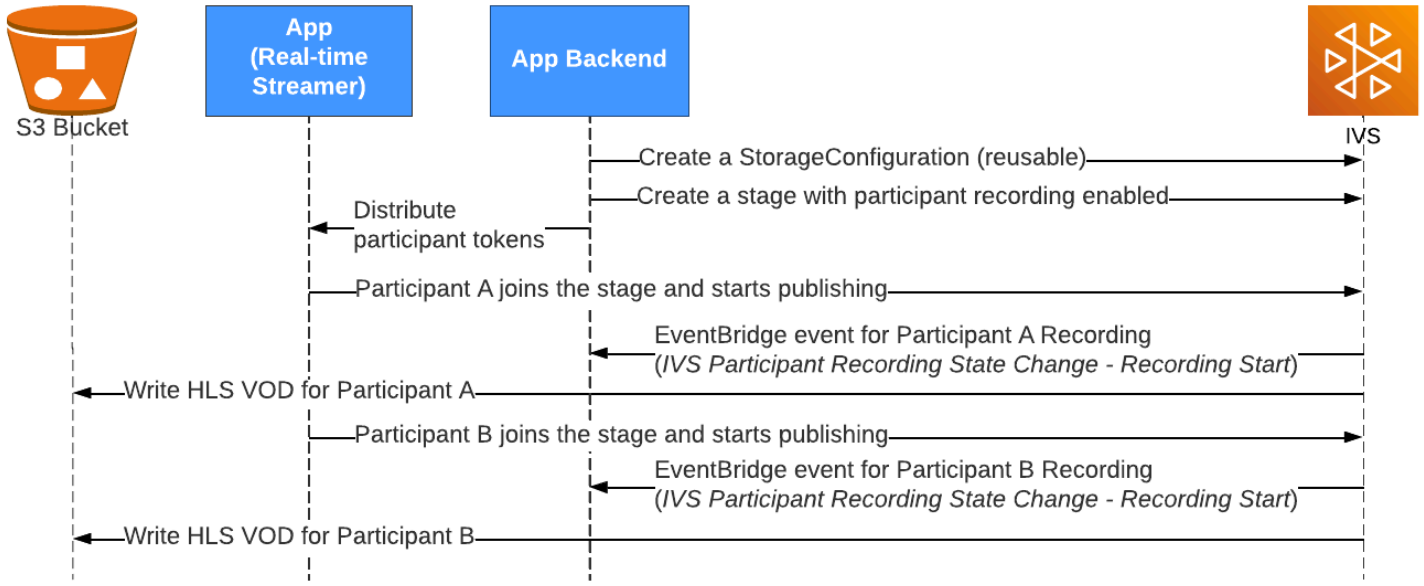
Einführung

Die Aufzeichnung einzelner Teilnehmer ermöglicht es IVS-Echtzeit-Streaming-Kunden, IVS-Stage-Publisher einzeln in S3-Buckets aufzuzeichnen. Wenn die Aufzeichnung einzelner Teilnehmer für eine Stage aktiviert ist, werden Publisher-Inhalte aufgezeichnet, sobald sie mit der Veröffentlichung für die Stage beginnen.

Hinweis: Wenn Sie alle Teilnehmer der Stage in einem einzigen Video mischen möchten, ist das Feature für zusammengesetzte Aufzeichnung besser geeignet. Eine Zusammenfassung der Aufzeichnung von IVS-Echtzeit-Streaming-Inhalten finden Sie unter Aufzeichnung.



Workflow



1. Erstellen eines S3-Bucket

Es muss ein S3-Bucket vorhanden sein, um VODs zu schreiben. Einzelheiten finden Sie in der S3-Dokumentation zum Erstellen von Buckets. Beachten Sie, dass für die Aufzeichnung einzelner Teilnehmer die S3-Buckets in derselben AWS-Region wie die IVS-Stage erstellt werden müssen.

Wichtig: Wenn Sie einen vorhandenen S3-Bucket nutzen, ist Folgendes zu berücksichtigen:

- Für die Einstellung Objekteigentümerschaft muss entweder Bucket-Eigentümer erzwungen oder Bucket-Eigentümer bevorzugt aktiviert sein.
- Die Standardverschlüsselung muss Serverseitige Verschlüsselung mit von Amazon S3 verwalteten Schlüsseln (SSE-S3) sein.

Einzelheiten finden Sie in der S3-Dokumentation zum Steuern der Eigentümerschaft von Objekten und zum Datenschutz durch Verschlüsselung.

2. Erstellen eines StorageConfiguration-Objekts

Rufen Sie nach dem Erstellen eines Buckets die IVS-Echtzeit-Streaming-API auf, um ein StorageConfiguration-Objekt zu erstellen. Sobald die Speicherkonfiguration erfolgreich erstellt wurde, hat IVS die Berechtigung, in den bereitgestellten S3-Bucket zu schreiben. Sie können dieses StorageConfiguration-Objekt für mehrere Stages wiederverwenden.

3. Erstellen einer Stage mit Teilnehmer-Token

Jetzt müssen Sie eine IVS-Stage erstellen, für welche die Aufzeichnung einzelner Teilnehmer aktiviert ist (indem Sie das AutoParticipantRecordingConfiguration-Objekt festlegen), sowie Teilnehmer-Token für jeden Publisher erstellen.

Mit der folgenden Anforderung wird eine Stage mit zwei Teilnehmer-Token und aktivierter Aufzeichnung einzelner Teilnehmer erstellt.

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

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

4. Beitritt zur Stage als aktiver Publisher

Verteilen Sie die Teilnehmer-Token an Ihre Publisher und lassen Sie sie der Stage beitreten und mit der Veröffentlichung dafür beginnen.

Wenn sie der Stage beitreten und mithilfe eines der Broadcast-SDKs für Echtzeit-Streaming von IVS beginnen, auf ihr zu veröffentlichen, startet der Aufzeichnungsvorgang für die Teilnehmer automatisch und sendet Ihnen ein EventBridge-Ereignis, das angibt, dass die Aufzeichnung gestartet wurde. (Das Ereignis lautet „Statusänderung der IVS-Teilnehmeraufzeichnung – Aufzeichnungsstart“.) Gleichzeitig beginnt der Prozess der Teilnehmeraufzeichnung mit dem Schreiben der VOD- und Metadatendateien in den konfigurierten S3-Bucket. Hinweis: Es kann nicht garantiert werden, dass Teilnehmer, die über einen extrem kurzen Zeitraum (weniger als 5 Sekunden) miteinander verbunden sind, aufgezeichnet werden.

Es gibt zwei Möglichkeiten, das S3-Präfix für jede Aufzeichnung zu erhalten:

- Hören Sie das EventBridge-Ereignis ab:

```
{
   "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"
   }
}
```

- Verwenden Sie den API-Vorgang GetParticipant – die Antwort enthält den S3-Bucket und das Präfix für den Ort, an dem ein Teilnehmer aufgezeichnet wird. Hier ist die Anforderung:

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

```
{
   "participantID": "xYz1c2d3e4f",
   "sessionId": "st-ZyXwvu1T2s",
   "stageArn": "arn:aws:ivs:us-west-2:123456789012:stage/AbCdef1G2hij"
}
```

Und hier ist die Antwort:

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

5. Wiedergabe des VOD

Nachdem die Aufzeichnung abgeschlossen ist, können Sie sie mit dem IVS-Player ansehen. Anweisungen zum Einrichten von CloudFront-Distributionen für die VOD-Wiedergabe finden Sie unter Wiedergabe von aufgezeichneten Inhalten aus privaten Buckets.

Nur Audioaufzeichnung

Wenn Sie die Aufzeichnung für einzelne Teilnehmer einrichten, können Sie festlegen, dass nur Audio-HLS-Segmente in Ihren S3-Bucket geschrieben werden. Um dieses Feature zu verwenden, wählen Sie bei der Erstellung der Stage `AUDIO_ONLY mediaType` aus:

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

{
   "autoParticipantRecordingConfiguration": {
      "storageConfigurationArn": "arn:aws:ivs:us-west-2:123456789012:storage-
configuration/AbCdef1G2hij",
      "mediaTypes": ["AUDIO_ONLY"],
      "thumbnailConfiguration": {
         "recordingMode": "DISABLED"
```

```
      }
   },
   "name": "TestStage",
   "participantTokenConfigurations": [
      {
         "capabilities": ["PUBLISH", "SUBSCRIBE"],
         "duration": 20160,
         "userId": "1"
      },
      {
         "capabilities": ["PUBLISH", "SUBSCRIBE"],
         "duration": 20160,
         "userId": "2"
      }
   ]
}
```

Nur Thumbnail-Aufzeichnung

Wenn Sie die Aufzeichnung für einzelne Teilnehmer einrichten, können Sie festlegen, dass nur Thumbnails in Ihren S3-Bucket geschrieben werden. Um dieses Feature zu verwenden, legen Sie beim Erstellen der Stage für `mediaType` `NONE` fest. Dadurch wird sichergestellt, dass keine HLS-Segmente generiert werden. Thumbnails werden trotzdem erstellt und in Ihren S3-Bucket geschrieben.

```
POST /CreateStage HTTP/1.1
Content-type: application/json
{
   "autoParticipantRecordingConfiguration": {
      "storageConfigurationArn": "arn:aws:ivs:us-west-2:123456789012:storage-
configuration/AbCdef1G2hij",
      "mediaTypes": ["NONE"],
      "thumbnailConfiguration": {
         "recordingMode": "INTERVAL",
         "storage": ["LATEST", "SEQUENTIAL"],
         "targetIntervalSeconds": 60
      }
   },
   "name": "TestStage",
   "participantTokenConfigurations": [
      {
         "capabilities": ["PUBLISH", "SUBSCRIBE"],
```

```
         "duration": 20160,
         "userId": "1"
      },
      {
         "capabilities": ["PUBLISH", "SUBSCRIBE"],
         "duration": 20160,
         "userId": "2"
      }
   ]
}
```

Inhalte der Aufnahme

Wenn die Aufzeichnung einzelner Teilnehmer aktiv ist, werden HLS-Videosegmente und Thumbnails in den S3-Bucket geschrieben, der bei der Erstellung der Stage bereitgestellt wurde. Dieser Inhalt ist für die Nachbearbeitung oder Wiedergabe als On-Demand-Video verfügbar.

Beachten Sie, dass nach Abschluss einer Aufzeichnung das Ereignis „Statusänderung der IVS-Teilnehmeraufzeichnung – Aufzeichnungsende“ über EventBridge gesendet wird. Es wird empfohlen, aufgezeichnete Streams erst wiederzugeben oder zu verarbeiten, nachdem dieses Ereignis empfangen wurde. Einzelheiten finden Sie unter Verwenden von EventBridge mit IVS-Echtzeit-Streaming.

Nachfolgend finden Sie eine Beispielverzeichnisstruktur und den Inhalt einer Aufzeichnung einer Live-IVS-Sitzung:

```
s3://mybucket/stageId/stageSessionId/participantId/timestamp
   events
      recording-started.json
      recording-ended.json
   media
      hls
  multivariant.m3u8
         high
            playlist.m3u8
            1.mp4
      thumbnails
         high
            1.jpg
            2.jpg
      latest_thumbnail
         high
```

```
            thumb.jpg
```

Der Ordner `events` enthält die Metadatendateien, die dem Aufzeichnungsereignis entsprechen. JSON-Metadatendateien werden generiert, wenn die Aufzeichnung gestartet, erfolgreich beendet oder mit Fehlern beendet wird:

- `events/recording-started.json`
- `events/recording-ended.json`
- `events/recording-failed.json`

Ein angegebener `events`-Ordner enthält `recording-started.json` und entweder `recording-ended.json` oder `recording-failed.json`. Diese enthalten Metadaten, die sich auf die aufgezeichnete Sitzung und ihre Ausgabeformate beziehen. JSON-Details sind unten angegeben.

Der Ordner `media` enthält die unterstützten Medieninhalte. Der Unterordner `hls` enthält alle Medien und Manifestdateien, die während der Aufzeichnungssitzung generiert wurden, und kann mit dem IVS-Player abgespielt werden. Falls konfiguriert, enthalten die Ordner `thumbnails` und die Unterordner `latest_thumbnail` JPEG-Thumbnail-Mediendateien, die während der Aufzeichnungssitzung generiert wurden.

Zusammenführen fragmentierter Aufzeichnungen einzelner Teilnehmer

Mit der Eigenschaft `recordingReconnectWindowSeconds` einer Aufzeichnungskonfiguration können Sie ein Zeitfenster (in Sekunden) angeben, in dem IVS versucht, im selben S3-Präfix wie bei der vorherigen Sitzung aufzuzeichnen, wenn ein Stage-Publisher die Verbindung zu einer Stage trennt und dann wieder herstellt. Mit anderen Worten: Wenn ein Publisher die Verbindung trennt und dann innerhalb des angegebenen Intervalls wieder herstellt, werden die einzelnen Aufzeichnungen als eine einzige Aufzeichnung betrachtet und zusammengeführt.

Wenn die Aufzeichnung von Miniaturansichten im Modus `SEQUENTIAL` aktiviert ist, werden die Miniaturansichten ebenfalls unter demselben `recordingS3Prefix` zusammengeführt. Beim Zusammenführen der Aufzeichnungen beginnt der Miniaturansichtenzähler wieder bei dem Wert, der für die vorherige Aufzeichnung geschrieben wurde.

Ereignisse zur Änderung des IVS-Aufzeichnungsstatus in Amazon EventBridge: Aufzeichnungsende-Ereignisse und entsprechende JSON-Metadatendateien werden um mindestens `recordingReconnectWindowSeconds` verzögert, da Amazon IVS wartet, damit kein neuer Stream gestartet wird.

Eine Anleitung zum Einrichten der Funktionalität zum Zusammenführen von Streams finden Sie in Schritt 2: Erstellen einer Stage mit optionaler Teilnehmeraufzeichnung unter Erste Schritte mit Amazon-IVS-Streaming in Echtzeit.

Berechtigung

Damit mehrere Aufzeichnungen mit demselben S3-Präfix zusammengeführt werden können, müssen für alle Aufzeichnungen bestimmte Bedingungen erfüllt sein:

- Der Wert der Eigenschaft `recordingReconnectWindowSeconds` der AutoParticipantRecordingConfiguration für die Stage ist auf einen Wert größer als 0 gesetzt.
- Der `StorageConfigurationArn`, mit dem die VOD-Artefakte geschrieben wurden, ist für alle Aufzeichnungen identisch.
- Die Zeitdifferenz in Sekunden zwischen dem Verlassen der Stage und dem Wiederbeitritt des Teilnehmers ist kleiner oder gleich `recordingReconnectWindowSeconds`.

Beachten Sie, dass der Standardwert von `recordingReconnectWindowSeconds` 0 lautet, wodurch das Zusammenführen deaktiviert wird.

Synchronisieren von Aufzeichnungen mehrerer Teilnehmer

Aufzeichnungen einzelner Teilnehmer enthalten `EXT-X-PROGRAM-DATE-TIME`-Tags in HLS-Playlisten, die präzise UTC-Zeitstempel mit einer Genauigkeit von Millisekunden für die Synchronisation von Aufzeichnungen mehrerer Teilnehmer während der Nachbearbeitung bereitstellen.

Wenn Sie mehrere Teilnehmer einzeln aufzeichnen und eine synchronisierte Zusammensetzung erstellen möchten (z. B. ein Layout nebeneinander oder ein Bild-in-Bild-Layout), können Sie diese Zeitstempel verwenden, um die Aufzeichnungen genau auszurichten, selbst wenn die Teilnehmer zu unterschiedlichen Zeiten der Stage beigetreten sind oder es zu Unterbrechungen kam, die möglicherweise durch Netzwerkunterbrechungen verursacht wurden.

Die HLS-Playlist jedes Teilnehmers enthält `EXT-X-PROGRAM-DATE-TIME`-Tags, die Folgendes kennzeichnen:

- Der Beginn der Aufnahme (erstes Segment).
- Alle Diskontinuitätspunkte während der Aufnahme, z. B. wenn es zu einem Stitching kommt.

Diese Zeitstempel sind auf Millisekunden genau und werden für alle Teilnehmer anhand derselben Zeitreferenz synchronisiert.

Beispiel einer HLS-Wiedergabeliste

```
#EXTM3U
#EXT-X-VERSION:7
#EXT-X-TARGETDURATION:12
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-MAP:URI="init-0.mp4"
#EXT-X-PROGRAM-DATE-TIME:2024-01-01T12:00:00.000Z
#EXTINF:3.30091,
0.mp4
#EXTINF:5.63794,
1.mp4
#EXTINF:2.74290,
2.mp4
#EXT-X-DISCONTINUITY
#EXT-X-MAP:URI="init-1.mp4"
#EXT-X-PROGRAM-DATE-TIME:2024-01-01T12:00:52.772Z
#EXTINF:2.54412,
3.mp4
#EXTINF:5.63649,
4.mp4
```

Die `EXT-X-PROGRAM-DATE-TIME`-Tags geben die genaue UTC-Zeit für das erste Segment und an jedem Diskontinuitätspunkt an und ermöglichen so eine präzise Synchronisation mit den Aufzeichnungen anderer Teilnehmer.

Synchronisierungs-Workflow

Um Aufzeichnungen mehrerer Teilnehmer zu synchronisieren, extrahieren Sie die `EXT-X-PROGRAM-DATE-TIME`-Zeitstempel aus der HLS-Wiedergabeliste jedes Teilnehmers und verwenden Sie sie, um Zeitversätze zu berechnen. Diese Offsets können dann während der Nachbearbeitung der Zusammensetzung mit Videoverarbeitungstools wie FFmpeg angewendet werden. Wenn die Aufzeichnungen Unterbrechungen aufweisen, bieten Zeitstempel an diesen Stellen die erforderlichen Zeitreferenzen, um eine genaue Synchronisation während der gesamten Aufnahme aufrechtzuerhalten.

Hinweis: Für eine synchronisierte Ausgabe in Echtzeit ohne Nachbearbeitung sollten Sie die serverseitige Zusammensetzung anstelle der Aufzeichnung durch einzelne Teilnehmer in Betracht ziehen.

JSON-Metadatendateien

Diese Metadaten weisen das JSON-Format auf. Es enthält die folgenden Informationen:

Feld	Typ	Erforderlich	Beschreibung
`stage_arn`	Zeichenfolge	Ja	ARN der Stage, die als Quelle für die Aufzeichnung verwendet wird.
`session_id`	Zeichenfolge	Ja	Zeichenfolge, welche die `session_id` der Stage angibt, auf welcher der Teilnehmer aufgezeichnet wird.
`participant_id`	Zeichenfolge	Ja	Zeichenfolge, die die Kennung des aufgezeichneten Teilnehmers darstellt.
`recording_started_at`	Zeichenfolge	Bedingt	RFC 3339 UTC-Zeitstempel, wenn die Aufnahme gestartet wurde. Dies ist nicht verfügbar, wenn sich der `recording_status` in `RECORDING_START_FAILED` befindet. Beachten Sie auch den Hinweis unten für `recording_ended_at` .
`recording_ended_at`	Zeichenfolge	Bedingt	RFC 3339 UTC-Zeitstempel, wenn die Aufnahme beendet wurde. Dies ist nur verfügbar , wenn `recording_status` `"RECORDING_ENDED"` oder

Feld	Typ	Erforderlich	Beschreibung
			`"RECORDING_ENDED_WITH_FAILURE"` ist. Hinweis: `recording_started_at` und `recording_ended_at` sind Zeitstempel, wenn diese Ereignisse generiert werden, und stimmen möglicherweise nicht genau mit den Zeitstempeln des HLS-Videosegments überein. Um die Dauer einer Aufnahme genau zu bestimmen, verwenden Sie das Feld `duration_ms` .
`recording_status`	Zeichenfolge	Ja	Aufzeichnungsstatus. Zulässige Werte: `"RECORDING_STARTED"` , `"RECORDING_ENDED"` , `"RECORDING_START_FAILED"` , `"RECORDING_ENDED_WITH_FAILURE"` .
`recording_status_message`	Zeichenfolge	Bedingt	Beschreibende Informationen über den Status. Dies ist nur verfügbar , wenn `recording_status` `"RECORDING_ENDED"` oder `"RECORDING_ENDED_WITH_FAILURE"` ist.
`media`	object	Ja	Objekt, das die Aufzählungsobjekte von Medieninhalten enthält, die für diese Aufzeichnung verfügbar sind. Zulässiger Wert: `"hls"`.

Feld	Typ	Erforderlich	Beschreibung
`hls`	object	Ja	Aufzählungsfeld, das die Ausgabe des Apple HLS-Formats beschreibt.
`duration_ms`	Ganzzahl	Bedingt	Dauer des aufgezeichneten HLS-Inhalts in Millisekunden. Dies ist nur verfügbar, wenn `recording_status` `"RECORDING_ENDED"` oder `"RECORDING_ENDED_WITH_FAILURE"` ist. Wenn ein Fehler aufgetreten ist, bevor eine Aufzeichnung durchgeführt wurde, ist dies 0.
`path`	Zeichenfolge	Ja	Relativer Pfad vom S3-Präfix, in dem HLS-Inhalt gespeichert wird.
`playlist`	Zeichenfolge	Ja	Name der HLS-Master-Wiedergabeliste.
`renditions`	Objekt	Ja	Array von Formatversionen (HLS-Varianten) von Metadatenobjekten. Es ist immer mindestens eine Formatvariante vorhanden.
`path`	Zeichenfolge	Ja	Relativer Pfad vom S3-Präfix, in dem HLS-Inhalt für diese Formatvariante gespeichert wird.
`playlist`	Zeichenfolge	Ja	Name der Medienwiedergabelistdatei für diese Formatvariante.

Feld	Typ	Erforderlich	Beschreibung
`thumbnails`	object	Bedingt	Aufzählungsfeld, das die Ausgabe von Miniaturansichten beschreibt. Diese Funktion ist nur verfügbar, wenn das Feld der Konfiguration `storage` `SEQUENTIAL` enthält.
`path`	Zeichenfolge	Ja	Relativer Pfad vom S3-Präfix, in dem sequentieller Thumbnail-Inhalt gespeichert wird.
`renditions`	object	Ja	Array von Formatversionen (Thumbnail-Varianten) von Metadatenobjekten. Es ist immer mindestens eine Formatvariante vorhanden.
`path`	Zeichenfolge	Ja	Relativer Pfad vom S3-Präfix, in dem Thumbnail-Inhalt für diese Formatvariante gespeichert wird.
`latest_thumbnail`	object	Bedingt	Aufzählungsfeld, das die Ausgabe von Miniaturansichten beschreibt. Diese Funktion ist nur verfügbar, wenn das Feld der Konfiguration `storage` `LATEST` enthält.
`path`	Zeichenfolge	Ja	Relativer Pfad vom S3-Präfix, in dem `latest_thumbnail` gespeichert wird.
`renditions`	object	Ja	Array von Formatversionen (Thumbnail-Varianten) von Metadatenobjekten. Es ist immer mindestens eine Formatvariante vorhanden.

Feld	Typ	Erforderlich	Beschreibung
`path`	Zeichenfolge	Ja	Relativer Pfad vom S3-Präfix, in dem das neueste Thumbnail für diese Formatvariante gespeichert wird.
`version`	Zeichenfolge	Ja	Die Version des Metadatenschemas.

Beispiel: recording-started.json

```
{
   "version": "v1",
   "stage_arn": "arn:aws:ivs:us-west-2:aws_account_id:stage/AbCdef1G2hij",
   "session_id": "st-ZyXwvu1T2s",
   "participant_id": "xYz1c2d3e4f",
   "recording_started_at": "2024-03-13T13:17:17Z",
   "recording_status": "RECORDING_STARTED",
   "media": {
      "hls": {
         "path": "media/hls",
         "playlist": "multivariant.m3u8",
         "renditions": [
            {
               "path": "high",
               "playlist": "playlist.m3u8"
            }
         ]
      },
      "thumbnails": {
         "path": "media/thumbnails",
         "renditions": [
            {
               "path": "high"
            }
         ]
      },
      "latest_thumbnail": {
         "path": "media/latest_thumbnail",
```

```
         "renditions": [
            {
               "path": "high"
            }
         ]
      }
   }
}
```

Beispiel: recording-ended.json

```
{
   "version": "v1",
   "stage_arn": "arn:aws:ivs:us-west-2:aws_account_id:stage/AbCdef1G2hij",
   "session_id": "st-ZyXwvu1T2s",
   "participant_id": "xYz1c2d3e4f",
   "recording_started_at": "2024-03-13T19:44:19Z",
   "recording_ended_at": "2024-03-13T19:55:04Z",
   "recording_status": "RECORDING_ENDED",
   "media": {
      "hls": {
         "duration_ms": 645237,
         "path": "media/hls",
         "playlist": "multivariant.m3u8",
         "renditions": [
            {
               "path": "high",
               "playlist": "playlist.m3u8"
            }
         ]
      },
      "thumbnails": {
         "path": "media/thumbnails",
         "renditions": [
            {
               "path": "high"
            }
         ]
      },
      "latest_thumbnail": {
         "path": "media/latest_thumbnail",
         "renditions": [
            {
```

```
                "path": "high"
            }
         ]
      }
   }
}
```

Beispiel: 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"
            }
         ]
      },
      "thumbnails": {
         "path": "media/thumbnails",
         "renditions": [
            {
               "path": "high"
            }
         ]
      },
      "latest_thumbnail": {
         "path": "media/latest_thumbnail",
         "renditions": [
            {
               "path": "high"
            }
```

```
            ]
        }
    }
}
```

Konvertieren von Aufzeichnungen in MP4

Die Aufzeichnungen einzelner Teilnehmer werden im HLS-Format gespeichert, das aus Playlisten und fragmentierten MP4-Segmenten (fMP4) besteht. Um eine HLS-Aufzeichnung in eine einzelne MP4-Datei zu konvertieren, müssen Sie FFmpeg installieren und den folgenden Befehl ausführen:

```
ffmpeg -i /path/to/playlist.m3u8 -i /path/to/playlist.m3u8 -map 0:v -map 1:a -c copy
 output.mp4
```

Zusammengesetzte Aufzeichnung | Echtzeit-Streaming

In diesem Dokument wird erläutert, wie Sie das Feature zur Aufzeichnung von Zusammensetzungen innerhalb der serverseitigen Zusammensetzung verwenden. Mit der zusammengesetzten Aufzeichnung können Sie HLS-Aufzeichnungen einer IVS-Stage generieren, indem Sie mithilfe eines IVS-Servers alle Stage-Publisher effektiv in einer Ansicht kombinieren und das resultierende Video dann in einem S3-Bucket speichern.

Es fallen standardmäßige S3-Speicher- und Anforderungskosten an. Für Thumbnails fallen keine zusätzlichen IVS-Gebühren an. Details finden Sie unter Preise für Amazon IVS.

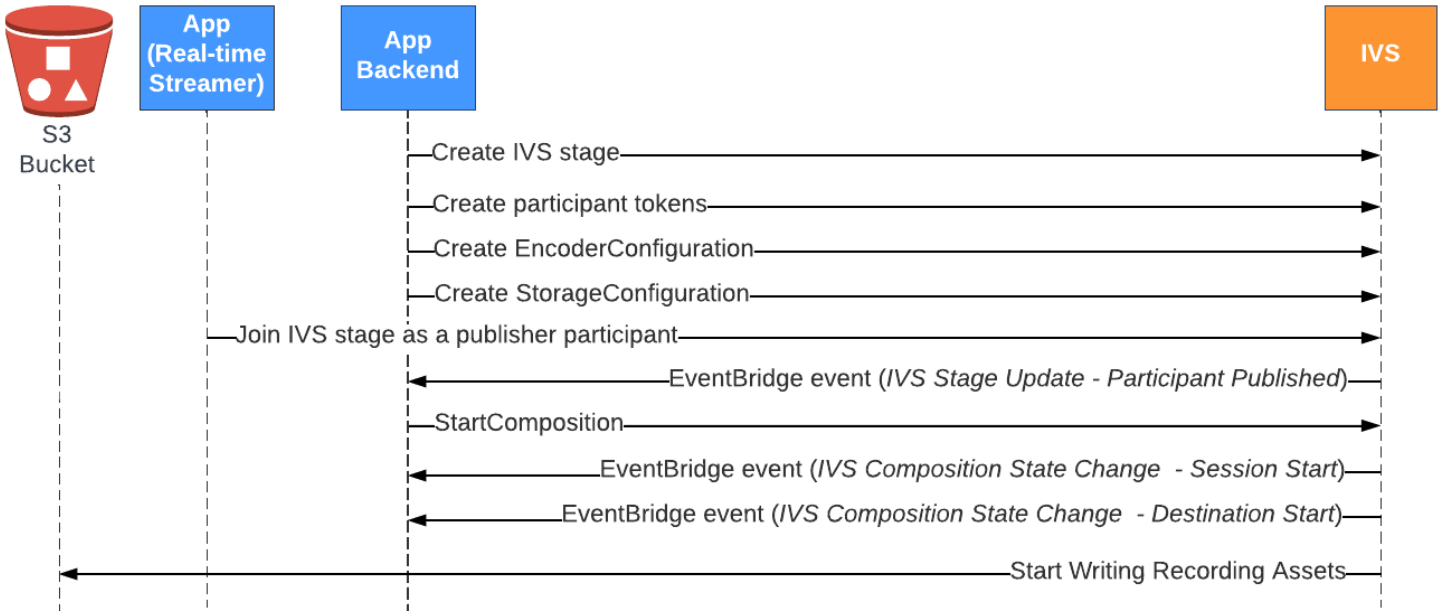
Voraussetzungen

Um die zusammengesetzte Aufzeichnung zu verwenden, benötigen Sie eine Stage mit aktiven Publishern und einen S3-Bucket, um ihn als Aufzeichnungsziel zu verwenden. Nachfolgend wird ein möglicher Workflow beschrieben, der EventBridge-Ereignisse verwendet, um eine Zusammensetzung in einem S3-Bucket aufzuzeichnen. Alternativ können Sie Zusammensetzungen basierend auf Ihrer eigenen App-Logik starten und stoppen.

1. Erstellen Sie eine IVS-Stage und Teilnehmer-Token für jeden Publisher.
2. Erstellen Sie eine EncoderConfiguration (ein Objekt, das angibt, wie das aufgenommene Video gerendert werden soll).
3. Erstellen Sie einen S3-Bucket und eine StorageConfiguration (in der die Aufzeichnungsinhalte gespeichert werden).

Wichtig: Wenn Sie einen vorhandenen S3-Bucket verwenden, muss die Einstellung Objekteigentümerschaft entweder Bucket-Eigentümer erzwungen oder Bucket-Eigentümer bevorzugt sein. Einzelheiten finden Sie in der S3-Dokumentation zum Steuern der Eigentümerschaft von Objekten.

4. Treten Sie der Stage bei und veröffentlichen Sie dort.
5. Wenn Sie ein vom Teilnehmer veröffentlichtes EventBridge-Ereignis erhalten, rufen Sie StartComposition mit einem S3 DestinationConfiguration-Objekt als Ziel auf
6. Nach einigen Sekunden sollten Sie sehen können, dass die HLS-Segmente in Ihren S3-Buckets beibehalten werden.



Hinweis: Eine Zusammensetzung wird nach 60 Sekunden Inaktivität der Publisher-Teilnehmer in der Stage automatisch heruntergefahren. An diesem Punkt wird die Zusammensetzung beendet und sie geht in einen `STOPPED`-Status über. Eine Zusammensetzung wird nach einigen Minuten im `STOPPED`-Status automatisch gelöscht. Einzelheiten finden Sie unter Zusammensetzungslebenszyklus unter Serverseitige Zusammensetzung.

Beispiel für eine zusammengesetzte Aufzeichnung: StartComposition mit einem S3-Bucket als Ziel

Das folgende Beispiel zeigt einen typischen Aufruf des Vorgangs StartComposition, bei dem S3 als einziges Ziel für die Zusammensetzung angegeben wird. Sobald die Zusammensetzung in

einen bestimmten `ACTIVE`-Status übergeht, werden Videosegmente und Metadaten in den durch das `storageConfiguration`-Objekt angegebenen S3-Bucket geschrieben. Informationen zum Erstellen von Zusammensetzungen mit unterschiedlichen Layouts finden Sie unter „Layouts“ im Abschnitt Serverseitige Zusammensetzung und in der API-Referenz zu IVS-Echtzeit-Streaming.

Anforderung

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

{
   "destinations": [
      {
         "s3": {
            "encoderConfigurationArns": [
              "arn:aws:ivs:ap-northeast-1:927810967299:encoder-configuration/
PAAwglkRtjge"
            ],
            "storageConfigurationArn": "arn:aws:ivs:ap-
northeast-1:927810967299:storage-configuration/ZBcEbgbE24Cq",
     "thumbnailConfigurations": [
        {
    "storage": ["LATEST", "SEQUENTIAL"],
    "targetIntervalSeconds": 30
               }
     ]
  }
      }
   ],
   "idempotencyToken": "db1i782f1g9",
   "stageArn": "arn:aws:ivs:ap-northeast-1:927810967299:stage/WyGkzNFGwiwr"
}
```

Antwort

```
{
    "composition": {
        "arn": "arn:aws:ivs:ap-northeast-1:927810967299:composition/s2AdaGUbvQgp",
        "destinations": [
            {
                "configuration": {
                    "name": "",
```

```
                    "s3": {
                        "encoderConfigurationArns": [
                            "arn:aws:ivs:ap-northeast-1:927810967299:encoder-
configuration/PAAwglkRtjge"
                        ],
                        "recordingConfiguration": {
                            "format": "HLS"
                        },
                        "storageConfigurationArn": "arn:aws:ivs:ap-
northeast-1:927810967299:storage-configuration/ZBcEbgbE24Cq",
                 "thumbnailConfigurations": [
                    {
                "storage": ["LATEST", "SEQUENTIAL"],
                "targetIntervalSeconds": 30
                          }
                 ]
                    }
                },
                "detail": {
                    "s3": {
                        "recordingPrefix": "MNALAcH9j2EJ/s2AdaGUbvQgp/2pBRKrNgX1ff/
composite"
                    }
                },
                "id": "2pBRKrNgX1ff",
                "state": "STARTING"
            }
        ],
        "layout": null,
        "stageArn": "arn:aws:ivs:ap-northeast-1:927810967299:stage/WyGkzNFGwiwr",
        "startTime": "2023-11-01T06:25:37Z",
        "state": "STARTING",
        "tags": {}
    }
}
```

Das in der StartComposition-Antwort vorhandene `recordingPrefix`-Feld kann verwendet werden, um zu bestimmen, wo die Aufnahmeinhalte gespeichert werden.

Inhalte der Aufnahme

Wenn die Zusammensetzung in einen `ACTIVE`-Status übergeht, werden HLS-Videosegmente und Thumbnails in den S3-Bucket geschrieben, der beim Aufruf von StartComposition bereitgestellt wurde. Dieser Inhalt ist für die Nachbearbeitung oder Wiedergabe als On-Demand-Video verfügbar.

Beachten Sie, dass nach dem Liveschalten einer Zusammensetzung das Ereignis „IVS Composition State Change“ ausgegeben wird und es einige Zeit dauern kann, bis die Manifestdateien, Videosegmente und Thumbnails geschrieben werden. Es wird empfohlen, aufgezeichnete Streams erst wiederzugeben oder zu verarbeiten, nachdem das Ereignis „IVS Composition State Change (Session End)“ empfangen wurde. Einzelheiten finden Sie unter Verwenden von EventBridge mit IVS-Echtzeit-Streaming.

Nachfolgend finden Sie eine Beispielverzeichnisstruktur und den Inhalt einer Aufzeichnung einer Live-IVS-Sitzung:

```
MNALAcH9j2EJ/s2AdaGUbvQgp/2pBRKrNgX1ff/composite
   events
      recording-started.json
      recording-ended.json
   media
      hls
      thumbnails
      latest_thumbnail
```

Der Ordner `events` enthält die Metadatendateien, die dem Aufzeichnungsereignis entsprechen. JSON-Metadatendateien werden generiert, wenn die Aufzeichnung gestartet, erfolgreich beendet oder mit Fehlern beendet wird:

- `events/recording-started.json`
- `events/recording-ended.json`
- `events/recording-failed.json`

Ein gegebener `events`-Ordner enthält `recording-started.json` und entweder `recording-ended.json` oder `recording-failed.json`.

Diese enthalten Metadaten, die sich auf die aufgezeichnete Sitzung und ihre Ausgabeformate beziehen. JSON-Details sind unten angegeben.

Der Ordner `media` enthält die unterstützten Medieninhalte. Der Unterordner `hls` enthält alle Medien und Manifestdateien, die während der Zusammensetzungssitzung generiert wurden, und kann mit dem IVS-Player abgespielt werden. Das HLS-Manifest befindet sich im Ordner `multivariant.m3u8`. Falls konfiguriert, enthalten die Ordner `thumbnails` und die Unterordner `latest_thumbnail` JPEG-Thumbnail-Mediendateien, die während der Zusammenstellungssitzung generiert wurden.

Bucket-Richtlinie für StorageConfiguration

Wenn ein StorageConfiguration-Objekt erstellt wird, erhält IVS Zugriff zum Schreiben von Inhalten in den angegebenen S3-Bucket. Dieser Zugriff wird durch Änderungen an der Richtlinie des S3-Buckets gewährt. Wenn die Richtlinie für den Bucket so geändert wird, dass der Zugriff von IVS aufgehoben wird, schlagen laufende und neue Aufzeichnungen fehl.

Das folgende Beispiel zeigt eine S3-Bucket-Richtlinie, die IVS das Schreiben in den S3-Bucket ermöglicht:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CompositeWrite-y1d212y",
            "Effect": "Allow",
            "Principal": {
                "Service": "ivs-composite.ap-northeast-1.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::my-s3-bucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control"
                },
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
```

```
        }
    ]
}
```

JSON-Metadatendateien

Diese Metadaten weisen das JSON-Format auf. Es enthält die folgenden Informationen:

Feld	Typ	Erforderlich	Beschreibung
`stage_arn`	Zeichenfolge	Ja	ARN der Stage, die als Quelle für die Zusammensetzung verwendet wird.
`media`	object	Ja	Objekt, das die Aufzählungsobjekte von Medieninhalten enthält, die für diese Aufzeichnung verfügbar sind. Zulässige Werte: `"hls"`.
`hls`	object	Ja	Aufzählungsfeld, das die Ausgabe des Apple HLS-Formats beschreibt.
`duration_ms`	Ganzzahl	Bedingt	Dauer des aufgezeichneten HLS-Inhalts in Millisekunden. Dies ist nur verfügbar, wenn `recording_status "RECORDING_ENDED"` oder `"RECORDING_ENDED_WITH_FAILURE"` ist. Wenn ein Fehler aufgetreten ist, bevor eine Aufzeichnung durchgeführt wurde, ist dies 0.
`path`	Zeichenfolge	Ja	Relativer Pfad vom S3-Präfix, in dem HLS-Inhalt gespeichert wird.

Feld	Typ	Erforderlich	Beschreibung
`playlist`	Zeichenfolge	Ja	Name der HLS-Master-Wiedergabeliste.
`renditions`	Objekt	Ja	Array von Formatvarianten (HLS-Variante) von Metadatenobjekten. Es ist immer mindestens eine Formatvariante vorhanden.
`path`	Zeichenfolge	Ja	Relativer Pfad vom S3-Präfix, in dem HLS-Inhalt für diese Formatvariante gespeichert wird.
`playlist`	Zeichenfolge	Ja	Name der Medienwiedergabeliste für diese Formatvariante.
`resolution_height`	int	Bedingt	Pixelauflösungshöhe des codierten Videos. Diese Option ist nur verfügbar, wenn die Formatvariante eine Videospur enthält.
`resolution_width`	int	Bedingt	Pixelauflösungsbreite des codierten Videos. Diese Option ist nur verfügbar, wenn die Formatvariante eine Videospur enthält.
`thumbnails`	object	Bedingt	Aufzählungsfeld, das die Ausgabe von Miniaturansichten beschreibt. Diese Funktion ist nur verfügbar, wenn das Feld der Konfiguration `storage SEQUENTIAL` enthält.
`path`	Zeichenfolge	Ja	Relativer Pfad vom S3-Präfix, in dem sequentieller Thumbnail-Inhalt gespeichert wird.

Feld	Typ	Erforderlich	Beschreibung
`resolutions`	object	Ja	Array von Auflösungen (Thumbnail-Varianten) von Metadatenobjekten. Es ist immer mindestens eine Auflösung vorhanden.
`path`	Zeichenfolge	Ja	Relativer Pfad vom S3-Präfix, in dem Thumbnail-Inhalt für diese Auflösung gespeichert wird.
`resolution_height`	int	Ja	Pixelauflösungshöhe des Thumbnails.
`resolution_width`	int	Ja	Pixelauflösungsbreite des Thumbnails.
`latest_thumbnail`	object	Bedingt	Aufzählungsfeld, das die Ausgabe von Miniaturansichten beschreibt. Diese Funktion ist nur verfügbar, wenn das Feld der Konfiguration `storage` `LATEST` enthält.
`path`	Zeichenfolge	Ja	Relativer Pfad vom S3-Präfix, in dem `latest_thumbnail` gespeichert wird.
`resolutions`	object	Ja	Array von Auflösungen (Thumbnail-Varianten) von Metadatenobjekten. Es ist immer mindestens eine Auflösung vorhanden.
`path`	Zeichenfolge	Ja	Relativer Pfad vom S3-Präfix, in dem das aktuelle Thumbnail für diese Auflösung gespeichert wird.
`resolution_height`	int	Ja	Pixelauflösungshöhe des aktuellen Thumbnails.

Feld	Typ	Erforderlich	Beschreibung
`resolution_width`	int	Ja	Pixelauflösungsbreite des aktuellen Thumbnails.
`recording_ended_at`	Zeichenfolge	Bedingt	RFC 3339 UTC-Zeitstempel, wenn die Aufnahme beendet wurde. Dies ist nur verfügbar, wenn `recording_status` `"RECORDING_ENDED"` oder `"RECORDING_ENDED_WITH_FAILURE"` ist. `recording_started_at` und `recording_ended_at` sind Zeitstempel, wenn diese Ereignisse generiert werden, und stimmen möglicherweise nicht genau mit den Zeitstempeln des HLS-Videosegments überein. Um die Dauer einer Aufnahme genau zu bestimmen, verwenden Sie das Feld `duration_ms`.
`recording_started_at`	Zeichenfolge	Bedingt	RFC 3339 UTC-Zeitstempel, wenn die Aufnahme gestartet wurde. Dies ist nicht verfügbar, wenn sich der `recording_status` in `RECORDING_START_FAILED` befindet. Beachten Sie den oben stehenden Hinweis zu `recording_ended_at`.

Feld	Typ	Erforderlich	Beschreibung
`recording_status`	Zeichenfolge	Ja	Aufzeichnungsstatus. Zulässige Werte: `"RECORDING_STARTED"`, `"RECORDING_ENDED"`, `"RECORDING_START_FAILED"`, `"RECORDING_ENDED_WITH_FAILURE"`.
`recording_status_message`	Zeichenfolge	Bedingt	Beschreibende Informationen über den Status. Dies ist nur verfügbar, wenn `recording_status` `"RECORDING_ENDED"` oder `"RECORDING_ENDED_WITH_FAILURE"` ist.
`version`	Zeichenfolge	Ja	Die Version des Metadatenschemas.

Beispiel: recording-started.json

```
{
  "version": "v1",
  "stage_arn": "arn:aws:ivs:ap-northeast-1:123456789012:stage/aAbBcCdDeE12",
  "recording_started_at": "2023-11-01T06:01:36Z",
  "recording_status": "RECORDING_STARTED",
  "media": {
    "hls": {
      "path": "media/hls",
      "playlist": "multivariant.m3u8",
      "renditions": [
        {
          "path": "720p30-abcdeABCDE12",
          "playlist": "playlist.m3u8",
          "resolution_width": 1280,
          "resolution_height": 720
        }
      ]
```

```
    },
    "thumbnails": {
      "path": "media/thumbnails",
      "resolutions": [
        {
          "path": "1280x720",
          "resolution_width": 1280,
          "resolution_height": 720
        }
      ]
    },
    "latest_thumbnail": {
      "path": "media/latest_thumbnail",
      "resolutions": [
        {
          "path": "1280x720",
          "resolution_width": 1280,
          "resolution_height": 720
        }
      ]
    }
  }
}
```

Beispiel: recording-ended.json

```
{
  "version": "v1",
  "stage_arn": "arn:aws:ivs:ap-northeast-1:123456789012:stage/aAbBcCdDeE12",
  "recording_started_at": "2023-10-27T17:00:44Z",
  "recording_ended_at": "2023-10-27T17:08:24Z",
  "recording_status": "RECORDING_ENDED",
  "media": {
    "hls": {
      "duration_ms": 460315,
      "path": "media/hls",
      "playlist": "multivariant.m3u8",
      "renditions": [
        {
          "path": "720p30-abcdeABCDE12",
          "playlist": "playlist.m3u8",
          "resolution_width": 1280,
          "resolution_height": 720
```

```
        }
      ]
    },
    "thumbnails": {
      "path": "media/thumbnails",
      "resolutions": [
        {
          "path": "1280x720",
          "resolution_width": 1280,
          "resolution_height": 720
        }
      ]
    },
    "latest_thumbnail": {
      "path": "media/latest_thumbnail",
      "resolutions": [
        {
          "path": "1280x720",
          "resolution_width": 1280,
          "resolution_height": 720
        }
      ]
    }
  }
}
```

Beispiel: 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
        }
      ]
    },
    "thumbnails": {
      "path": "media/thumbnails",
      "resolutions": [
        {
          "path": "1280x720",
          "resolution_width": 1280,
          "resolution_height": 720
        }
      ]
    },
    "latest_thumbnail": {
      "path": "media/latest_thumbnail",
      "resolutions": [
        {
          "path": "1280x720",
          "resolution_width": 1280,
          "resolution_height": 720
        }
      ]
    }
  }
}
```

Wiedergabe von aufgezeichneten Inhalten aus privaten Buckets

Standardmäßig sind die aufgezeichneten Inhalte privat. Aus diesem Grund ist die Wiedergabe dieser Objekte über die direkte S3-URL nicht möglich. Wenn Sie versuchen, die multivariate HLS-Wiedergabeliste (m3u8-Datei) zur Wiedergabe mit dem IVS-Player oder einem anderen Player zu öffnen, erhalten Sie eine Fehlermeldung (z. B. „Sie haben keine Berechtigung zum Zugriff auf die angeforderte Ressource“). Stattdessen können Sie diese Dateien mit dem Amazon-CloudFront-CDN (Content Delivery Network) wiedergeben.

Ihre CloudFront-Verteilungen können so konfiguriert werden, dass Inhalt von privaten Buckets bereitgestellt wird. In der Regel ist dies vorzuziehen, offen zugängliche Buckets zu haben, in denen Lesevorgänge die von CloudFront angebotenen Steuerelemente umgehen. Ihre Verteilung kann für den Service von einem privaten Bucket aus eingerichtet werden, indem Sie eine Ursprungs-Zugriffskontrolle erstellen. Dabei handelt es sich um einen speziellen CloudFront-Benutzer, der über

Leseberechtigungen für den privaten Ursprungs-Bucket verfügt. Sie können die OAC erstellen, wenn Sie Ihre Verteilung erstellen oder sie anschließend über die CloudFront-Konsole oder API hinzufügen. Weitere Informationen finden Sie unter Erstellen einer neuen Ursprungs-Zugriffskontrolle im Amazon CloudFront-Entwicklerhandbuch.

Einrichten der Wiedergabe mithilfe von CloudFront mit aktiviertem CORS

In diesem Beispiel wird gezeigt, wie ein Entwickler eine CloudFront-Verteilung mit aktiviertem CORS einrichten kann, um die Wiedergabe seiner Aufzeichnungen von jeder Domain aus zu ermöglichen. Dies ist besonders während der Entwicklungsphase nützlich. Sie können das folgende Beispiel jedoch an Ihre Produktionsanforderungen anpassen.

Schritt 1: S3-Bucket erstellen

Erstellen Sie einen S3-Bucket, der zum Speichern der Aufzeichnungen verwendet wird. Beachten Sie, dass sich der Bucket in derselben Region befinden muss, die Sie für Ihren IVS-Workflow verwenden.

Fügen Sie dem Bucket eine CORS-Berechtigungserichtlinie hinzu:

1. Navigieren Sie in der AWS-Konsole zur Registerkarte S3-Bucket-Berechtigungen.
2. Kopieren Sie die untenstehende CORS-Richtlinie und fügen Sie sie unter Cross-Origin Resource Sharing (CORS) ein. Dadurch wird der CORS-Zugriff auf den S3-Bucket aktiviert.

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE",
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": [
            "x-amz-server-side-encryption",
            "x-amz-request-id",
```

```
            "x-amz-id-2"
        ]
    }
]
```

Schritt 2: Eine CloudFront-Verteilung erstellen

Weitere Informationen finden Sie unter Erstellen einer CloudFront-Verteilung im CloudFront-Entwicklerhandbuch.

Geben Sie über die AWS-Konsole die folgenden Informationen ein:

Für dieses Feld …	Wählen Sie dies ...
Ursprungs-Domain	Der im vorherigen Schritt erstellte S3-Bucket
Ursprungszugriff	Einstellungen für die Ursprungs-Zugriffs kontrolle (empfohlen) unter Verwendung von Standardparametern
Standard-Cache-Verhalten: Viewer-Protokollrichtlinie	Redirect HTTP to HTTPS
Standard-Cache-Verhalten: Zulässige HTTP-Methoden	GET, HEAD und OPTIONS
Standard-Cache-Verhalten: Cache-Schlüssel- und Ursprungsanfragen	CachingDisabled-Richtlinie
Standard-Cache-Verhalten: Ursprungs-Anforderungsrichtlinie	CORS-S3Origin
Standard-Cache-Verhalten: Richtlinie für Antwort-Header	SimpleCORS
Webanwendungs-Firewall	Aktivieren von Sicherheitsmaßnahmen

Speichern Sie dann die CloudFront-Verteilung.

Schritt 3: Einrichten der S3-Bucket-Richtlinie

1. Löschen Sie jede StorageConfiguration, die Sie für den S3-Bucket eingerichtet haben. Dadurch werden alle Bucket-Richtlinien entfernt, die beim Erstellen der Richtlinie für diesen Bucket automatisch hinzugefügt wurden.
2. Navigieren Sie zu Ihrer CloudFront-Verteilung, stellen Sie sicher, dass sich alle Verteilungsfelder in den im vorherigen Schritt definierten Status befinden, und Kopieren Sie die Bucket-Richtlinie (verwenden Sie die Schaltfläche Richtlinie kopieren).
3. Navigieren Sie zu Ihrem S3-Bucket. Wählen Sie auf der Registerkarte Berechtigungen die Option Bucket-Richtlinie bearbeiten aus und fügen Sie die Bucket-Richtlinie ein, die Sie im vorherigen Schritt kopiert haben. Nach diesem Schritt sollte die Bucket-Richtlinie ausschließlich die CloudFront-Richtlinie enthalten.
4. Erstellen Sie eine StorageConfiguration unter Angabe des S3-Buckets.

Nachdem die StorageConfiguration erstellt wurde, sehen Sie zwei Elemente in der S3-Bucket-Richtlinie: eines erlaubt CloudFront das Lesen von Inhalten und ein anderes erlaubt IVS das Schreiben von Inhalten. Ein Beispiel für eine endgültige Bucket-Richtlinie mit CloudFront- und IVS-Zugriff wird in Beispiel: S3-Bucket-Richtlinie mit CloudFront- und IVS-Zugriff gezeigt.

Schritt 4: Wiedergabe von Aufnahmen

Nachdem Sie die CloudFront-Verteilung erfolgreich eingerichtet und die Bucket-Richtlinie aktualisiert haben, sollten Sie Aufzeichnungen mit dem IVS-Player wiedergeben können:

1. Starten Sie erfolgreich eine Zusammensetzung und stellen Sie sicher, dass eine Aufzeichnung im S3-Bucket gespeichert ist.
2. Nachdem Sie die Schritte 1 bis Schritt 3 in diesem Beispiel ausgeführt haben, sollten die Videodateien für die Nutzung über die CloudFront-URL verfügbar sein. Ihre CloudFront-URL ist der Domainname für die Verteilung auf der Registerkarte Einzelheiten in der Amazon-CloudFront-Konsole. Sie ist in etwa wie folgt:

   ```
   a1b23cdef4ghij.cloudfront.net
   ```
3. Um das aufgezeichnete Video über die CloudFront-Verteilung wiederzugeben, suchen Sie den Objektschlüssel für Ihre `multivariant.m3u8`-Datei im S3-Bucket. Sie ist in etwa wie folgt:

   ```
   FDew6Szq5iTt/9NIpWJHj0wPT/fjFKbylPb3k4/composite/media/hls/
   multivariant.m3u8
   ```

4. Hängen Sie den Objektschlüssel an das Ende Ihrer CloudFront-URL an. Ihre endgültige URL sieht etwa folgendermaßen aus:

   ```
   https://a1b23cdef4ghij.cloudfront.net/FDew6Szq5iTt/9NIpWJHj0wPT/
   fjFKbylPb3k4/composite/media/hls/multivariant.m3u8
   ```

5. Sie können jetzt die endgültige URL zum Quellattribut eines IVS-Players hinzufügen, um die vollständige Aufzeichnung anzusehen. Um das aufgezeichnete Video anzusehen, können Sie die Demo unter Erste Schritte im IVS Player SDK: Web Guide verwenden.

Beispiel: S3-Bucket-Richtlinie mit CloudFront und IVS-Zugriff

Der folgende Ausschnitt veranschaulicht eine S3-Bucket-Richtlinie, die es CloudFront ermöglicht, Inhalte in den privaten Bucket zu lesen und IVS-Inhalte in den Bucket zu schreiben. Hinweis: Kopieren Sie den unten stehenden Ausschnitt nicht und fügen Sie ihn nicht in Ihren eigenen Bucket ein. Ihre Richtlinie sollte die IDs enthalten, die für Ihre CloudFront-Verteilung und Speicherkonfiguration relevant sind.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CompositeWrite-7eiKaIGkC9DO",
      "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"
        }
      }
    }
  ]
}
```

Fehlerbehebung

- Die Zusammensetzung wird nicht in den S3-Bucket geschrieben – Stellen Sie sicher, dass der S3-Bucket und die StorageConfiguration-Objekte erstellt werden und sich in derselben Region befinden. Stellen Sie außerdem sicher, dass IVS Zugriff auf den Bucket hat, indem Sie Ihre Bucket-Richtlinie überprüfen. Weitere Informationen finden Sie unter Bucket-Richtlinie für die Speicherkonfiguration.
- Ich kann beim Ausführen von ListCompositions keine Zusammensetzung finden – Zusammensetzungen sind kurzlebige Ressourcen. Sobald diese in einen endgültigen Status übergehen, werden sie nach einigen Minuten automatisch gelöscht.
- Meine Zusammensetzung hält automatisch an – Eine Zusammensetzung stoppt automatisch, wenn sich mehr als 60 Sekunden lang kein Publisher in der Stage befindet.

Bekanntes Problem

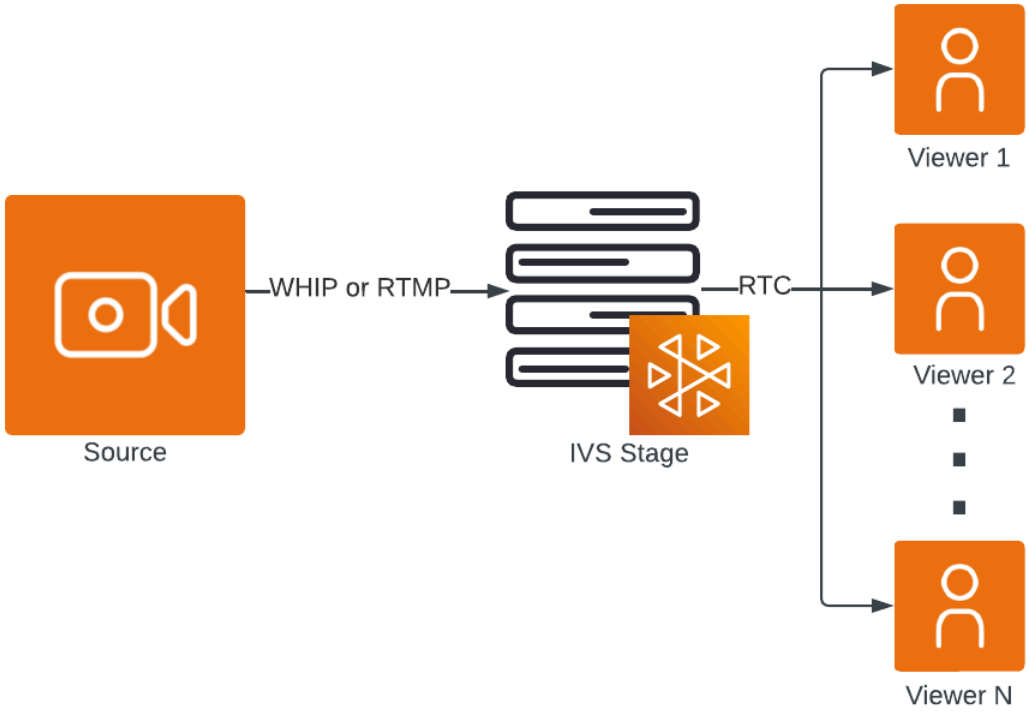
Die von der zusammengesetzten Aufhzeichnung geschriebene Medien-Wiedergabeliste erhält das Tag `#EXT-X-PLAYLIST-TYPE:EVENT`, während die Zusammensetzung läuft. Wenn die Zusammensetzung abgeschlossen ist, wird das Tag auf `#EXT-X-PLAYLIST-TYPE:VOD` aktualisiert.

Für ein reibungsloses Wiedergabeerlebnis empfiehlt es sich, diese Wiedergabeliste erst zu verwenden, nachdem die Zusammensetzung erfolgreich abgeschlossen wurde.

IVS-Streamerfassung | Echtzeit-Streaming

Als Alternative zur Verwendung des IVS-Broadcast-SDK können Sie Videos aus einer WHIP- oder RTMP-Quelle auf einer IVS-Bühne veröffentlichen. Dieser Ansatz bietet Flexibilität für Workflows, bei denen die Verwendung des SDK nicht möglich oder nicht erwünscht ist, z. B. bei der Veröffentlichung von Videos aus OBS Studio oder einem Hardware-Encoder. Wir empfehlen, wann immer möglich, die Verwendung des IVS-Broadcast-SDK, da wir die Leistung oder Kompatibilität von Drittanbieterlösungen mit IVS nicht garantieren können.

Dieses Diagramm zeigt, wie das Veröffentlichen mit WHIP und RTMP funktioniert:



Unterstützte Protokolle

IVS-Echtzeit-Streaming unterstützt mehrere Erfassungsprotokolle:

- RTMP und RTMPS – RTMP (Real-Time Messaging Protocol) ist der Branchenstandard für die Übertragung von Videos über ein Netzwerk. Bei RTMPS handelt es sich um die sichere Version von RTMP, die über TLS ausgeführt wird.

IVS unterstützt die Multitrack-Video-Funktion von E-RTMP (Enhanced RTMP). Weitere Informationen finden Sie im Abschnitt E-RTMP-Multitrack-Videos in der Dokumentation zur IVS-RTMP-Veröffentlichung.

- WHIP (WebRTC-HTTP Ingestion Protocol) – ein IETF-Entwurf, der zur Standardisierung der WebRTC-Erfassung entwickelt wurde

Eine ausführliche Anleitung zur Verwendung dieser Protokolle finden Sie in unserer RTMP- und WHIP-Dokumentation.

Unterstützte Medienspezifikationen

- Audioeingabeformat
  - Codec: AAC-LC für RTMP und Opus für WHIP
  - Kanäle: 2 (Stereo) oder 1 (Mono)
  - Abtastrate: 44,1 kHz oder 48 kHz
  - Maximale Bitrate: 160 kbit/s
- Videoeingabeformat
  - Codec: H.264
  - H.264-Profil: Baseline
  - IDR-Intervall: 1 oder 2 Sekunden
  - Bildrate: 10 bis 60 FPS
  - B-Frames: 0

    Hinweis: Im IVS-Broadcast-SDK sind B-Frames standardmäßig aktiviert, doch ab Version 1.25.0 werden B-Frames bei der Übertragung zu einer IVS-Bühne automatisch deaktiviert. Für das Echtzeit-Streaming mit anderen RTMP-Encodern müssen Entwickler B-Frames deaktivieren. Wenn Entwickler, die andere RTMP-Encoder nutzen, B-Frames nicht deaktivieren, werden ihre Streams getrennt.
  - Auflösung: Maximum: 720p. Minimum: 160p
  - Maximale Bitrate: 8,5 Mbit/s

    Hinweis: Bei Einzeltrack-RTMP-Streams gilt dieses Limit für den betreffenden Track. Bei Multitrack-Videos, die mit Enhanced RTMP veröffentlicht werden, gilt das Limit für die kombinierte Bitrate aller Videotracks.

- Encoder-Konfiguration: Wir empfehlen die Verwendung der Einstellungen `veryfast` und `zerolatency` für einen H.264-Encoder. Außerdem: Die Option `sliced_threads` x264 ist in den `zerolatency`-Voreinstellungen enthalten, und wir empfehlen, sie zu deaktivieren. Wenn Sie beispielsweise FFmpeg verwenden, sollte Ihr Befehl Folgendes beinhalten: `-preset:v veryfast -tune zerolatency -x264-params sliced-threads=0`

IVS-RTMP-Veröffentlichung | Streaming

In diesem Dokument wird der Prozess der Veröffentlichung für eine IVS-Stage mithilfe von RTMP beschrieben. Weitere Informationen zu den verschiedenen Aufzeichnungsoptionen finden Sie in der Dokumentation zu Streamerfassung.

Voraussetzungen

Erstellen einer Stage

Verwenden Sie den folgenden Befehl, um eine Stage zu erstellen:

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

Einzelheiten, einschließlich der Antwort, finden Sie unter CreateStage.

Wichtig: Achten Sie in der Antwort auf das Feld `endpoints`, in dem sowohl RTMP- als auch RTMPS-Endpunkte aufgeführt sind. Diese sind für die Einrichtung Ihres RTMP-Encoders erforderlich.

Erfassungskonfiguration erstellen

Für die Veröffentlichung in einer Stage mithilfe von RTMP müssen Sie zunächst eine Erfassungskonfiguration erstellen und diese Ihrer Stage zuordnen. Wenn Sie auf der Stage veröffentlichen (mithilfe des Stream-Schlüssels aus der Aufzeichnungskonfiguration und des RTMP-Endpunkts der Stage), werden die Medien als Teilnehmer auf der Stage veröffentlicht. Sie haben die Möglichkeit, eine `userId` und benutzerdefinierte `attributes` anzugeben, die mit dem Teilnehmer verknüpft werden, der sich mit der Stage verbindet.

```
aws ivs-realtime create-ingest-configuration \
  --name 'test' \
  --stage-arn arn:aws:ivs:us-east-1:123456789012:stage/8faHz1SQp0ik \
  --user-id '123' \
  --ingest-protocol 'RTMPS'
```

Einzelheiten, einschließlich der Antwort, finden Sie unter CreateIngestConfiguration.

Wenn Sie eine Erfassungskonfiguration erstellen, können Sie diese im Voraus einem bestimmten Stage-ARN zuordnen. Ohne diese Zuordnung ist der Stream-Schlüssel unbrauchbar. Außerdem können Erfassungskonfigurationen (einschließlich des Felds `stageArn`) über den Vorgang UpdateIngestConfiguration aktualisiert werden, sodass Sie dieselbe Konfiguration für verschiedene Stages wiederverwenden können.

Hinweis: Das Erfassungskonfigurationsfeld `insecureIngest` ist standardmäßig auf `false` festgelegt, was die Verwendung von RTMPS erfordert. RTMP-Verbindungen werden abgelehnt. Wenn Sie RTMP verwenden müssen, legen Sie `insecureIngest` auf `true` fest. Außer für bestimmte und verifizierte Anwendungsfälle, für die RTMP zwingend erforderlich ist, empfiehlt sich die Nutzung von RTMPS.

RTMP-Einzeltrack-Videos

Nachfolgend wird die Verwendungsweise von OBS Studio beschrieben. Sie können jedoch jeden RTMP-Encoder verwenden, der den Medienspezifikationen von IVS entspricht.

OBS-Leitfaden

1. Laden Sie die Software herunter und installieren Sie sie: https://obsproject.com/download.
2. Klicken Sie auf Settings (Einstellungen). Wählen Sie im Bereich Stream der Einstellungen in der Dropdownliste Service die Option Benutzerdefiniert aus.
3. Geben Sie für den Server den RTMP- oder RTMPS-Endpunkt der Stage ein.
4. Geben Sie für den Stream-Schlüssel den `streamKey` aus der Erfassungskonfiguration ein.
5. Konfigurieren Sie Ihre Videoeinstellungen wie gewohnt, mit einigen Einschränkungen:
   a. IVS-Echtzeit-Streaming unterstützt Eingaben bis zu 720p bei 8,5 Mbit/s. Wenn Sie einen dieser Grenzwerte überschreiten, wird Ihr Stream getrennt.
   b. Wir empfehlen, das Keyframe-Intervall im Bereich Ausgabe auf 1 oder 2 Sek. einzustellen. Ein niedriges Keyframe-Intervall ermöglicht es Zuschauern, die Videowiedergabe schneller zu starten. Wir empfehlen außerdem, die Voreinstellung für die CPU-Auslastung auf veryfast und die Einstellung Optimieren auf Latenz zerolatency einzustellen, um die niedrigste Latenz zu erzielen.
   c. Da OBS Simulcast nicht unterstützt, empfehlen wir, die Bitrate unter 2,5 Mbit/s zu halten. Dadurch können Zuschauer mit Verbindungen mit geringerer Bandbreite zuschauen.

d. Deaktivieren Sie B-Frames, da Streams mit B-Frames automatisch getrennt werden. Führen Sie eine der folgenden Aktionen aus:

   - Geben Sie in den x264-Optionen ein `bframes=0 sliced-threads=0`.
   - Legen Sie B-Frames auf 0 fest, wenn dies eine Option ist (z. B. für NVENC).

   Beachten Sie: RTMP-Streams müssen sowohl Audio- als auch Videospuren enthalten, da sie sonst getrennt werden.

6. Wählen Sie Streaming starten aus.

Wichtig: Wenn die maximale Bitrate Ihres Encoders auf 8,5 Mbit/s festgelegt ist, verschwindet der Publisher gelegentlich aus der Sitzung. Das liegt daran, dass es sich bei der Einstellung für die maximale Bitrate nur um ein Ziel handelt und Encoder das Ziel gelegentlich überschreiten. Um dies zu verhindern, stellen Sie die maximale Bitrate Ihres Encoders niedriger ein, z. B. auf 6 Mbit/s.

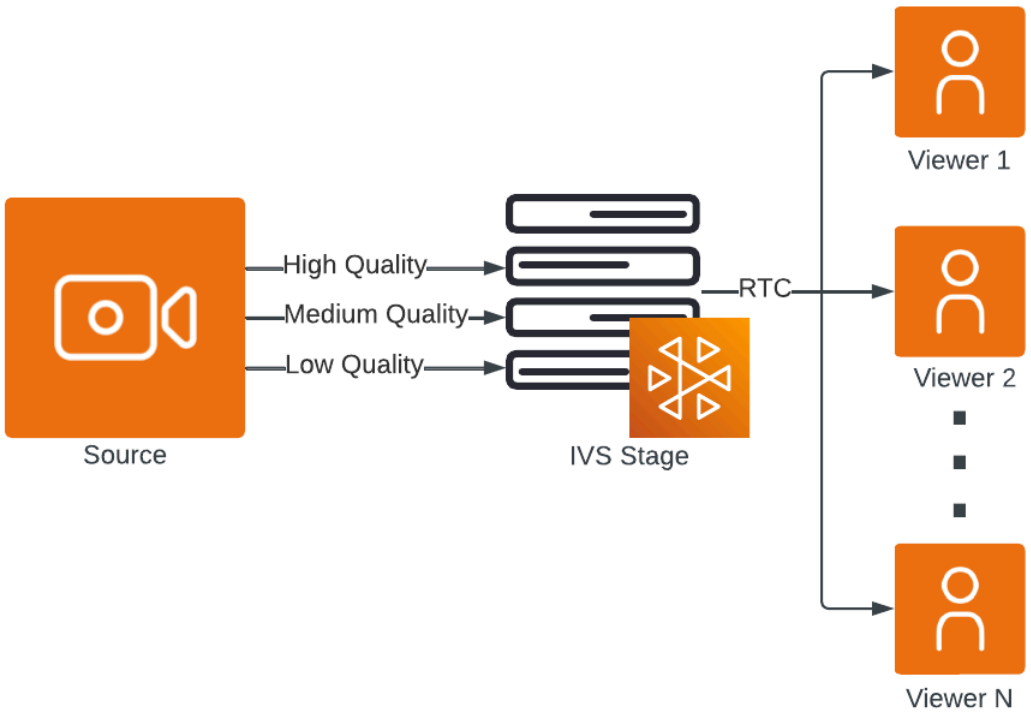
E-RTMP-Multitrack-Videos

IVS unterstützt die Multitrack-Video-Funktion von E-RTMP (Enhanced Real-Time Messaging Protocol), womit Sie Videos unterschiedlicher Qualität in einem einzigen RTMP-Stream auf der IVS-Stage veröffentlichen können. Dies ermöglicht das Streaming mit adaptiver Bitrate, sodass Abonnenten Videos automatisch in der für ihre jeweilige Netzwerkverbindung besten Qualität ansehen können.

Nach der Erfassung werden die Videos mit unterschiedlicher Qualität den Abonnenten als Simulcast-Layer zur Verfügung gestellt. Informationen zum Konfigurieren der Schichten, die von den Abonnenten empfangen werden, finden Sie in den Abschnitten unter „Mehrschichtige Kodierung mit Simulcast“ in den Broadcast-SDK-Anleitungen zum Echtzeit-Streaming: Android, iOS und Web.

Beispielcode finden Sie auf GitHub unter aws-samples/sample-amazon-ivs-multitrack-video.

Das folgende Diagramm zeigt, wie das Veröffentlichen mit Multitrack-Videos funktioniert:



OBS-Leitfaden

1. Laden Sie OBS Studio herunter und installieren Sie es:
   a. Windows: Multitrack-Videos werden ab OBS Studio 30.2 unterstützt.
   b. MacOS: Multitrack-Videos werden ab OBS Studio 31.1 Beta unterstützt (nur Apple Silicon).
   c. Das Programm kann unter folgender Adresse heruntergeladen werden: https://obsproject.com/download.
2. Klicken Sie auf Settings (Einstellungen). Wählen Sie im Bereich Stream der Einstellungen in der Dropdown-Liste Service die Option Amazon IVS aus.
3. Belassen Sie die Einstellung für den Server auf Auto.
4. Geben Sie für den Stream-Schlüssel den `streamKey` aus der Erfassungskonfiguration ein.
5. Aktivieren Sie im Abschnitt Multitrack-Video die Option Multitrack-Video aktivieren.
6. Stellen Sie im Bereich Video die gewünschte Grundauflösung (Canvas-Auflösung) und die gewünschte (skalierte) Ausgabeauflösung ein. IVS-Echtzeit-Streaming unterstützt Inputs von bis zu 720p. Bei Überschreiten dieses Limits wird der Stream getrennt.

Wenn Multitrack-Videos aktiviert sind, werden die Einstellungen wie die Anzahl der Videotracks, die Bitraten und das Keyframe-Intervall automatisch entsprechend den Leistungsmerkmalen des Geräts konfiguriert.

7. Wählen Sie Streamen starten aus.

Veröffentlichen mit FFmpeg

Sie können FFmpeg verwenden, um Live-Video und Audio als IVS-Echtzeit-Streaming über RTMP zu veröffentlichen. FFmpeg ist ein kostenloses Open-Source-Projekt, das eine umfangreiche Suite von Software-Bibliotheken für den Umgang mit Video-, Audio- und anderen Multimedia-Dateien und Streams umfasst.

Der folgende Beispielbefehl veröffentlicht einen Stream, der ein Farbmuster und einen Ton enthält:

```
ffmpeg \
 -re \
 -f lavfi -i testsrc=d=300:s=1280x720:r=60,format=yuv420p \
 -f lavfi -i sine=f=440:b=4:d=300 \
 -c:v libx264 \
 -b:v 2500k \
 -g 60 -bf 0 \
 -profile:v baseline \
 -preset veryfast \
 -tune zerolatency \
 -x264opts sliced-threads=0 \
 -c:a aac \
 -ac 2 \
 -b:a 160k \
 -ar 48000 \
 -f flv \
 rtmps://$INGEST_ENDPOINT/app/$STREAM_KEY
```

Ersetzen Sie in diesem Beispiel `$INGEST_ENDPOINT` und `$STREAM_KEY` durch Ihre eigenen Werte aus der IVS-Konsole oder API.

Diese Konfiguration erfüllt die unterstützten Medienspezifikationen für IVS-Echtzeit-Streaming, einschließlich H.264-Video (Basisprofil, keine B-Frames, keine geschnittenen Threads) und AAC-Audio.

Private Erfassung für Stage

Mithilfe eines Schnittstellen-VPC-Endpunkts können Sie aus Ressourcen in Ihrer Amazon-VPC oder in Direct Connect RTMP(S)- und E-RTMP(S)-Streams auf einer Stage veröffentlichen. Das ermöglicht eine private Verbindung zwischen der VPC und IVS, sodass Erfassungsverkehr im AWS-Netzwerk verbleibt. Informationen zum Einrichten und Konfigurieren eines Schnittstellen-VPC-Endpunkts für IVS finden Sie unter Private Erfassung in IVS im Benutzerhandbuch für IVS-Streaming mit niedriger Latenz.

IVS-WHIP-Veröffentlichung | Streaming

In diesem Dokument wird erklärt, wie Sie WHIP-kompatible Encoder wie OBS verwenden, um in IVS-Echtzeit-Streaming zu veröffentlichen. WHIP (WebRTC-HTTP Ingestion Protocol) ist ein IETF-Entwurf, der zur Standardisierung der WebRTC-Erfassung entwickelt wurde.

WHIP ermöglicht die Kompatibilität mit Software wie OBS und bietet eine Alternative (zum IVS-Broadcast-SDK) für Desktop-Publishing. Erfahrenere Streamer, die mit OBS vertraut sind, bevorzugen es möglicherweise aufgrund seiner erweiterten Produktionsfeatures wie Szenenübergänge, Audiomischung und Overlay-Grafiken. Dies bietet Entwicklern eine vielseitige Option: Verwenden Sie das Web-Broadcast-SDK von IVS für die direkte Veröffentlichung im Browser oder ermöglichen Sie Streamern, OBS auf ihrem Desktop zu verwenden, um leistungsstärkere Tools zu erhalten.

WHIP ist auch von Vorteil, wenn die Verwendung des IVS-Broadcast-SDK nicht möglich oder nicht erwünscht ist. Beispielsweise ist das IVS-Broadcast-SDK in Setups mit Hardware-Encodern möglicherweise keine Option. Wenn der Encoder jedoch WHIP unterstützt, können Sie trotzdem direkt vom Encoder in IVS veröffentlichen.

WHIP-Anforderungen:

- Ihr SDP-Angebot muss eine H.264-Videospur enthalten, selbst wenn Sie nur Ton veröffentlichen. Enthält das Angebot keine Videospur, wird die Verbindung abgelehnt.
- Der globale WHIP-Endpunkt (https://global.whip.live-video.net) gibt eine temporäre Umleitung (307) zurück. WHIP-Clients müssen 307-Umleitungen korrekt verarbeiten und Header in der umgeleiteten Anfrage persistieren, wie es die WHIP-Spezifikation verlangt.

OBS-Leitfaden

OBS unterstützt WHIP ab Version 30. Laden Sie zunächst OBS v30 oder neuer herunter: https://obsproject.com/.

Gehen Sie wie folgt vor, um mithilfe von OBS über WHIP für eine IVS-Bühne zu veröffentlichen:

1. Generieren Sie ein Teilnehmer-Token mit Veröffentlichungsfunktion. Bei WHIP ist ein Teilnehmer-Token ein Bearer-Token. Standardmäßig laufen Teilnehmer-Token nach 12 Stunden ab, Sie können die Dauer jedoch auf bis zu 14 Tage verlängern.
2. Klicken Sie auf Settings (Einstellungen). Wählen Sie im Bereich Stream der Einstellungen in der Dropdownliste Service die Option WHIP aus.
3. Geben Sie für den Server „https://global.whip.live-video.net“ ein.
4. Geben Sie für das Bearer-Token das Teilnehmer-Token ein, das Sie in Schritt 1 generiert haben.
5. Konfigurieren Sie Ihre Videoeinstellungen wie gewohnt, mit einigen Einschränkungen:
   a. IVS-Echtzeit-Streaming unterstützt Eingaben bis zu 720p bei 8,5 Mbit/s. Wenn Sie einen dieser Grenzwerte überschreiten, wird Ihr Stream getrennt.
   b. Wir empfehlen, das Keyframe-Intervall im Bereich Ausgabe auf 1 oder 2 Sek. einzustellen. Ein niedriges Keyframe-Intervall ermöglicht es Zuschauern, die Videowiedergabe schneller zu starten. Wir empfehlen außerdem, die Voreinstellung für die CPU-Auslastung auf veryfast und die Einstellung Optimieren auf Latenz zerolatency einzustellen, um die niedrigste Latenz zu erzielen.
   c. Da OBS Simulcast nicht unterstützt, empfehlen wir, die Bitrate unter 2,5 Mbit/s zu halten. Dadurch können Zuschauer mit Verbindungen mit geringerer Bandbreite zuschauen.
6. Drücken Sie auf Streaming starten.

Hinweis: Wir sind uns der Qualitätsprobleme (wie zeitweiliges Einfrieren von Videos) bewusst, die bei WHIP in OBS auftreten können. Diese treten typischerweise auf, wenn das Netzwerk des Broadcasters instabil ist. Wir empfehlen, WHIP in OBS zu testen, bevor Sie es für Produktions-Livestreams verwenden. Eine Senkung der Broadcast-Bitrate kann auch dazu beitragen, das Auftreten dieser Probleme zu verringern.

IVS-Teilnehmerreplikation | Echtzeit-Streaming

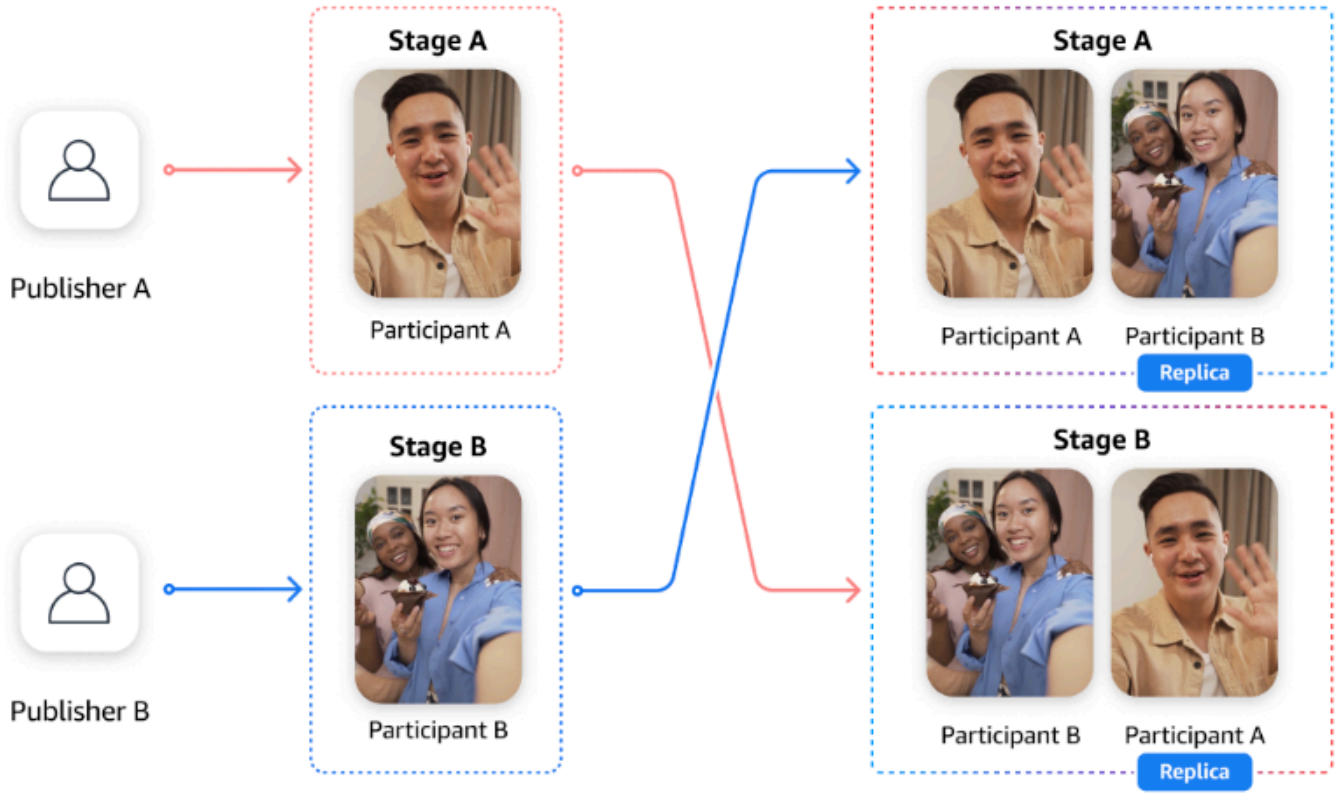
Mithilfe der Teilnehmerreplikation können Sie einen Teilnehmer von einer Bühne in eine andere kopieren. Das ist nützlich, wenn ein und derselbe Teilnehmer auf mehreren Bühnen gleichzeitig erscheinen soll, um bühnenübergreifende Interaktionen zu ermöglichen.

Ein häufiger Anwendungsfall für Live-Streaming-Anwendungen in sozialen Medien sind Wettbewerbe, oft auch als VS-Modus bezeichnet. Dabei werden zwei Streamer vorübergehend zusammengebracht, damit sie in Echtzeit miteinander interagieren können, während die Zuschauer jedes Streams beide Streamer sehen können.

Wichtige Konzepte:

- Quellbühne – Die Bühne, der der Teilnehmer ursprünglich beigetreten ist. Sie wird als Quelle für die Replikation genutzt.
- Zielbühne – Die Bühne, zu der der Teilnehmer repliziert wird.
- Replizierter Teilnehmer – Ein Teilnehmer in einer Bühne, der zu einer oder mehreren Zielbühnen repliziert wird.
- Replikatteilnehmer – Ein Teilnehmer in einer Zielbühne, der aus einer anderen Bühne (der Quellbühne) repliziert wurde.

Verwenden der Teilnehmerreplikation



Voraussetzungen

Für die Nutzung der Teilnehmerreplikation müssen mindestens zwei Bühnen erstellt worden sein. Im oben beschriebenen Szenario gibt es beispielsweise zwei aktive Publisher:

1. Teilnehmer A, verbunden mit Bühne A
2. Teilnehmer B, verbunden mit Bühne B

Wir werden Teilnehmer A vorübergehend zu Bühne B und Teilnehmer B zu Bühne A replizieren, um ein direktes Duell zu unterstützen.

Teilnehmerreplikation starten

Mit dem Vorgang StartParticipantReplication replizieren Sie Teilnehmer. Diesen Vorgang müssen Sie für jede Replikationsrichtung einmal aufrufen.

Teilnehmer A zu Bühne B replizieren:

```
aws ivs-realtime start-participant-replication \
  --source-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageA \
  --destination-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageB \
  --participant-id participant-a-id \
  --reconnect-window-seconds 10
```

Teilnehmer B zu Bühne A replizieren:

```
aws ivs-realtime start-participant-replication \
  --source-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageB \
  --destination-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageA \
  --participant-id participant-b-id \
  --reconnect-window-seconds 10
```

Nach dem Start der Replikation bleiben die Teilnehmer repliziert, bis Sie die Replikation mit dem Vorgang StopParticipantReplication explizit beenden. Ein replizierter Teilnehmer, der die Verbindung trennt und innerhalb des von `reconnectWindowSeconds` angegebenen Intervalls wieder herstellt, erscheint automatisch wieder in der Quell- und Zielbühne. Der Standardwert von `reconnectWindowSeconds` lautet 0.

Teilnehmerreplikation stoppen

Zum Stoppen der Replikation rufen Sie den Vorgang StopParticipantReplication auf.

Replikation von Teilnehmer A von Bühne A zu Bühne B stoppen:

```
aws ivs-realtime stop-participant-replication \
  --source-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageA \
  --destination-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageB \
  --participant-id participant-a-id
```

Replikation von Teilnehmer B von Bühne B zu Bühne A stoppen:

```
aws ivs-realtime stop-participant-replication \
  --source-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageB \
  --destination-stage-arn arn:aws:ivs:us-east-1:123456789012:stage/StageA \
  --participant-id participant-b-id
```

IVS Service Quotas | Echtzeit-Streaming

Im Folgenden finden Sie Service Quotas und Limits für Amazon Interactive Video Service (IVS)-Endpunkte, Ressourcen und andere Vorgänge. Service Quotas (auch als Limits bezeichnet) sind die maximale Anzahl von Service-Ressourcen oder Vorgängen für Ihr AWS-Konto. Das heißt, diese Grenzwerte gelten je AWS Konto, sofern in der Tabelle nichts anderes angegeben ist. Lesen Sie auch den Abschnitt AWS-Service-Quotas.

Um programmgesteuert eine Verbindung zu einem AWS-Service herzustellen, verwenden Sie einen Endpunkt. Lesen Sie auch den Abschnitt AWS-Service Quotas.

Alle Kontingente werden pro Konto in einer bestimmten AWS-Region durchgesetzt.

Erhöhte Service Quotas

Für einstellbare Kontingente können Sie eine Ratenerhöhung über die AWS-Konsole anfragen. Verwenden Sie die Konsole, um Informationen über Service Quotas anzuzeigen.

Kontingente für API-Anrufraten sind nicht anpassbar.

API-Aufrufratenquoten

Vorgangstyp	Operation	Standard
Composition	`GetComposition`	5 TPS
Composition	`ListCompositions`	5 TPS
Composition	`StartComposition`	5 TPS
Composition	`StopComposition`	5 TPS
IngestConfiguration	`CreateIngestConfiguration`	5 TPS
IngestConfiguration	`DeleteIngestConfiguration`	5 TPS
IngestConfiguration	`GetIngestConfiguration`	5 TPS
IngestConfiguration	`ListIngestConfigurations`	5 TPS

Vorgangstyp	Operation	Standard
IngestConfiguration	`UpdateIngestConfiguration`	5 TPS
MediaEncoder	`CreateEncoderConfiguration`	5 TPS
MediaEncoder	`DeleteEncoderConfiguration`	5 TPS
MediaEncoder	`GetEncoderConfiguration`	5 TPS
MediaEncoder	`ListEncoderConfigurations`	5 TPS
ParticipantReplication	`ListParticipantReplicas`	5 TPS
ParticipantReplication	`StartParticipantReplication`	5 TPS
ParticipantReplication	`StopParticipantReplication`	5 TPS
PublicKey	`DeletePublicKey`	3 TPS
PublicKey	`GetPublicKey`	3 TPS
PublicKey	`ImportPublicKey`	3 TPS
PublicKey	`ListPublicKeys`	3 TPS
Stage	`CreateParticipantToken`	50 TPS
Stage	`CreateStage`	5 TPS
Stage	`DeleteStage`	5 TPS
Stage	`DisconnectParticipant`	5 TPS
Stage	`GetParticipant`	5 TPS
Stage	`GetStage`	5 TPS
Stage	`GetStageSession`	5 TPS
Stage	`ListStages`	5 TPS

Vorgangstyp	Operation	Standard
Stage	`UpdateStage`	5 TPS
Stage	`ListParticipants`	5 TPS
Stage	`ListParticipantEvents`	5 TPS
Stage	`ListStageSessions`	5 TPS
StorageConfiguration	`CreateStorageConfiguration`	5 TPS
StorageConfiguration	`DeleteStorageConfiguration`	5 TPS
StorageConfiguration	`GetStorageConfiguration`	5 TPS
StorageConfiguration	`ListStorageConfigurations`	5 TPS
Tags	`ListTagsForResource`	10 TPS
Tags	`TagResource`	10 TPS
Tags	`UntagResource`	10 TPS

Andere Kontingente

Ressource oder Feature	Standard	Anpassbar	Beschreibung
Ziele für die Zusammensetzung	2	Nein	Maximale Anzahl von Zielobjekten in einer Zusammensetzungsressource.
Zusammensetzung: maximale Dauer	24	Nein	Maximale Dauer, für die eine Zusammensetzung existieren kann, in Stunden.
Zusammensetzungen	20	Ja	Maximale Anzahl gleichzeitiger Zusammensetzungsressourcen pro Konto.

Ressource oder Feature	Standard	Anpassbar	Beschreibung
Zusammensetzungen pro Stufe	5	Ja	Maximale Anzahl gleichzeitiger Zusammensetzungsressourcen pro Stufe.
Gleichzeitige Teilnehmerreplikationen	5	Nein	Höchstzahl gleichzeitiger Teilnehmerreplikationen pro Teilnehmer für alle Stufen einer AWS-Region.
Gleichzeitige Publisher	1.000	Ja	Höchstzahl von Teilnehmern, die Inhalte auf allen Bühnen einer AWS-Region veröffentlichen können.
Gleichzeitige Abonnements	20 000	Ja	Höchstzahl gleichzeitiger Verbindungen zwischen Publishern und Abonnenten für alle Bühnen einer AWS-Region.
EncoderConfigurations	20	Ja	Maximale Anzahl der EncoderConfiguration-Ressourcen pro Konto.
IngestConfigurations	100	Ja	Maximale Anzahl von IngestConfiguration-Ressourcen pro Konto.
Download-Bitrate des Teilnehmers	8,5 Mbit/s	Nein	Maximale aggregierte Download-Bitrate für alle Abonnements eines Teilnehmers.
Bitrate für Teilnehmerveröffentlichung	8,5 Mbit/s	Nein	Maximale Bits pro Sekunde, die an eine Bühne gestreamt werden.

Ressource oder Feature	Standard	Anpassbar	Beschreibung
Dauer der Veröffentlichung oder des Abonnements eines Teilnehmers	24	Nein	Die maximale Zeitspanne (in Stunden), in der ein Teilnehmer etwas auf einer Bühne veröffentlichen oder diese abonnieren kann.
Teilnehmer veröffentlicht Resolution	720p	Nein	Maximale Auflösung des von den Teilnehmern veröffentlichten Videos.
PublicKeys	3	Nein	Maximale Anzahl öffentlicher Schlüssel pro AWS-Region.
Teilnehmer der Bühne (Publisher)	12	Nein	Höchstzahl von Teilnehmern, die gleichzeitig zu einer Bühne veröffentlichen können.
Teilnehmer der Bühne (Subscriber)	10.000	Ja	Höchstzahl von Teilnehmern, die gleichzeitig eine Bühne abonnieren können.
Stufen	1.000	Ja	Höchstzahl von Stages pro AWS-Region
StorageConfigurations	5	Ja	Maximale Anzahl der StorageConfiguration-Ressourcen pro Konto.

Optimierungen für IVS-Echtzeit-Streaming

Um sicherzustellen, dass Ihre Benutzer das beste Erlebnis beim Streamen und Ansehen von Videos mithilfe von IVS Echtzeit-Streaming haben, gibt es verschiedene Möglichkeiten, das Erlebnis zu verbessern oder Teile des Erlebnisses zu optimieren, indem Sie die Features verwenden, die wir heute anbieten.

Einführung

Bei der Optimierung der Benutzererlebnisqualität ist es wichtig, die gewünschte Benutzererfahrung zu berücksichtigen. Diese kann sich je nach den Inhalten, die sie sich ansehen, und den Netzwerkbedingungen ändern.

In diesem Leitfaden konzentrieren wir uns auf Benutzer, die entweder Publisher von Streams oder Subscriber von Streams sind und wir berücksichtigen die gewünschten Aktionen und Erfahrungen dieser Nutzer.

Mit den IVS-SDKs können Sie die maximale Bitrate, Framerate und Auflösung von Streams konfigurieren. Wenn es bei Publishern zu Netzwerküberlastungen kommt, nimmt das SDK automatisch eine Anpassung vor und verringert die Videoqualität, indem es die Bitrate, Framerate und Auflösung senkt. Unter Android und iOS ist es möglich, die Einstellung zur Leistungsreduzierung bei Überlastungen auswählen. Das gleiche Verhalten tritt unabhängig davon auf, ob Sie die mehrschichtige Codierung mit Simulcast aktivieren oder die Standardkonfiguration beibehalten.

Adaptives Streaming: Mehrschichtige Kodierung mit Simulcast

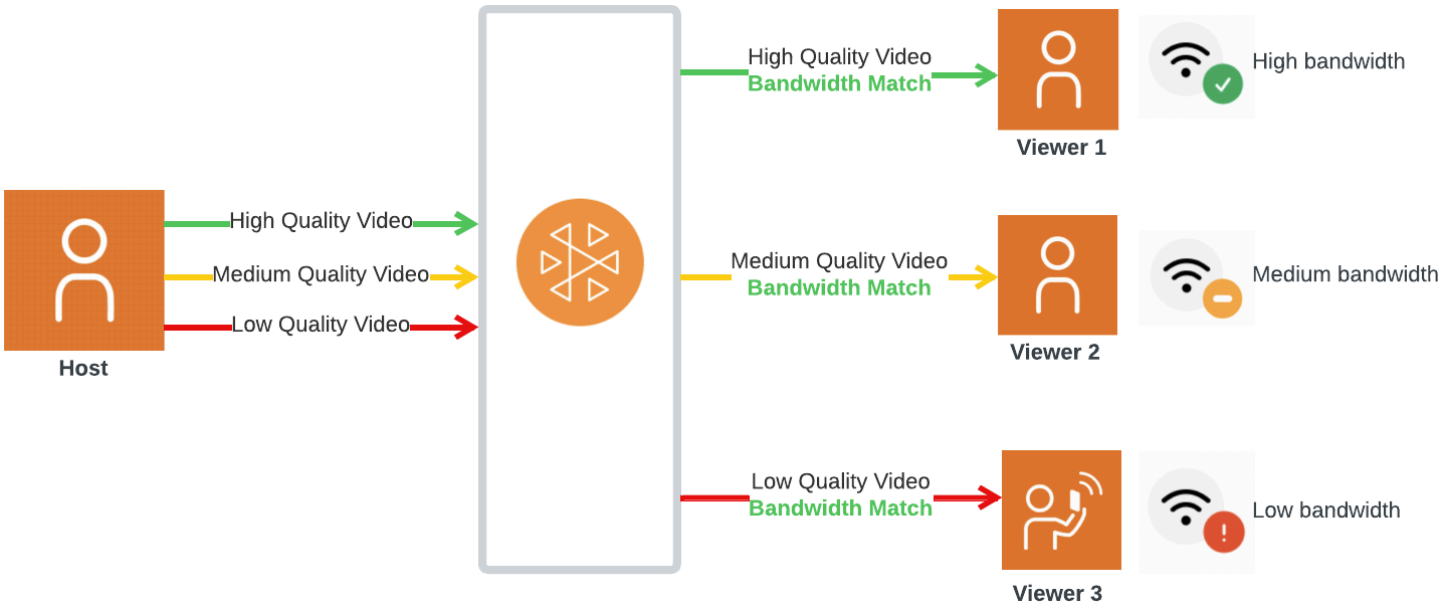
Dieses Feature wird nur von den folgenden Client-Versionen unterstützt:

- iOS und Android 1.18.0+
- Web 1.12.0+

Bei Verwendung der Echtzeit-Broadcast-SDKs von IVS können Publisher standardmäßig mehrere Video-Layer codieren und die Subscriber übernehmen automatisch die für ihr Netzwerk am besten geeignete Qualität oder wechseln zu dieser. Wir nennen das mehrschichtige Kodierung mit Simulcast.

Das Codieren von Layern mit Simulcast wird unter Android und iOS sowie in den Chrome- und Edge-Desktopbrowsern (für Windows und macOS) unterstützt. Wir unterstützen keine mehrschichtige Kodierung in anderen Browsern.

In der Abbildung unten sendet der Host drei Videoqualitäten (hoch, mittel und niedrig). IVS leitet auf der Grundlage der verfügbaren Bandbreite das Video von höchster Qualität an jeden Zuschauer weiter. Dies bietet jedem Zuschauer ein optimales Erlebnis. Wenn sich die Netzwerkverbindung von Zuschauer 1 von gut zu schlecht ändert, beginnt IVS automatisch damit, Video mit geringerer Qualität an Zuschauer 1 zu senden, sodass Zuschauer 1 den Stream weiterhin ohne Unterbrechung ansehen kann (mit der bestmöglichen Qualität).



Standard-Layers, Eigenschaften und Framerates

Die Standardqualitäten und Ebenen, die für Mobil- und Webbenutzer bereitgestellt werden, lauten wie folgt:

Mobil (Android, iOS)	Web (Chrome)
Hohe Schicht (oder benutzerdefiniert): • Maximale Bitrate: 900 000 bps • Bildrate: 15 fps	Hohe Schicht (oder benutzerdefiniert): • Maximale Bitrate: 1 700 000 bps • Bildrate: 30 fps

Mobil (Android, iOS)	Web (Chrome)
Mittelschicht: keine (nicht erforderl ich, da der Unterschied zwischen den Bitraten auf hoher und niedriger Ebene auf Mobilgeräten gering ist)	Mittelschicht: • Maximale Bitrate: 700 000 bps • Bildrate: 20 fps
Niedrige Schicht: • Maximale Bitrate: 100 000 BpS • Bildrate: 15 fps	Niedrige Schicht: • Maximale Bitrate: 200 000 bps • Bildrate: 15 fps

Auflösung von Layern

Die Auflösungen der mittleren und unteren Layer werden gegenüber dem oberen Layer automatisch herunterskaliert, um das gleiche Seitenverhältnis beizubehalten.

Mittlere und niedrige Layer werden ausgeschlossen, wenn ihre Auflösungen zu nahe an der Auflösung des darüber liegenden Layers liegen. Wenn die konfigurierte Auflösung beispielsweise 320 × 180 ist, sendet das SDK nicht auch Layer mit niedrigerer Auflösung.

Die folgende Tabelle zeigt die Auflösungen von Layern, die für verschiedene konfigurierte Auflösungen generiert wurden. Die aufgeführten Werte sind für das Querformat gedacht, können aber auch umgekehrt für Inhalte im Hochformat angewendet werden.

Eingabeauflösung	Auflösungen des Ausgabe-Layers: Mobil	Auflösungen des Ausgabe-Layers: Web
720p (1280 × 720)	Hoch (1280 × 720) Niedrig (320 × 180)	Hoch (1280 × 720) Mittel (640 × 360) Niedrig (320 × 180)
540p (960 × 540)	Hoch (960 × 540) Niedrig (320 × 180)	Hoch (960 × 540) Niedrig (320 × 180)
360p (640 × 360)	Hoch (640 × 360)	Hoch (640 × 360)

Eingabeauflösung	Auflösungen des Ausgabe-Layers: Mobil	Auflösungen des Ausgabe-Layers: Web
	Niedrig (360 × 180)	Niedrig (360 × 180)
270p (480 × 270)	Hoch (480 × 270)	Hoch (480 × 270)
180p (320 × 180)	Hoch (320 × 180)	Hoch (320 × 180)

Benutzerdefinierte Eingabeauflösungen, die oben nicht abgebildet sind, können Sie sie mit dem folgenden Tool berechnen.

Konfigurieren mehrschichtiger Kodierung mit Simulcast (Publisher)

Um die Codierung von Layern mit Simulcast verwenden zu können, müssen Sie das Feature auf dem Client aktiviert haben. Wenn Sie es aktivieren, werden Sie feststellen, dass der Publisher mehr Upload-Bandbreite nutzt, was möglicherweise dazu führt, dass das Video für Zuschauer weniger einfriert.

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
```

Ausführliche Informationen zur Konfiguration einzelner Ebenen finden Sie unter „Konfigurieren mehrschichtiger Kodierung (Publisher)“ in jedem Broadcast-SDK-Handbuch: Android, iOS und Web.

Konfigurieren mehrschichtiger Kodierung mit Simulcast (Subscriber)

Informationen zum Konfigurieren der Schichten, die von Subscribern empfangen werden, finden Sie in den Abschnitten unter „Mehrschichtige Kodierung mit Simulcast“ in den SDK-Leitfäden zum Echtzeit-Streaming:

- Broadcast-SDK für Android
- Broadcast-SDK für iOS
- Web-Broadcast-SDK

Mit der Subscriber-Konfiguration lässt sich die `InitialLayerPreference` definieren. Dies bestimmt, in welcher Qualität Videos anfänglich wiedergegeben werden, sowie den `preferredLayerForStream`, was wiederum bestimmt, welche Schicht bei der Videowiedergabe ausgewählt wird. Es gibt Ereignisse und Streaming-Methoden zur Benachrichtigung für den Fall, dass sich Schichten ändern, Anpassungen vorgenommen werden oder eine Schichtauswahl getroffen wird.

Streamingkonfigurationen

In diesem Abschnitt werden andere Konfigurationen beschrieben, die Sie an Ihren Video- und Audiostreams vornehmen können.

Ändern der Bitrate des Video-Streams

Verwenden Sie die folgenden Konfigurationsbeispiele, um die Bitrate Ihres Video-Streams zu ändern.

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
```

Ändern der Framerate des Video-Streams

Verwenden Sie die folgenden Konfigurationsbeispiele, um die Framerate Ihres Video-Streams zu ändern.

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
```

Optimieren der Audio-Bitrate und Stereo-Unterstützung

Verwenden Sie die folgenden Konfigurationsbeispiele, um die Bitrate und Stereoeinstellungen Ihres Audio-Streams zu ändern.

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
```

Ändern der Mindestverzögerung des Jitter-Puffers für Subscriber

Um die Mindestverzögerung des Jitter-Puffers für einen abonnierten Teilnehmer zu ändern, kann ein benutzerdefinierter `subscribeConfiguration`-Wert verwendet werden. Der Jitter-Puffer bestimmt, wie viele Pakete gespeichert werden, bevor die Wiedergabe beginnt. Die Mindestverzögerung stellt das Ziel für die minimale Datenmenge dar, die gespeichert werden soll. Eine Änderung der Mindestverzögerung kann dazu beitragen, dass die Wiedergabe bei Paketverlusten/Verbindungsproblemen stabiler wird.

Der Nachteil beim Vergrößern des Jitter-Puffers besteht darin, dass dadurch auch die Verzögerung vor dem Beginn der Wiedergabe erhöht wird. Eine Erhöhung der Mindestverzögerung sorgt für mehr Stabilität, geht jedoch zu Lasten der Zeit bis zum Video. Beachten Sie, dass eine Erhöhung der Mindestverzögerung bei der Wiedergabe einen ähnlichen Effekt hat: Die Wiedergabe wird kurz unterbrochen, damit der Jitter-Puffer gefüllt werden kann.

Wenn mehr Stabilität erforderlich ist, empfehlen wir, mit der Voreinstellung `MEDIUM` für die Mindestverzögerung zu beginnen und die Abonnementkonfiguration festzulegen, bevor die Wiedergabe beginnt.

Beachten Sie, dass die Mindestverzögerung nur angewendet wird, wenn ein Teilnehmer nur abonniert ist. Wenn ein Teilnehmer sich selbst veröffentlicht, wird die Mindestverzögerung nicht angewendet. Dadurch soll sichergestellt werden, dass mehrere Publisher ohne zusätzliche Verzögerung miteinander sprechen können.

In den folgenden Beispielen wird die voreingestellte Mindestverzögerung `MEDIUM` verwendet. Alle möglichen Werte finden Sie in der SDK-Referenzdokumentation.

Web

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

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

Android

```
@Override
public SubscribeConfiguration subscribeConfigrationForParticipant(@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
}
```

Empfohlene Optimierungen

Szenario	Empfehlungen
Streams mit Text oder sich langsam bewegenden Inhalten wie Präsentat ionen oder Folien	Verwenden Sie für Simulcast die Codierung auf mehreren Ebenen oder konfigurieren Sie Streams mit einer niedrigeren Framerate.
Streams mit Action oder viel Bewegung	Verwenden Sie für Simulcast die Codierung auf mehreren Ebenen.
Ströme mit Konversation oder wenig Bewegung	Verwenden Sie für Simulcast die Codierung auf mehreren Ebenen oder wählen Sie „Nur Audio“ (siehe „Teilnehmer abonnieren“ in den SDK-Leitfäden für Echtzeit-Streaming-Broadcasts: Web, Android und iOS.)

Szenario	Empfehlungen
Nutzer streamen mit begrenztem Datenvolumen	Verwenden Sie für Simulcast die Codierung auf mehreren Ebenen oder, wenn Sie eine geringere Datennutzung für alle wünschen, konfigurieren Sie eine niedrigere Framerate und senken Sie die Bitrate manuell.

Netzwerkanforderungen | Echtzeit-Streaming

Das Amazon-IVS-Streaming in Echtzeit basiert auf den Protokollen WebRTC und WebSocket für die Übertragung von Medien und Daten. Um ein reibungsloses Erlebnis zu gewährleisten, müssen die unten aufgeführten Ziele und Ports zugänglich sein. Jegliche Einschränkungen des eingehenden oder ausgehenden Datenverkehrs zu diesen Zielen können die Funktionalität des IVS-Echtzeit-Streamings beeinträchtigen.

Sie können das Netzwerktest-Tool verwenden, um sicherzustellen, dass Ihr Netzwerk ordnungsgemäß konfiguriert ist und dass der Datenverkehr zu und von den erforderlichen Zielen und Ports nicht blockiert wird.

Allgemein

Bestimmungsort	Ports
*.live-video.net	TCP:443

Medien

Standardmäßg nutzt IVS-Echtzeit-Streaming UDP für die Übertragung von Medien, um Streaming mit niedriger Latenz und hoher Leistung zu gewährleisten. Wenn UDP für Teilnehmer blockiert ist, die Medien abonnieren, wird TCP als Fallback verwendet. Das TCP-Fallback wird für die Veröffentlichung nicht unterstützt. Wenn der UDP-Datenverkehr also vollständig blockiert ist, schlägt die Veröffentlichung fehl.

Bestimmungsort	Ports
Alle Subnetze, die unter dem Service `IVS_REALTIME` in ip-ranges.json aufgeführt sind, müssen unabhängig von ihrer `region` oder der von Ihnen ausgewählten AWS-Region zugänglich sein. Teilnehmer werden unter Umständen automatisch mit einem Subnetz	UDP:3478 UDP: 443 TCP:3478 (Fallback) TCP:443

Bestimmungsort	Ports
verbunden. Details finden Sie unter Globale Lösung, regionale Kontrolle.	

IVS-Kosten | Echtzeit-Streaming

Details zu den Kosten für IVS finden Sie auf der Seite mit IVS-Preisen.

- Abonnieren von und Veröffentlichen auf Bühnen: Das Abonnieren und Veröffentlichen verbraucht Ressourcen. Für die Zeit, in der Sie mit der Bühne verbunden sind, wird ein Stundensatz berechnet.
- Aufzeichnung – Für die Aufzeichnung einzelner Teilnehmer fallen keine zusätzlichen Amazon-IVS-Gebühren an, während bei der zusammengesetzten Aufzeichnung Gebühren für den Stundensatz für die codierten Videos anfallen. Bei beiden Aufzeichnungsoptionen fallen standardmäßige S3-Speicher- und Anforderungskosten an. Für Thumbnails fallen keine zusätzlichen IVS-Gebühren an.
- Teilnehmerreplikation – Replikatteilnehmer werden genauso berechnet wie normale Teilnehmer.

  Nehmen wir zum Beispiel an, Sie haben zwei Bühnen: Bühne A mit Teilnehmer A und Bühne B mit Teilnehmer B. Dann werden Ihnen zwei Teilnehmer in Rechnung gestellt.

  Wenn Teilnehmer A zu Bühne B repliziert wird, haben Sie drei verbundene Teilnehmer (Teilnehmer A, Teilnehmer B und das Replikat von Teilnehmer A). Für die Dauer der Replikation werden Ihnen dann drei Teilnehmer in Rechnung gestellt.

Weitere Informationen finden Sie auf der Seite mit den IVS-Preisen.

IVS-Ressourcen und -Support | Echtzeit-Streaming

In diesem Dokument finden Sie Ressourcen, um Sie bei der Nutzung von Amazon-IVS-Streaming in Echtzeit zu unterstützen.

Demos und weitere Ressourcen

https://ivs.rocks/ ist eine spezielle Website, auf der Sie veröffentlichte Inhalte (Demos, Codebeispiele, Blogbeiträge) durchsuchen, die Kosten einschätzen und Amazon IVS anhand von Live-Demos erleben können. Demos und Codebeispiele finden Sie unter https://ivs.rocks/examples.

Die Seite zu Amazon IVS auf der Website der DEV-Community enthält eine Vielzahl von Demos und Blogbeiträgen. Unter Getting Started with Amazon Interactive Video Service Series' Articles zum Beispiel finden Sie eine Reihe von Artikeln zur Nutzung von Amazon IVS für Einsteiger. Die Artikel enthalten schrittweise Anleitungen zu IVS-APIs mit interaktiven Demos, die in die Beiträge eingebettet sind. Alle Demos können über einen eingebetteten CodePen direkt im Beitrag selbst ausgeführt werden.

Die Website AWS Blogs enthält zahlreiche IVS-Blogbeiträge zu verschiedensten Themen. Filtern Sie nach Amazon IVS, indem Sie rechts auf der Seite nacheinander Product or solution > Media Services > Amazon Interactive Video Service auswählen.

Auf GitHub zeigt die Demo zum IVS-Echtzeit-Streaming für iOS und Android Entwicklern, wie sie mit IVS eine überzeugende Anwendung für die Echzeit-Generierung von Inhalten durch Social-Media-Benutzer entwickeln können:

Diese Anwendung bietet einen scrollbaren Feed mit benutzergenerierten Echtzeit-Streams. Benutzer können Videostreams und Nur-Audioräume erstellen. Videostream-Gäste können im Gastmodus oder im Versus-Modus (VS) teilnehmen. Anweisungen zur Bereitstellung des erforderlichen Backends und zum Erstellen der Anwendung finden Sie in den folgenden GitHub-Repositorys:

- 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/

Support

Das AWS-Supportcenter bietet eine Reihe von Plänen, die den Zugriff auf Tools und das Know-how zur Unterstützung Ihrer AWS-Lösungen bieten. Alle Supportpläne bieten Zugriff auf Kundenservice rund um die Uhr. Wenn Sie technischen Support und Zugriff auf zusätzliche Ressourcen benötigen, um Ihre AWS-Umgebung zu planen, bereitzustellen und zu verbessern, können Sie einen Support-Plan auswählen, der für Ihren AWS-Anwendungsfall optimal angepasst ist.

AWS Premium Support ist ein direkter und schneller Supportkanal, der Sie beim Entwickeln und Ausführen von Anwendungen in AWS unterstützt.

AWS re:Post ist eine Community-basierte Q&A-Website, die für Entwickler eingerichtet wurde, um über technische Fragen zu Amazon IVS zu diskutieren.

AWS kontaktieren enthält Links für nicht-technische Anfragen zu Ihrer Abrechnung oder Ihrem Konto. Technische Fragen stellen Sie bitte in den Diskussionsforen oder über die Support-Links.

IVS-Glossar

Weitere Informationen finden Sie im AWS-Glossar. In der folgenden Tabelle steht LL für IVS-Streaming mit niedriger Latenz. RT und IVS-Echtzeit-Streaming.

Begriff	Beschreibung	LL	RT	Chat
AAC	Erweiterte Audio-Kodierung. AAC ist ein Audio-Kodierungsstandard für verlustbehaftete digitale Audiokomprimierung. Als Nachfolger des MP3-Formats konzipiert, erreicht AAC bei gleicher Bitrate im Allgemeinen eine höhere Klangqualität als MP3. AAC wurde von ISO und IEC als Teil der Spezifikationen MPEG-2 und MPEG-4 standardi siert.	✓	✓	
Streaming mit adaptiver Bitrate	Beim Streaming mit adaptiver Bitrate (ABR) kann der IVS-Player auf eine niedrigere Bitrate umschalten, wenn die Verbindungsqualität beeinträchtigt ist, und auf eine höhere Bitrate zurückschalten, wenn sich die Verbindungsqualität verbessert.	✓		
Adaptives Streaming	Weitere Informationen finden Sie unter Mehrschic htige Kodierung mit Simulcast.		✓	
Administratorbenut zer	Ein AWS-Benutzer mit Administratorzugriff auf Ressourcen und Services, die in einem AWS-Konto verfügbar sind. Weitere Informationen finden Sie unter Terminologie im Benutzerhandbuch zur AWS-Einrichtung.	✓	✓	✓
ARN	Amazon-Ressourcenname, eine eindeutige Kennung für eine AWS-Ressource. Spezifisc he ARN-Formate hängen vom Ressourcentyp ab. Informationen zu den von IVS-Ressourcen	✓	✓	✓

Begriff	Beschreibung	LL	RT	Chat
	verwendeten ARN-Formaten finden Sie in der Service-Autorisierungsreferenz.			
Seitenverhältnis	Beschreibt das Verhältnis der Rahmenbreite zur Rahmenhöhe. Beispielsweise ist 16:9 das Seitenverhältnis, das der Full-HD- oder 1080p-Auflösung entspricht.	✓	✓	
Audio-Modus	Eine voreingestellte oder benutzerdefinierte Audiokonfiguration, die für verschiedene Arten von Benutzern mobiler Geräte und die von ihnen verwendeten Geräte optimiert ist. Weitere Informationen finden Sie unter IVS Broadcast SDK: Mobile Audiomodi (Echtzeit-Streaming).		✓	
AVC, H.264, MPEG-4 Teil 10	Erweiterte Video-Kodierung, auch als H.264 oder MPEG-4 Teil 10 bezeichnet, ein Videokomprimierungsstandard für verlustbehaftete digitale Videokomprimierung.	✓	✓	
Ersetzen des Hintergrunds	Eine Art Kamerafilter, der es Livestream-Erstellern ermöglicht, ihren Hintergrund zu ändern. Weitere Informationen finden Sie unter Ersetzen des Hintergrunds in IVS Broadcast SDK: Kamera-Filter von Drittanbietern (Echtzeit-Streaming).		✓	
Bitrate	Eine Streaming-Metrik für die Anzahl der pro Sekunde übertragenen oder empfangenen Bits.	✓	✓	
Broadcast, Sender	Andere Begriffe für Stream, Streamer.	✓		

Begriff	Beschreibung	LL	RT	Chat
Pufferung	Ein Zustand, der auftritt, wenn das Wiedergab egerät den Inhalt nicht herunterladen kann, bevor der Inhalt abgespielt werden soll. Pufferung kann sich auf verschiedene Weise äußern: Inhalte können zufällig anhalten und starten (auch Stottern genannt), Inhalte können für längere Zeit anhalten (auch Einfrieren genannt) oder der IVS-Player kann die Wiedergabe anhalten.	✓	✓	
Wiedergabeliste im Bytebereich	Eine differenziertere Wiedergabeliste als die standardmäßige HLS-Wiedergabeliste. Die standardmäßige HLS-Wiedergabeliste besteht aus 10-sekündigen Mediendateien. Bei einer Wiedergabeliste mit Byte-Bereich entspricht die Segmentdauer dem Keyframe-Intervall, das für den Stream konfiguriert wurde. Die Wiedergabeliste im Bytebereich ist nur für Übertragungen verfügbar, die automatisch in einem S3-Bucket aufgezeichnet wurden. Diese wird zusätzlich zur HLS-Wiedergabeliste erstellt. Weitere Informationen finden Sie unter Wiedergab elisten im Byte-Bereich in Automatische Aufzeichn ung in Amazon S3 (Streaming mit niedriger Latenz).	✓		

Begriff	Beschreibung	LL	RT	Chat
CBR	Konstante Bitrate, eine Ratensteuerungsmethode für Encoder, die während der gesamten Wiedergabe eines Videos auf eine einheitliche Bitrate warten, unabhängig davon, was während der Übertragung passiert. Tiefpunkte im Geschehen können aufgefüllt werden, um die gewünschte Bitrate zu erreichen, und Spitzen können quantisiert werden, indem die Qualität der Kodierung an die Ziel-Bitrate angepasst wird. Wir empfehlen nachdrücklich die Verwendung von CBR anstelle von VBR.	✓	✓	
CDN	Netzwerk für die Inhaltsbereitstellung oder Netzwerk für die Inhaltsübermittlung, eine geografisch verteilte Lösung, die die Bereitstellung von Inhalten wie Streaming-Videos optimiert, indem sie diese näher an den Standort der Benutzer bringt.	✓		
Kanal	Eine IVS-Ressource, die die Konfiguration für das Streaming speichert, einschließlich eines Aufnahmeservers, eines Stream-Schlüssels, einer Wiedergabe-URL und Aufzeichnungsoptionen. Streamer verwenden den Streamschlüssel, der einem Kanal zugeordnet ist, um eine Übertragung zu starten. Alle während einer Übertragung generierten Metriken und Ereignisse werden einer Kanalressource zugeordnet.	✓		
Kanaltyp	Legt die zulässige Auflösung und Bildfrequenz für den Kanal fest. Siehe Kanaltypen in der Referenz der API von IVS-Streaming mit niedriger Latenz.	✓		
Chat-Protokollierung	Eine erweiterte Option, die durch die Zuordnung einer Protokollierungskonfiguration zu einem Chatroom ermöglicht wird.			✓

Begriff	Beschreibung	LL	RT	Chat
Chatroom	Eine IVS Ressource, die die Konfiguration für eine Chatsitzung speichert, einschließlich optionale r Features wie Handler zur Nachrichtenüberprü fung und Chat-Protokollierung. Weitere Informati onen finden Sie unter Schritt 2: Erstellen eines Chatrooms unter Erste Schritte mit Amazon IVS Chat.			✓
Clientseitige Zusammensetzung	Verwendet ein Host-Gerät zum Mischen von Audio- und Videostreams von Stage-Teilnehmern und sendet sie dann als zusammengesetzten Stream an einen IVS-Kanal. Dies ermöglicht eine bessere Kontrolle über das Erscheinungsbild der Zusammensetzung, allerdings auf Kosten einer höheren Auslastung der Client-Ressourcen und eines höheren Risikos, dass sich ein Stage- oder Host-Problem auf die Zuschauer auswirkt. Weitere Informationen finden Sie unter serversei tige Zusammensetzung.	✓	✓	
CloudFront	Ein von Amazon bereitgestellter CDN-Service.	✓		
CloudTrail	Ein AWS-Service zur Erfassung, Überwachung, Analyse und Beibehaltung von Ereignissen und Kontoaktivitäten von AWS und externen Quellen. Weitere Informationen finden Sie unter Protokoll ierung von IVS-API-Aufrufen mit AWS CloudTrail.	✓	✓	✓

Begriff	Beschreibung	LL	RT	Chat
CloudWatch	Ein AWS-Service zur Überwachung von Anwendungen, zur Reaktion auf Leistungs änderungen, zur Optimierung der Ressource nnutzung und zur Bereitstellung von Einblicken in den Betriebszustand. Sie können CloudWatc h verwenden, um IVS-Metriken zu überwachen. Weitere Informationen finden Sie unter Überwachu ng von IVS-Echtzeit-Streaming und Überwachung von IVS-Streaming mit niedriger Latenz.	✓	✓	✓
Composition	Der Prozess des Zusammenführens von Audio- und Videostreams aus mehreren Quellen zu einem einzigen Stream.	✓	✓	
Pipeline der Zusammensetzung	Eine Abfolge von Verarbeitungsschritten, die zum Kombinieren mehrerer Streams und zum Kodieren des resultierenden Streams erforderlich sind.	✓	✓	
Komprimierung	Kodierung von Informationen mit weniger Bits als in der ursprünglichen Darstellung. Jede einzelne Komprimierung ist entweder verlustfrei oder verlustbehaftet. Die verlustfreie Komprimierung reduziert die Anzahl der Bits durch die Identifiz ierung und Beseitigung statistischer Redundanz. Bei der verlustfreien Komprimierung gehen keine Informationen verloren. Bei der verlustbehafteten Komprimierung werden Bits reduziert, indem unnötige oder weniger wichtige Informationen entfernt werden.	✓	✓	
Steuerebene	Speichert Informationen zu IVS-Ressourcen wie Kanälen, Stages oder Chatrooms und stellt Schnittstellen zum Erstellen und Verwalten dieser Ressourcen bereit. Es ist regional (basierend auf AWS-Regionen).	✓	✓	✓

Begriff	Beschreibung	LL	RT	Chat
CORS	Herkunftsübergreifende Ressourcenfreigabe, ein AWS-Feature, mit dem Client-Webanwendungen, die in einer Domain geladen sind, mit Ressourcen wie S3-Buckets in einer anderen Domain interagieren können. Der Zugriff kann basierend auf Headern, HTTP-Methoden und Ursprungsdomains konfiguriert werden. Weitere Informationen finden Sie unter Nutzung der herkunftsübergreifenden Ressourcennutzung (CORS) – Amazon Simple Storage Service im Benutzerhandbuch für Amazon Simple Storage Service.	✓		
Benutzerdefinierte Bildquelle	Eine Schnittstelle, die vom IVS Broadcast SDK bereitgestellt wird und es einer Anwendung ermöglicht, ihre eigene Audioeingabe bereitzustellen, anstatt auf das integrierte Mikrofon des Geräts beschränkt zu sein.		✓	
Benutzerdefinierte Bildquelle	Eine vom IVS Broadcast SDK bereitgestellte Schnittstelle, über die eine Anwendung ihre eigene Bildeingabe bereitstellen kann, anstatt auf die voreingestellten Kameras beschränkt zu sein.	✓	✓	
Benutzerdefinierte Teilnehmeranordnung	Ermöglicht die Positionierung von Bühnenteilnehmern sowohl im Grid- als auch im PiP-Layout auf der Grundlage von benutzerdefinierten Attributwerten in Teilnehmer-Token.		✓	
Datenebene	Die Infrastruktur, die Daten von der Aufnahme bis zum Ausgang überträgt. Der Betrieb basiert auf der in der Steuerebene verwalteten Konfiguration und ist nicht auf eine AWS-Region beschränkt.	✓	✓	✓

Begriff	Beschreibung	LL	RT	Chat
Encoder, Verschlüsselung	Der Vorgang der Konvertierung von Video- und Audioinhalten in ein für Streaming geeignetes digitales Format. Die Kodierung kann hardware- oder softwarebasiert sein.	✓	✓	
E-RTMP	Verbessertes RTMP-Protokoll. IVS unterstützt die Funktionen von E-RTMP, die für Multitrack-Video erforderlich sind.	✓		
Ereignis	Eine automatische Benachrichtigung, die von IVS an den AmazonEventBridge-Überwachungsservice veröffentlicht wird. Ein Ereignis stellt eine Zustands- oder Zustandsänderung einer Streaming-Ressource dar, beispielsweise einer Stage oder einer Zusammensetzungs-Pipeline. Weitere Informationen finden Sie unter Verwendung von Amazon EventBridge mit IVS-Streaming mit niedriger Latenz und Verwendung von Amazon EventBridge mit IVS-Echtzeit-Streaming.	✓	✓	✓
FFmpeg	Ein kostenloses Open-Source-Softwareprojekt, das aus einer Reihe von Bibliotheken und Programmen zur Verarbeitung von Video- und Audiodateien und -streams besteht. FFmpeg bietet eine plattformübergreifende Lösung zum Aufzeichnen, Konvertieren und Streamen von Audio und Video.	✓		

Begriff	Beschreibung	LL	RT	Chat
Fragmentierter Stream	Wird erstellt, wenn eine Übertragung innerhalb des in der Aufzeichnungskonfiguration des Kanals angegebenen Intervalls unterbrochen und dann erneut verbunden wird. Die resultier enden mehreren Streams werden als eine einzelne Übertragung betrachtet und zu einem einzigen aufgezeichneten Stream zusammengeführt. Weitere Informationen finden Sie unter Zusammenf ühren fragmentierter Streams in Automatische Aufzeichnung in Amazon S3 (Streaming mit niedriger Latenz).	✓		
Bildrate	Eine Streaming-Metrik für die Anzahl der übertrage nen oder empfangenen Videobilder pro Sekunde.	✓	✓	
HLS	HTTP Live Streaming (HLS), ein HTTP-basiertes Streaming-Kommunikationsprotokoll mit adaptiver Bitrate, das zur Bereitstellung von IVS-Streams an Zuschauer verwendet wird.	✓		
HLS-Wiedergabelist e	Eine Liste von Mediensegmenten, aus denen ein Stream besteht. Standard-HLS-Wiedergabelisten bestehen aus 10-sekündigen Mediendateien. HLS unterstützt auch detailliertere Wiedergabelisten im Bytebereich.	✓		
Host	Ein Echtzeit-Benutzer, der eine Stage erstellt.		✓	
IAM	Identity and Access Management, ein AWS-Servi ce, der Benutzern die sichere Verwaltung von Identitäten und Zugriff auf AWS-Services und -Ressourcen, einschließlich IVS, ermöglicht.	✓	✓	✓

Begriff	Beschreibung	LL	RT	Chat
Ergest	IVS-Prozess zum Empfang von Videostreams von einem Host oder Sender zur Verarbeitung oder Bereitstellung an Zuschauer oder andere Teilnehmer.	✓	✓	
Server Ingest	Empfängt Videostreams und überträgt sie an ein Transkodierungssystem, wo Streams für die Bereitstellung an die Zuschauer in HLS transmuxiert oder transkodiert werden. Aufnahme-Server sind spezielle IVS-Komponenten, die Streams für Kanäle zusammen mit einem Aufnahmeprotokoll (RTMP, RTMPS) empfangen. Weitere Informationen zum Erstellen eines Kanals finden Sie unter Erste Schritte mit IVS-Streaming mit niedriger Latenz.	✓		
Video mit Zeilensprung	Überträgt und zeigt nur ungerade oder gerade Zeilen von aufeinanderfolgenden Frames an, um eine wahrgenommene Verdoppelung der Bildfrequenz zu erreichen, ohne zusätzliche Bandbreite zu verbrauchen. Aufgrund von Bedenken hinsichtlich der Videoqualität wird die Verwendung von Video mit Zeilensprung nicht empfohlen.	✓	✓	
JSON	JavaScript Object Notation, ein Open-Standard-Dateiformat, das für Menschen lesbaren Text verwendet, um Datenobjekte zu übertragen, die aus Attribut-Wert-Paaren und Array-Datentypen oder anderen serialisierbaren Werten bestehen.	✓	✓	✓

Begriff	Beschreibung	LL	RT	Chat
Keyframe, Delta-Frame, Keyframe-Intervall	Der Keyframe (auch als intra-kodiert oder i-Frame bezeichnet) ist ein Vollbild des Bildes in einem Video. Nachfolgende Frames, die Deltaframes (auch als prognostizierte oder p-Frames bezeichnet), enthalten nur die geänderten Informationen. Abhängig vom im Encoder definierten Keyframe-Intervall werden Keyframes innerhalb eines Streams mehrmals angezeigt.	✓	✓	
Lambda	Ein AWS-Service zum Ausführen von Code (als Lambda-Funktionen bezeichnet) ohne Bereitstellung einer Serverinfrastruktur. Lambda-Funktionen können als Reaktion auf Ereignisse und Aufrufanfragen oder basierend auf einem Zeitplan ausgeführt werden. IVS Chat verwendet beispielsweise Lambda-Funktionen, um die Nachrichtenüberprüfung für einen Chatroom zu ermöglichen.	✓	✓	✓
Latenz, Glas-zu-Glas-Latenz	Eine Verzögerung bei der Datenübertragung. IVS definiert Latenzbereiche wie folgt: • Niedrige Latenz: unter 3 Sekunden • Latenz in Echtzeit: unter 300 ms Glas-zu-Glas-Latenz bezieht sich auf die Verzögerung, wenn eine Kamera einen Livestream aufnimmt, bis zu dem Zeitpunkt, an dem der Stream auf dem Bildschirm eines Betrachters angezeigt wird.	✓	✓	
Mehrschichtige Kodierung mit Simulcast	Ermöglicht die gleichzeitige Kodierung und Veröffentlichung mehrerer Videostreams mit unterschiedlichen Qualitätsstufen. Weitere Informationen finden Sie unter Adaptives Streaming: Mehrschichtige Kodierung mit Simulcast in Streaming-Optimierungen in Echtzeit.		✓	

Begriff	Beschreibung	LL	RT	Chat
Handler für Nachrichtenüberprüfung	Ermöglicht es IVS-Chat-Kunden, Benutzer-Chat-Nachrichten automatisch zu überprüfen/zu filtern, bevor sie an den Chatroom übermittelt werden. Dies wird durch die Verknüpfung einer Lambda-Funktion mit einem Chatroom ermöglicht. Weitere Informationen finden Sie unter Erstellen einer Lambda-Funktion im Handler für Nachrichtenüberprüfung.			✓
Mischpult	Ein Feature der IVS Mobile Broadcast SDKs, die mehrere Audio- und Videoquellen aufnimmt und eine einzige Ausgabe generiert. Diese Funktion unterstützt die Verwaltung von Video- und Audioelementen auf dem Bildschirm, die Quellen wie Kameras, Mikrofone, Bildschirmaufnahmen sowie von der Anwendung generiertes Audio und Video darstellen. Die Ausgabe kann anschließend an IVS gestreamt werden. Weitere Informationen finden Sie unter Konfigurieren einer Broadcast-Sitzung zum Mischen im IVS Broadcast SDK: Mixer-Handbuch (Streaming mit niedriger Latenz).	✓		
Streaming auf mehreren Hosts	Kombiniert Streams von mehreren Hosts zu einem einzigen Stream. Dies kann entweder durch clientseitige oder serverseitige Zusammensetzung erreicht werden. Das Streaming mit mehreren Hosts ermöglicht Szenarien wie die Einladung von Zuschauern auf eine Stage für Fragen und Antworten, Wettbewerbe zwischen Hosts, Videochats und Gespräche zwischen den Hosts vor einem großen Publikum.		✓	

Begriff	Beschreibung	LL	RT	Chat
Multitrack-Video	Ermöglicht Broadcaster-Softwaretools das Kodieren und Streaming mehrerer Videoqualitäten direkt von einem GPU-gestützten Computer aus. Siehe Multitrack-Video von Amazon IVS.	✓		
Multivariante Wiedergabeliste	Ein Index aller Varianten-Streams, die für eine Übertragung verfügbar sind.	✓		
OAC	Origin Access Control, ein Mechanismus zur Einschränkung des Zugriffs auf einen S3-Bucket , sodass Inhalte wie ein aufgezeichneter Stream nur über CloudFront CDN bereitgestellt werden können.	✓		
OBS	Open Broadcaster Software, kostenlose und Open-Source-Software für Videoaufzeichnung und Live-Streaming. OBS bietet eine Alternative (zum IVS Broadcast SDK) für Desktop-Publishing. Erfahrene re Streamer, die mit OBS vertraut sind, bevorzuge n es möglicherweise aufgrund seiner erweitert en Produktionsfeatures wie Szenenübergänge, Audiomischung und Overlay-Grafiken.	✓	✓	
Teilnehmer	Ein Echtzeit-Benutzer, der als Publisher oder Subscriber mit einer Stage verbunden ist.		✓	
Teilnehmerreihenfo lge	Die Reihenfolge, in der die Teilnehmer der Phase in Grid- und PiP-Layouts positioniert sind.		✓	
Teilnehmer-Token	Authentifiziert einen Teilnehmer eines Ereigniss es in Echtzeit, wenn er einer Stage beitritt. Ein Teilnehmer-Token steuert auch, ob ein Teilnehmer Videos an die Stage senden kann.		✓	

Begriff	Beschreibung	LL	RT	Chat
Wiedergabe-Token, Wiedergabe-Schlüsselpaar	Ein Autorisierungsmechanismus, mit dem Kunden die Videowiedergabe auf privaten Kanälen beschränken können. Wiedergabe-Token werden aus einem Wiedergabe-Schlüsselpaar generiert. Ein Wiedergabe-Schlüsselpaar ist das öffentlic h-private Schlüsselpaar, das zum Signieren und Validieren des Viewer-Autorisierungs-Token für die Wiedergabe verwendet wird. Siehe Erstellen oder Importieren eines IVS-Wiedergabeschlüssels unter Einrichten von IVS-Privatkanälen und die Vorgänge für Wiedergabeschlüsselpaare unter Referenz zur API von IVS mit niedriger Latenz.	✓		
Wiedergabe-URL	Gibt die Adresse an, die ein Zuschauer verwendet , um die Wiedergabe für einen bestimmten Kanal zu starten. Diese Adresse kann global verwendet werden. Amazon IVS wählt automatisch den besten Standort im globalen Netzwerk für die Inhaltsbe reitstellung von IVS aus, um das Video an jeden Zuschauer zu übermitteln. Weitere Informationen zum Erstellen eines Kanals finden Sie unter Erste Schritte mit IVS-Streaming mit niedriger Latenz.	✓		
Privater Kanal	Ermöglicht Kunden die Einschränkung des Zugriffs auf ihre Streams mithilfe eines Autorisierungsmech anismus, der auf Wiedergabe-Tokens basiert. Siehe Workflow für private IVS-Kanäle unter Einrichten von privaten IVS-Kanälen.	✓		
Private Erfassung	Ermöglicht eine sichere private Verbindung zwischen Ihrer Amazon VPC und IVS mithilfe von VPC-Schnittstellen-Endpunkten, die von AWS PrivateLink betrieben werden. Siehe Private Erfassung in IVS	✓		

Begriff	Beschreibung	LL	RT	Chat
Progressives Video	Überträgt und zeigt alle Zeilen jedes Frames der Reihe nach an. Es empfiehlt sich die Verwendun g von progressivem Video in allen Stages einer Übertragung.	✓	✓	
Publisher	Ein Teilnehmer eines Echtzeit-Ereignisses, der Video und/oder Audio auf einer Stage veröffentlicht. Weitere Informationen finden Sie unter Was ist IVS-Echtzeit-Streaming?.		✓	
Kontingente	Die maximale Anzahl von IVS-Serviceressourcen oder -Vorgängen für Ihr AWS-Konto. Das heißt, diese Grenzwerte gelten pro AWS-Konto, sofern nicht anders angegeben. Alle Kontingente werden pro Region erzwungen. Weitere Informationen finden Sie unter Endpunkten und Kontingenten von Amazon Interactive Video Service im Allgemeinen AWS-Referenzhandbuch.	✓	✓	✓

Begriff	Beschreibung	LL	RT	Chat
Regionen	Bieten Sie Zugriff auf AWS-Services, die sich physisch in einem bestimmten geografischen Gebiet befinden. Regionen bieten Fehlertoleranz, Stabilität und Ausfallsicherheit und können auch die Latenz verkürzen. Mit Regionen können Sie redundante Ressourcen erstellen, die verfügbar bleiben und von einem regionalen Ausfall nicht betroffen werden. Die meisten AWS-Serviceanfragen beziehen sich auf eine bestimmte geografische Region. Die Ressourcen, die Sie in einer Region erstellen, sind in keiner anderen Region vorhanden, es sei denn, Sie verwenden ausdrücklich ein Replikati onsfeature, die von einem AWS-Service angeboten wird. Beispielsweise unterstützt Amazon S3 die regionsübergreifende Replikation. Einige Services, wie etwa IAM, verfügen über keine regionsüb ergreifende Ressourcen.	✓	✓	✓
Auflösung	Beschreibt die Anzahl der Pixel in einem einzelnen Videobild. Full HD oder 1080p definiert beispiels weise ein Frame mit 1920x1080 Pixeln.	✓	✓	
Stammbenutzer	Der Besitzer eines AWS-Kontos. Der Root-Benu tzer hat vollständigen Zugriff auf alle AWS-Services und Ressourcen im AWS-Konto.	✓	✓	✓
RTMP, RTMPS	Real-Time Messaging Protocol, ein Branchens tandard zum Übertragen von Audio, Video und Daten über ein Netzwerk. RTMPS ist die sichere Version von RTMP, die über eine Transport Layer Security (TLS/SSL)-Verbindung ausgeführt wird.	✓	✓	

Begriff	Beschreibung	LL	RT	Chat
S3-Bucket	Eine Sammlung von Objekten, die in Amazon S3 gespeichert sind. Viele Richtlinien, einschließlich Zugriff und Replikation, werden auf Bucket-Ebene definiert und gelten für alle Objekte im Bucket. Beispielsweise wird eine IVS-Übertragung in Form mehrerer Objekte in einem S3-Bucket gespeichert.	✓		
SDK	Software Development Kit, eine Sammlung von Bibliotheken für Entwickler, die Anwendungen mit IVS erstellen. Das IVS Player SDK dient der Wiedergabe von IVS-Streams. Es nutzt die IVS-Architektur und ist für die IVS-Wiedergabe mit niedriger Latenz optimiert. IVS bietet ein Broadcast-SDK für Web, Android und iOS, das Sie in Ihre Anwendung integrieren können. Das IVS-Broadcast-SDK ist für Entwickler konzipiert, die Android- oder iOS-Anwendungen mit IVS erstellen. Dieses SDK nutzt die IVS-Architektur und wird zusammen mit IVS kontinuierlich verbessert. Als natives Broadcast-SDK wurde es entwickelt, um die Leistungsauswirkungen auf Ihre Anwendungen und auf die Geräte, mit denen Ihre Benutzer auf Ihre Anwendungen zugreifen, zu minimieren. Es gibt IVS-Broadcast-SDKs für Web, Android und iOS für Streaming mit niedriger Latenz und Echtzeit-Streaming.	✓	✓	✓

Begriff	Beschreibung	LL	RT	Chat
Selfie-Segmentation	Ermöglicht das Ersetzen des Hintergrunds in einem Live-Stream mithilfe einer Client-spezifischen Lösung. Diese akzeptiert ein Kamerabild als Eingabe und gibt eine Maske zurück, die für jedes Pixel des Bildes eine Wertung bereitstellt und angibt, ob es sich im Vordergrund oder im Hintergrund befindet. Weitere Informationen finden Sie unter Ersetzen des Hintergrunds in IVS Broadcast SDK: Kamera-Filter von Drittanbietern (Echtzeit-Streaming).		✓	
Semantische Versionsverwaltung	Ein Versionsformat in Form von Major.Minor.Patch. Fehlerkorrekturen, die sich nicht auf die API auswirken, erhöhen die Patch-Version, rückwärts kompatible API-Ergänzungen/Änderungen erhöhen die Nebenversion und rückwärtsinkompatible API-Änderungen erhöhen die Hauptversion.	✓	✓	✓
Serverseitige Zusammensetzung	Verwendet einen IVS-Server zum Mischen von Audio und Video von Teilnehmern einer Stage und sendet dieses gemischte Video dann an einen IVS-Kanal, um ein größeres Publikum zu erreichen oder um es in einem S3-Bucket zu speichern. Die serverseitige Zusammensetzung reduziert die Client-Auslastung, verbessert die Stabilität der Übertragung und ermöglicht eine effizientere Nutzung der Bandbreite. Weitere Informationen finden Sie unter Clientseitige Zusammensetzung.		✓	

Begriff	Beschreibung	LL	RT	Chat
Servicekontingente	Ein AWS-Service, mit dem Sie Ihre Kontingente für viele AWS-Services von einem Standort aus verwalten können. Sie können über die Service–Quotas-Konsole Kontingentwerte abfragen und außerdem Kontingenterhöhungen anfordern.	✓	✓	✓
Serviceverknüpfte Rolle	Ein eindeutiger IAM-Rollentyp, der direkt mit einem AWS-Service verknüpft ist. Serviceverknüpfte Rollen werden automatisch von IVS erstellt und enthalten alle Berechtigungen, die der Service benötigt, um andere AWS-Services in Ihrem Namen aufzurufen, z. B. für den Zugriff auf einen S3-Bucket. Weitere Informationen finden Sie unter Nutzung serviceverknüpfter Rollen für IVS in IVS-Sicherheit.	✓		
Stage	Eine IVS Ressource, die eine virtuelle Umgebung darstellt, in dem die Echtzeit-Teilnehmer eines Ereignisses Videos in Echtzeit austauschen können. Weitere Informationen finden Sie unter Erstellen einer Stage mit optionaler Teilnehme raufzeichnung in Erste Schritte mit IVS-Echtzeit-Streaming.		✓	
Stages-Sitzung	Beginnt, wenn der erste Teilnehmer eine Stage betritt und endet einige Minuten, nachdem der letzte Teilnehmer die Veröffentlichung in der Stage beendet hat. Eine langlebige Stage kann im Laufe ihrer Lebensdauer möglicherweise mehrere Sitzungen haben.		✓	
Stream	Daten, die Video- oder Audioinhalte darstellen und fortlaufend von einer Quelle an ein Ziel gesendet werden.	✓	✓	

Begriff	Beschreibung	LL	RT	Chat
Stream-Schlüssel	Eine von IVS beim Erstellen eines Kanals zugewiesene Kennung. Es wird verwendet, um das Streaming zum Kanal zu autorisieren. Behandeln Sie den Stream-Schlüssel wie ein Geheimnis, da jeder mit ihm berechtigt ist auf den Kanal zu streamen. Weitere Informationen finden Sie unter Erste Schritte mit IVS-Streaming mit niedriger Latenz.	✓		
Stream-Starvation	Eine Verzögerung oder ein Stopp bei der Stream-Übermittlung an IVS. Dies tritt auf, wenn IVS nicht die erwartete Anzahl an Bits empfängt, die das Kodierungsgerät angekündigt hat und die es über einen bestimmten Zeitraum senden würde. Das Auftreten einer Stream-Starvation führt zu einem Stream-Starvation-Ereignis. Aus der Sicht eines Zuschauers kann eine Stream-Starvation als verzögertes, pufferndes oder einfrierendes Video erscheinen. Die Stream-St arvation kann kurz (weniger als 5 Sekunden) oder lang (mehrere Minuten) sein, abhängig von der spezifischen Situation, die zur Stream-Starvation geführt hat. Weitere Informationen finden Sie unter Was ist Stream-Starvation in den Häufig gestellten Fragen zur Fehlerbehebung.	✓	✓	
Streamer	Eine Person oder ein Gerät, das einen Video- oder Audio-Stream an IVS sendet.	✓	✓	
Subscriber	Ein Teilnehmer eines Echtzeit-Ereignisses, der Video und/oder Audio von Stage-Publishern empfängt. Weitere Informationen finden Sie unter Was ist IVS-Echtzeit-Streaming?.		✓	

Begriff	Beschreibung	LL	RT	Chat
Tag	Ein Tag ist eine Markierung, die Sie einer AWS-Ressource zuordnen. Mithilfe von Tags können Sie Ihre AWS-Ressourcen leichter identifizieren und anordnen. Auf der Startseite der IVS-Dokumentation finden Sie den Abschnitt „Tagging“ in einer beliebigen IVS-API-Dokumentation (für Echtzeit-Streaming, Streaming mit niedriger Latenz oder Chat).	✓	✓	✓
Kamerafilter von Drittanbietern	Softwarekomponenten, die in das IVS Broadcast SDK integriert werden können, damit eine Anwendung Bilder verarbeiten kann, bevor sie dem Broadcast SDK als benutzerdefinierte Image-Quelle bereitgestellt werden. Ein Kamerafilter eines Drittanbieters kann Bilder von der Kamera verarbeiten, einen Filtereffekt anwenden usw.	✓	✓	
Miniaturansicht	Ein verkleinertes Bild, das aus einem Stream aufgenommen wurde. Standardmäßig werden Miniaturansichten alle 60 Sekunden generiert, es kann jedoch ein kürzeres Intervall konfiguriert werden. Die Auflösung der Miniaturansicht hängt vom Kanaltyp ab. Weitere Informationen finden Sie unter Aufzeichnen von Inhalten in Automatische Aufnahme in Amazon S3 (Streaming mit niedriger Latenz).	✓		

Begriff	Beschreibung	LL	RT	Chat
Zeitgesteuerte Metadaten	An bestimmte Zeitstempel innerhalb eines Streams gebundene Metadaten. Dies kann programmg esteuert mithilfe der IVS-API hinzugefügt werden und wird bestimmten Frames zugeordnet. Dadurch wird sichergestellt, dass alle Zuschauer die Metadaten an der gleichen Stelle relativ zum Stream erhalten. Zeitgesteuerte Metadaten können verwendet werden, um Aktionen auf dem Client auszulöse n, z. B. die Aktualisierung von Teamstatistiken während einer Sportveranstaltung. Weitere Informationen finden Sie unter Einbettung von Metadaten in einen Video-Stream.	✓		
Token-Austausch	Eine vom IVS Broadcast SDK bereitgestellte Schnittstelle, mit der Teilnehmer-Token-Funktionen aktualisiert oder herabgestuft und Token-Att ribute aktualisiert werden können, ohne dass die Teilnehmer erneut eine Verbindung herstelle n müssen. Dies ermöglicht Szenarien wie Co-Hosting, bei denen Teilnehmer zunächst nur über Subscribe-Funktionen verfügen und später Veröffentlichungsfunktionen benötigen.		✓	
Transkodierung	Wandelt Video und Audio von einem Format in ein anderes um. Ein eingehender Stream kann in ein anderes Format mit mehreren Bitraten und Auflösungen transkodiert werden, um eine Reihe von Wiedergabegeräten und Netzwerkbedingungen zu unterstützen.	✓	✓	

Begriff	Beschreibung	LL	RT	Chat
Transmuxing	Ein einfaches Umpacken eines aufgenommenen Streams in IVS ohne erneute Kodierung des Videostreams. „Transmux“ ist die Abkürzung für Transcode-Multiplexing, ein Prozess, der das Format einer Audio- und/oder Videodatei ändert und dabei einige oder alle der ursprünglichen Streams beibehält. Transmuxing konvertiert in ein anderes Containerformat, ohne den Dateiinhalt zu ändern. Unterscheidet sich von Transkodierung.	✓	✓	
Varianten-Streams	Eine Reihe von Kodierungen derselben Übertragung in verschiedenen Qualitätsstufen. Jeder Varianten-Stream wird als separate HLS-Wiedergabeliste kodiert. Ein Index der verfügbaren Varianten-Streams wird als multivariante Wiedergabeliste bezeichnet. Nachdem der IVS-Player eine multivariante Playlist von IVS empfangen hat, kann er während der Wiedergabe zwischen den Varianten-Streams auswählen und bei sich ändernden Netzwerkbedingungen nahtlos hin und her wechseln.	✓		
VBR	Variable Bitrate, eine Methode der Ratensteuerung für Encoder, die eine dynamische Bitrate verwendet, die sich während der Wiedergabe je nach erforderlicher Detailebene ändert. Aus Gründen der Videoqualität raten wir nachdrücklich davon ab, VBR zu verwenden. Verwenden Sie stattdessen CBR.	✓	✓	

Begriff	Beschreibung	LL	RT	Chat
Anzeigen	Eine einzelne Anzeigesitzung, die aktiv Mediendat eien herunterlädt oder abspielt. Aufrufe sind die Grundlage für das Kontingent gleichzeitiger Aufrufe. Eine Ansicht beginnt, wenn eine Anzeigesitzung die Videowiedergabe beginnt. Eine Ansicht endet, wenn eine Anzeigesitzung die Videowiedergabe stoppt. Die Wiedergabe ist der einzige Indikator für die Zuschauerschaft; Interaktionsheuristiken wie Audiopegel, Browser-Tab-Fokus und Videoqual ität werden nicht berücksichtigt. Beim Zählen der Aufrufe berücksichtigt IVS nicht die Legitimität einzelner Zuschauer und versucht auch nicht, lokalisierte Zuschauerzahlen zu deduplizieren, z. B. mehrere Videoplayer auf einem einzigen Computer. Weitere Informationen finden Sie unter Andere Kontingente in Service Quotas (Streaming mit niedriger Latenz).	✓		
Zuschauer	Eine Person, die einen Stream von IVS empfängt.	✓		

Begriff	Beschreibung	LL	RT	Chat
WebRTC	Web Real-Time Communication, ein Open-Source-Projekt, das Webbrowsern und mobilen Anwendungen Echtzeitkommunikation bietet. Es ermöglicht die Audio- und Videokommunikation innerhalb von Webseiten, indem es eine direkte Peer-to-Peer-Kommunikation erlaubt, ohne dass Plugins installiert oder native Anwendungen heruntergeladen werden müssen. Die Technologien hinter WebRTC werden als offener Webstandard implementiert und sind als reguläre JavaScript-APIs in allen gängigen Browsern oder als Bibliotheken für native Clients wie Android und iOS verfügbar.	✓	✓	

Begriff	Beschreibung	LL	RT	Chat
WHIP	WebRTC-HTTP Ingestion Protocol, ein HTTP-basiertes Protokoll, das die WebRTC-basierte Aufnahme von Inhalten in Streaming-Services und/oder CDNs ermöglicht. WHIP ist ein IETF-Entwurf, der zur Standardisierung der WebRTC-Aufnahme entwickelt wurde. WHIP ermöglicht die Kompatibilität mit Software wie OBS und bietet eine Alternative (zum IVS-Broadcast-SDK) für Desktop-Publishing. Erfahrenere Streamer, die mit OBS vertraut sind, bevorzugen es möglicherweise aufgrund seiner erweiterten Produktionsfeatures wie Szenenübergänge, Audiomischung und Overlay-Grafiken. WHIP ist auch von Vorteil, wenn die Verwendung des IVS-Broadcast-SDK nicht möglich oder nicht erwünscht ist. Beispielsweise ist das IVS-Broadcast-SDK in Setups mit Hardware-Encodern möglicherweise keine Option. Wenn der Encoder jedoch WHIP unterstützt, können Sie trotzdem direkt vom Encoder in IVS veröffentlichen. Siehe IVS-WHIP-Unterstützung (Echtzeit-Streaming).		✓	
WSS	WebSocket Secure, ein Protokoll zum Einrichten von WebSockets über eine verschlüsselte TLS-Verbindung. Dies wird zum Herstellen einer Verbindung mit IVS-Chat-Endpunkten verwendet. Weitere Informationen finden Sie unter Schritt 4: Senden und Empfangen Ihrer ersten Nachricht in Erste Schritte mit IVS-Chat.			✓

IVS-Dokumentenverlauf | Echtzeit-Streaming

In den folgenden Tabellen werden die wichtigen Änderungen an der Dokumentation für Amazon-IVS-Streaming in Echtzeit beschrieben. Wir aktualisieren die Dokumentation regelmäßig, um neue Versionen zu berücksichtigen und auf Ihr Feedback einzugehen.

Änderungen im Benutzerhandbuch für Echtzeit-Streaming

Änderung	Beschreibung	Datum
Broadcast-SDK: Web 1.33.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise.	12. März 2026
Broadcast-SDK: Android 1.40.0, iOS 1.40.0	Versionsnummer und Artefakt-Links in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	12. März 2026
EventBridge	Für vier Ereignisse (Zielfehler, Sitzungsfehler, Fehler beim Aufzeichnungsstart, Fehler beim Aufzeichungsende) haben wir die Beschreibungen aktualisiert und das Feld `error_code` sowie eine Tabelle mit Fehlercodes und Ursachen hinzugefügt.	27. Februar 2026
Broadcast-SDK: Android 1.39.0, iOS 1.39.0	Versionsnummer und Artefakt-Links in den Broadcast-SDK-Anleitungen zu	13. Februar 2026

	Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise. Für die iOS-Integration wird CocoaPods nicht mehr verwendet. Es wurden entsprechende Änderungen an der Dokumentation vorgenommen unter: • Erste Schritte mit IVS-Streaming in Echtzeit – „Schritt 4: Integration des IVS-Broadcast-SDK“ > „iOS“ • Handbuch zum iOS Broadcast SDK – „Installieren der Bibliothek“	
Broadcast-SDK: Web 1.32.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise.	12. Februar 2026
Service Quotas	TPS-Werte für die drei ParticipantReplication-Operationen (List/Start/Stop) hinzugefügt.	30. Januar 2026
Broadcast-SDK: Android 1.38.0, iOS 1.38.0	Versionsnummer und Artefakt-Links in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	13. Januar 2026

Broadcast-SDK: Android 1.37.1	Versionsnummer und Artefakt-Links in den Broadcast -SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android. Siehe auch Versionshinweise.	11. Dezember 2025
Broadcast SDK – Benutzerdefinierte Audio-Quellen	Diese neue Seite wurde hinzugefügt.	10. Dezember 2025
Teilnehmer-Token-Austausch	Wir haben eine neue Seite Token-Austausch unter IVS Broadcast SDK hinzugefügt. In Verwenden von Amazon EventBridge mit IVS haben wir das Aktualisierungsereignis „Token ausgetauscht“ für IVS Stage Update hinzugefügt In Beispiele: Stage Update haben wir auch ein Beispiel für den Austausch von Token hinzugefügt. API-Änderungen sind in der API-Referenztabelle beschrieben.	9. Dezember 2025
Broadcast-SDK: Web 1.31.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise.	5. Dezember 2025

Broadcast-SDK: Android 1.37.0, iOS 1.37.0	Versionsnummer und Artefakt-Links in den Broadcast -SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	5. Dezember 2025
CloudWatch-Metriken	Unter „Echtzeit-Streaming überwachen > CloudWatch-Metriken“ haben wir zahlreiche neue Metriken für DroppedFrames, PublishBitrate und SubscribeBitrate hinzugefügt. Wir haben auch einige Beschreibungen der Metriken aktualisiert.	18. November 2025
IPR-Synchronisierung	In der Aufzeichnung einzelner Teilnehmer haben wir die Option Aufzeichnungen mehrerer Teilnehmer synchronisieren hinzugefügt.	7. November 2025
Serverseitige Zusammensetzung	Bekannte Probleme und Problemumgehungen wurde hinzugefügt.	30. Oktober 2025
Broadcast-SDK: Web 1.30.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise.	30. Oktober 2025

Broadcast-SDK: Android 1.36.0, iOS 1.36.0	Versionsnummer und Artefakt-Links in den Broadcast -SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise. Wir haben auch neue Abschnitte zum Thema „Nachrichten einbetten“ hinzugefügt: Android und iOS.	30. Oktober 2025
Serverseitige Zusammensetzung	Zusammensetzungslebenszyklus (wenn die Zusammensetzung automatisch heruntergefahren wird) wurde aktualisiert.	27. Oktober 2025
RTMP	Veröffentlichen mit FFmpeg wurde hinzugefügt.	27. Oktober 2025
Kontingent für Zusammensetzungen aktualisieren	Unter „Service Quotas > Andere Kontingente“ haben wir das Kontingent für die „maximale Anzahl gleichzeitiger Zusammensetzungsressourcen pro Konto“ von 5 auf 20 aktualisiert.	14. Oktober 2025
Aktualisierung des Web-Broadcast-SDK	Der Abschnitt zum Veröffentlichen und Abonnieren mit dem Web-Broadcast-SDK von IVS wurde umgeschrieben > Abrufen von WebRTC-Statistiken.	3. Oktober 2025

CloudWatch-Metriken	Für die Metriken `ConcurrentPublishers` und `ConcurrentSubscriptions` wurde die Beschreibung aktualisiert und die Dimension gelöscht.	2. Oktober 2025
Broadcast-SDK: Web 1.29.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise.	2. Oktober 2025
Broadcast-SDK: Android 1.35.0, iOS 1.35.0	Versionsnummer und Artefakt-Links in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	2. Oktober 2025
CloudWatch-Metriken	Es wurden weitere DownloadPacketLoss-Metriken hinzugefügt.	23. September 2025
Benutzerdefinierte Teilnehmeranordnung für serverseitige Zusammensetzung	Es wurden verschiedene Änderungen am Abschnitt Serverseitige Zusammensetzung vorgenommen, u. a. wurden `participantOrderAttribute` und „Benutzerdefinierte Teilnehmeranordnung“ hinzugefügt. API-Änderungen sind in der API-Referenztabelle beschrieben.	16. September 2025

Broadcast-SDK: Android 1.34.0, iOS 1.34.0	Versionsnummer und Artefakt-Links in den Broadcast -SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	11. September 2025
Private Erfassung für Stage	Neuer Abschnitt für die Veröffentlichung von Schnittstellen-VPC-Endpunkten.	10. September 2025
Broadcast-SDK: Web 1.28.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise.	4. September 2025
Broadcast-SDK: iOS	Den neuesten Versionshinweisen für das iOS-Broadcast-SDK (1.33.0) wurde ein 1.32.1 iOS-Broadcast-SDK-Fix hinzugefügt.	21. August 2025
Broadcast-SDK: Web	Updates unter Erste Schritte > Importe, sowohl unter Verwendung eines Skript-Tags als auch unter Verwendung von npm.	8. August 2025
Broadcast-SDK: Web 1.27.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise.	7. August 2025

Broadcast-SDK: Android 1.33.0, iOS 1.33.0	Versionsnummer und Artefakt-Links in den Broadcast -SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise. Im Broadcast iOS SDK-Leitfaden haben wir den Abschnitt „Empfehlung: Integrieren Sie das Player SDK (Swift Package Manager)“ hinzugefügt und die vorhandenen Informationen zur CocoaPods-Integration aktualisiert.	7. August 2025
Broadcast-SDK: Gemischte Geräte	Dies ist ein neues Dokument. (Gemischte Geräte ist sowohl im Benutzerhandbuch für IVS-Low-Latency-Streaming als auch im Benutzerhandbuch für IVS-Real-Time-Streaming identisch.)	28. Juli 2025
Broadcast-SDK: Android 1.32.2	Die Versionsnummer und die Artefakt-Links in der Anleitung zum Broadcast-SDK für Echtzeit-Streaming wurden aktualisiert: Android. Siehe auch Versionshinweise.	25 Juli 2025
Limit für gleichzeitige Teilnehmerreplikationen hinzugefügt	Unter Service Quotas > Andere Quotas haben wir ein Quota für „gleichzeitige Teilnehmerreplikationen“ hinzugefügt.	15. Juli 2025

Broadcast-SDK: Android 1.32.1, iOS 1.32.1	Versionsnummer und Artefakt-Links in den Broadcast -SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	10. Juli 2025
Broadcast-SDK: Web 1.26.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise. Außerdem wurden folgende Änderungen vorgenommen: • Im Abschnitt Umgang mit Netzwerkproblemen wurde das Beispiel aktualisiert. • Im Abschnitt StageErrors wurden Informationen zum Fehler FAILED hinzugefügt, zudem wurden die Fehler STAGE_DISCONNECTED und PARTICIPANT_DISCONNECTED hinzugefügt.	7. Juli 2025

Hinzufügen von Limits für gleichzeitige Herausgeber und Abonnements	Im Abschnitt „Service Quotas“ > Andere Kontingente wurden Kontingente für „gleichzeitige Herausgeber“ und „gleichze itige Abonnements“ hinzugefü gt. Im Abschnitt „Überwachen von Echtzeit-Streaming“ > CloudWatch-Metriken wurden Metriken für Concurren tPublishers und Concurren tSubscriptions hinzugefügt. Im Handbuch für das Web-Broadcast-SDK wurde der Tabelleneintrag STAGE_AT_ CAPACITY unter „Fehlerbe handlung“ > Stagefehler aktualisiert.	23. Juni 2025

Unterstützung für die Erfassung von E-RTMP-Multitrack-Videos	Im Abschnitt „Streamerfassung“ > Unterstützte Protokolle wurde die Unterstützung für E-RTMP-Multitrack-Videos (Enhanced RTMP) hinzugefügt. Im Abschnitt IVS-RTMP-Veröffentlichung wurden Informationen zum Streamen von E-RTMP-Multitrack-Videos mit den Broadcast-SDKs für Echtzeit-Streaming und mit OBS Studio hinzugefügt. Im Abschnitt „Überwachen von Echtzeit-Streaming“ > CloudWatch Metrics wurde die Metrik PublishFrameRate mit den Dimensionen Stage, Participant, SimulcastLayer und MediaType hinzugefügt. Außerdem wurden die Werte der Dimension Simulcast Layer aktualisiert („no-rid“ und „disabled“ wurden durch „none“ ersetzt).	20. Juni 2025

Verschlüsselung mit SSE-S3	Folgenden Abschnitten wurden neue Informationen hinzugefügt: • Aufzeichnung einzelner Teilnehmer – Unter 1. Erstellen eines S3-Buckets • Serverseitige Zusammensetzung – Unter „Erste Schritte mit der serversei tigen Zusammensetzung in IVS“ > Voraussetzungen	19. Juni 2025
Broadcast-SDK: Web 1.25.1	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisi ert: Web. Siehe auch Versionshinweise.	16. Juni 2025
Broadcast-SDK: Web 1.25.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisi ert: Web. Siehe auch Versionshinweise.	12. Juni 2025
Broadcast-SDK: Android 1.31.0, iOS 1.31.0	Versionsnummer und Artefakt-Links in den Broadcast -SDK-Anleitungen zu Echtzeit-Streaming aktualisi ert: Android und iOS. Siehe auch Versionshinweise.	12. Juni 2025

B-Frames in der Streamerfassung	Unter „Streamerfassung“ > Unterstützte Medienspezifikationen wurden die Informationen zu B-Frames aktualisiert.	30. Mai 2025

Teilnehmerreplikation	Erste Veröffentlichung dieser neuen Funktionalität. Sehen Sie sich diese Dokumentationsänderungen an: • Erste Schritte mit Amazon IVS – Unter Schritt 2: Erstellen einer Stage mit optionaler Teilnehmeraufzeichnung wurden die Konsolenanweisungen und die Screenshots aktualisiert. Außerdem wurde der CLI-Antwort für das Erstellen einer Stage mit der Aufzeichnung einzelner Teilnehmer `recordParticipantReplicas` hinzugefügt. • Überwachung von Echtzeit-Streaming – Unter CloudWatch-Metriken: IVS-Echtzeit-Streaming wurde ein Hinweis zur Teilnehmerreplikation hinzugefügt. • EventBridge – Unter Beispiele: Stageaktualisierung wurden zwei Ereignisse hinzugefügt: Start der Teilnehmerreplikation und Ende der Teilnehmerreplikation. Außerdem wurden „Teilnehmer veröffentlicht“, „Teilnehmer nicht veröffentlicht“ und „Fehler bei Teilnehme	29. Mai 2025

	rveröffentlichung“ aktualisiert. • Teilnehmerreplikation – Dieses neue Dokument enthält eine Übersicht und CLI-Anweisungen. • Kosten – Dieses neue Dokument enthält die Kosten im Zusammenhang mit dem IVS-Echtzeit-Streaming. API-Änderungen sind in der API-Referenztabelle beschrieben.	
Broadcast-SDK: Android 1.30.1	Die Versionsnummer und die Artefakt-Links in der Anleitung zum Broadcast-SDK für Echtzeit-Streaming wurden aktualisiert: Android. Siehe auch Versionshinweise.	26. Mai 2025
Broadcast-SDK: Web 1.24.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise.	15. Mai 2025
Broadcast-SDK: Android 1.30.0, iOS 1.30.0	Versionsnummer und Artefakt-Links in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	15. Mai 2025

Broadcast-SDK: Web 1.23.1	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise.	2. Mai 2025
Broadcast-SDK: Web 1.23.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise. Wir haben auch zusätzliche Informationen und Beispiele zu „Konfigurieren mehrschichtiger Kodierung (Publisher)“ im Leitfaden zum Web-Broadcast-SDK hinzugefügt.	17. April 2025
Broadcast-SDK: Android 1.29.0, iOS 1.29.0	Versionsnummer und Artefakt-Links in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise. Wir haben auch zusätzliche Informationen und Beispiele zu „Konfigurieren mehrschichtiger Kodierung (Publisher)“ in den Broadcast SDK-Handbüchern für Android und iOS hinzugefügt.	17. April 2025

Service Quotas	Es wurde ein Kontingent für „Zusammensetzungen pro Stage“ hinzugefügt.	2. April 2025
Streaming-Optimierungen	In der Einleitung wurde ein Absatz zur Konfiguration der maximalen Bitrate, Framerate und Auflösung sowie zur Konfiguration der Einstellung zur Leistungsreduzierung (auf Mobilgeräten) hinzugefügt.	21. März 2025
Broadcast-SDK: Web 1.22.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise. Außerdem wurde in der Einleitung ein weiteres Beispiel mit Beispielcode hinzugefügt (Einfache Wiedergabe). Unter „SEI (Supplemental Enhancement Information, Ergänzende Informationen zur Verbesserung) > Einfügen von SEI-Nutzdaten“ wurde ein Hinweis zur Speichernutzung hinzugefügt. Und unter „Bekannte Probleme und Problemumgehungen“ wurde ein Artikel zu `stage.leave()` hinzugefügt.	20. März 2025

Broadcast-SDK: Android 1.28.1, iOS 1.28.1	Versionsnummer und Artefakt-Links in den Broadcast -SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	20. März 2025
Broadcast-SDK: Android 1.27.2, iOS 1.27.2	Versionsnummer und Artefakt-Links in den Broadcast -SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	19. März 2025
Dauer des Zielsegments	Einige Screenshots unter „Erste Schritte mit IVS-Echtzeit-Streaming > Schritt 2: Erstellen einer Stage mit optionaler Teilnehmeraufzeichnung“ wurden aktualisiert und zeigen nun das neue Feld „Dauer des Zielsegments“.	13. März 2025
Aufzeichnung einzelner Teilnehmer	Der Abschnitt Konvertieren von Aufzeichnungen in MP4 wurde hinzugefügt.	7. März 2025

Zusammenfügen der Aufzeichnungen einzelner Teilnehmer	Erste Veröffentlichung dieser neuen Funktionalität. Sehen Sie sich diese Dokumentationsänderungen an: • Erste Schritte mit Amazon IVS – Die Konsolen- und CLI-Anleitungen unter Schritt 2: Erstellen einer Stage mit optionaler Teilnehmeraufzeichnung wurden aktualisiert. • Aufzeichnung einzelner Teilnehmer – Der Abschnitt Zusammenführen fragmentierter Aufzeichnungen einzelner Teilnehmer wurde hinzugefügt. • EventBridge – Unter Beispiele: Statusänderung der Aufzeichnung einzelner Teilnehmer wurden Informationen zur Konstruktion von S3-Präfixen für den Fall hinzugefügt, dass die Zusammenführung für die Aufzeichnung einzelner Teilnehmer aktiviert ist.	6. März 2025
WHIP-Anforderungen	In der Einleitung wurde die Anforderung hinzugefügt, dass WHIP-Clients 307-Umleitungen verarbeiten können müssen.	5. März 2025

Broadcast-SDK: iOS 1.27.1	Die Versionsnummer und die Artefakt-Links in der Anleitung zum Broadcast-SDK für Echtzeit-Streaming wurden aktualisiert: iOS. Siehe auch Versionshinweise.	03. März 2025
Broadcast-SDK: Web 1.21.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisi ert: Web. Siehe auch Versionshinweise.	20. Februar 2025
Broadcast-SDK: Android 1.27.0, iOS 1.27.0	Versionsnummer und Artefakt-Links in den Broadcast -SDK-Anleitungen zu Echtzeit-Streaming aktualisi ert: Android und iOS. Siehe auch Versionshinweise.	20. Februar 2025
Broadcast-SDK: Android 1.26.0, iOS 1.26.0	Versionsnummer und Artefakt-Links in den Broadcast -SDK-Anleitungen zu Echtzeit-Streaming aktualisi ert: Android und iOS. Siehe auch Versionshinweise.	30. Januar 2025

Broadcast-SDK: Web 1.20.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise. Wir haben auch „Zusätzliche erweiterte Informationen“ aktualisiert.	23. Januar 2025
RTMP-Veröffentlichung	In Veröffentlichen mit einem RTMP-Encoder haben wir darauf hingewiesen, dass Streams sowohl Audio- als auch Videospuren enthalten müssen, da sie sonst getrennt werden.	21. Januar 2025
Netzwerkanforderungen	Diese neue Top-Level-Seite wurde hinzugefügt.	21. Januar 2025
Streaming-Optimierungen	Der Abschnitt „Konfigurieren mehrschichtiger Kodierung mit Simulcast“ wurde in „Konfigurieren mehrschichtiger Kodierung mit Simulcast (Publisher)“ umbenannt und der Abschnitt „Konfigurieren mehrschichtiger Kodierung mit Simulcast (Subscriber)“ hinzugefügt.	12. Dezember 2024

Broadcast-SDK: Web 1.19.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise. Außerdem haben wir den Abschnitt „Mehrschichtige Kodierung mit Simulcast“ und drei Simulcast-Elemente unter „Ereignisse“ hinzugefügt.	12. Dezember 2024
Broadcast-SDK: Android 1.25.0, iOS 1.25.0	Versionsnummer und Artefakt-Links in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise. In allen Leitfäden haben wir außerdem: • den Abschnitt „Abrufen von SEI-Daten (Supplemental Enhancement Information)“ hinzugefügt; • den Abschnitt „Mehrschichtige Kodierung mit Simulcast“ hinzugefügt; • drei Simulcast-Elemente unter „Renderer“ hinzugefügt; • den Abschnitt „Aktivieren/Deaktivieren der mehrschichtigen Kodierung mit Simulcast“ gelöscht.	12. Dezember 2024

Thumbnail-Konfiguration in Echtzeit	Unter Aufzeichnung einzelner Teilnehmer und Zusammengesetzte Aufzeichnung haben wir Beispiele und Informationen zu JSON-Metadaten aktualisiert und Preisinformationen hinzugefügt. Unter Aufzeichnung einzelner Teilnehmer wurde die Option „Nur Thumbnail-Aufzeichnungen“ hinzugefügt.	10. Dezember 2024
Broadcast-SDK: Android 1.24.0, iOS 1.24.0	Versionsnummer und Artefakt-Links in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	13. November 2024
Kamerafilter von Drittanbietern	Zahlreiche Änderungen unter Verwendung von Snap mit dem IVS Broadcast SDK > Web	12. November 2024
Broadcast-SDK: Web 1.18.0	Versionsnummer und Artefakt-Links in der Broadcast-SDK-Anleitung zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise. Wir haben dem SDK-Leitfaden den neuen Abschnitt Abrufen von SEI-Daten (Supplemental Enhancement Information) hinzugefügt.	12. November 2024

RTMP	Aktualisiertes Beispiel unter „Erfassungskonfiguration erstellen“ (`--ingest-protocol` hinzugefügt)	7. November 2024
Broadcast-SDK: Web 1.17.0	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und im Broadcast-SDK-Leitfaden zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise.	10. Oktober 2024
Broadcast-SDK: Android 1.23.0, iOS 1.23.0	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und in den Broadcast-SDK-Leitfäden zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise. Für Android haben wir Informationen zum Verwenden des SDK mit Debug-Symbolen hinzugefügt.	10. Oktober 2024
Service Quotas	Wir haben ein Kontingent für „Bitrate für Teilnehmerveröffentlichung“ hinzugefügt.	25. September 2024
Überwachen von IVS-Echtzeit-Streaming	Die CloudWatch-Metrik `PublishFramerate` wurde hinzugefügt.	13. September 2024

Broadcast-SDK: Web 1.16.0	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und im Broadcast-SDK-Leitfaden zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise.	11. September 2024
Broadcast-SDK: Android 1.22.0, iOS 1.22.0	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und in den Broadcast-SDK-Leitfäden zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise. Wir haben auch den Abschnitt „Erste Schritte > Bibliothek installieren“ in der Broadcast-SDK-Anleitung für Android aktualisiert.	11. September 2024

RTMP-Erfassung	Wir haben die Seite IVS-Streamerfassung hinzugefügt. Darunter befinden sich zwei Seiten, RTMP (neu) und WHIP. In Verwenden von EventBridge mit IVS-Echtzeit-Streaming haben wir das Aktualisierungsereignis „Fehler bei Teilnehmerveröffentlichung“ für IVS-Stage hinzugefügt In Service Quotas haben wir TPS-Werte für die fünf neuen API-Vorgänge und ein neues IngestConfiguration-Kontingent (unter „Andere Kontingente“) hinzugefügt. API-Änderungen sind in der API-Referenztabelle beschrieben.	9. September 2024
Optimierungen für Echtzeit-Streaming	Es wurden verschiedene Simulcast-bezogene Aktualisierungen vorgenommen und „Auflösung von Layern“ hinzugefügt.	22. August 2024
Veröffentlichen/Abonnieren in der Konsole	In Erste Schritte mit IVS-Echtzeit-Streaming haben wir unter Video veröffentlichen und abonnieren Informationen zum Veröffentlichen und Abonnieren von Videos in der Konsole hinzugefügt.	19. August 2024

Broadcast-SDK: Web 1.15.0	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und im Broadcast-SDK-Leitfaden zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise. Außerdem haben wir dem Leitfaden zum Web-Broadcast-SDK den neuen Abschnitt Konfiguration für das Abonnieren von Teilnehmern hinzugefügt. Unter Streaming-Optimierungen haben wir den neuen Abschnitt Ändern der Mindestverzögerung des Jitter-Puffers für Subscriber hinzugefügt. Dieser enthält Informationen zu den Web-, Android- und iOS-Broadcast-SDKs.	15. August 2024

Broadcast-SDK: Android 1.21.0, iOS 1.21.0	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und in den Broadcast-SDK-Leitfäden zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise. Außerdem haben wir den Broadcast-SDK-Leitfäden für Android und iOS den neuen Abschnitt „Konfiguration für das Abonnieren von Teilnehmern“ hinzugefügt.	15. August 2024
Erklärung zur Aufzeichnung	Unter Aufzeichnung einzelner Teilnehmer (in 1: S3-Bucket erstellen) und Zusammengesetzte Aufzeichnung (in Voraussetzungen, Schritt 3) wurde ein Hinweis zur Verwendung eines vorhandenen S3-Buckets hinzugefügt. Für die Einstellung Objekteigentümerschaft muss entweder Bucket-Eigentümer erzwungen oder Bucket-Eigentümer bevorzugt aktiviert sein.	13. August 2024
Broadcast-SDK: Web 1.14.0	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und im Broadcast-SDK-Leitfaden zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise.	18. Juli 2024

Broadcast-SDK: Android 1.20.0, iOS 1.20.0	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und in den Broadcast-SDK-Leitfäden zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	18. Juli 2024
Erste Schritte mit Echtzeit-Streaming	Informationen zu Attributen zu „Verteilen von Teilnehmertoken“ wurden sowohl in „Token-Schema: Nutzdaten“ als auch in „Erstellen von Token mit der IVS-Echtzeit-Streaming-API“ hinzugefügt.	12. Juli 2024
Service Quotas	Die maximale Anzahl von Stage-Subscribern wurde von 10 000 auf 25 000 erhöht.	27. Juni 2024
Generieren von Teilnehmer-Token mit einem Schlüsselpaar	In Erste Schritte mit IVS-Echtzeit-Streaming haben wir Verteilen von Teilnehmer-Token aktualisiert, um die beiden Methoden zur Generierung von Token (API und Schlüsselpaar) zu erläutern, und haben „Erstellen von Token mit einem Schlüsselpaar“ hinzugefügt.	26. Juni 2024

Aufzeichnung einzelner Teilnehmer	Für Aufzeichnung einzelner Teilnehmer (neu) und „Zusammengesetzte Aufzeichnung“ (bereits vorhanden) wurde ein neuer Dokumentationsabschnitt mit Unterdokumenten zu Aufzeichnung hinzugefügt. Außerdem haben wir Ereignisse und Beispiele für „Statusänderung der Aufzeichnung einzelner Teilnehmer“ zu Verwenden von EventBridge mit IVS-Echtzeit-Streaming hinzugefügt. API-Änderungen sind in der API-Referenztabelle beschrieben.	20. Juni 2024
Service Quotas	Das Stagekontigent wurde von 100 auf 1 000 erhöht.	14. Juni 2024
Broadcast-SDK: Web 1.13.0	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und im Broadcast-SDK-Leitfaden zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise. Im Leitfaden wurden die Informationen zur Fehlerbehandlung für das neue Stageereignis `ERROR` aktualisiert.	13. Juni 2024

Broadcast-SDK: Android 1.19.0, iOS 1.19.0	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und in den Broadcast-SDK-Leitfäden zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	13. Juni 2024
Broadcast-SDK: Web 1.12.0	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und im Broadcast-SDK-Leitfaden zu Streaming mit niedriger Latenz aktualisiert: Web. Siehe auch Versionshinweise. Im Leitfaden wurden in Umgang mit Netzwerkproblemen die Informationen zum Stageverbindungsstatus `ERROR` aktualisiert.	20. Mai 2024
Optimierungen für Echtzeit-Streaming	Unter Standard-Layers, Eigenschaften und Framerates wurde die maximale Bitrate für Mobilgeräte, der untere Layer, von 150 000 in 100 000 BpS geändert.	16. Mai 2024
Broadcast-SDK: Android 1.18.0, iOS 1.18.0	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und in den Broadcast-SDK-Leitfäden zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	16. Mai 2024

Broadcast-SDK: Web 1.11.0	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und im Broadcast-SDK-Leitfaden zu Echtzeit-Streaming aktualisiert: Web. Siehe auch Versionshinweise.	6. Mai 2024
Broadcast-SDK: Web 1.10.1	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und im Broadcast-SDK-Leitfaden zu Echtzeit-Streaming aktualisiert: Web. Siehe auch Versionshinweise.	30. April 2024
Broadcast-SDK: Android 1.15.2, iOS 1.15.2	Versionsnummer und Artefakt-Links auf der Startseite der IVS-Dokumentation und in den Broadcast-SDK-Leitfäden zu Echtzeit-Streaming aktualisiert: Android und iOS. Siehe auch Versionshinweise.	30. April 2024
Broadcast-SDK: iOS-Leitfaden	In Veröffentlichen eines Medienstreams haben wir das Codebeispiel aktualisiert.	26. April 2024

Broadcast-SDK: Android 1.17.0, iOS 1.17.0	Aktualisierte Versionsnummern und Artefakt-Links für die neue Version in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming: Android und iOS. Auf der Startseite für die Amazon-IVS-Dokumentation wurden die Broadcast-SDK-Referenzlinks mit Verweisen auf die neue Version aktualisiert. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version.	22. April 2024
Serverseitige Zusammensetzung	In SSC wurden verschiedene Änderungen vorgenommen, insbesondere im Bereich „Layout“, um PiP- und Raster-Layouts zu erklären. Im Leitfaden zum Web-Broadcast-SDK wurde Serverseitige Rendering-Unterstützung hinzugefügt.	26. März 2024
OBS- und WHIP-Unterstützung	Es wurde ein Hinweis zu Qualitätsproblemen (wie zeitweiliges Einfrieren des Videos) hinzugefügt, die bei WHIP in OBS auftreten können.	22. März 2024

Broadcast-SDK: Android 1.16.0, iOS 1.16.0, Web 1.10.0	Versionsnummer und Artefakt-Links für das neue Release in den Broadcast-SDK-Leitfäden zu Echtzeit-Streaming aktualisiert: Android, iOS und Web. Auf der Startseite für die Amazon-IVS-Dokumentation wurden die Broadcast-SDK-Referenzlinks mit Verweisen auf die neue Version aktualisiert. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version.	21. März 2024
Broadcast-SDK: Android 1.15.1, iOS 1.15.1	Aktualisierte Versionsnummern und Artefakt-Links für die neue Version in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming: Android und iOS. Auf der Startseite für die Amazon-IVS-Dokumentation wurden die Broadcast-SDK-Referenzlinks mit Verweisen auf die neue Version aktualisiert. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version.	13. März 2024

Broadcast-SDK: Mobile Audiomodi	Unter „Voreinstellungen für den Audio-Modus“ wurden Informationen zur Voreinstellungskategorie „Lautstärkenregler“ und ein bekanntes iOS-Problem mit der Voreinstellung „Videochat“ hinzugefügt. Unter „Fortschrittliche Anwendungsfälle“ wurden ein Hinweis zur Vermeidung falscher Konfigurationen hinzugefügt und die Abschnitte „iOS-Echounterdrückung“ und „Benutzerdefinierte iOS-Audio-Quellen“ hinzugefügt.	1. März 2024
Broadcast-SDK: Android 1.15.0, iOS 1.15.0, Web 1.9.0	Versionsnummer und Artefakt-Links für das neue Release in den Broadcast-SDK-Leitfäden zu Echtzeit-Streaming aktualisiert: Android, iOS und Web. Auf der Startseite für die Amazon-IVS-Dokumentation wurden die Broadcast-SDK-Referenzlinks mit Verweisen auf die neue Version aktualisiert. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version.	22. Februar 2024

OBS- und WHIP-Unterstützung	Eine neue Seite wurde hinzugefügt. In diesem Dokument wird erklärt, wie Sie WHIP-kompatible Encoder wie OBS verwenden, um in IVS-Echtzeit-Streaming zu veröffentlichen. WHIP (WebRTC-HTTP Ingestion Protocol) ist ein IETF-Entwurf, der zur Standardisierung der WebRTC-Erfassung entwickelt wurde.	6. Februar 2024

Broadcast-SDK: Android 1.14.1, iOS 1.14.1, Web 1.8.0	Versionsnummer und Artefakt-Links für das neue Release in den Broadcast-SDK-Leitfäden zu Echtzeit-Streaming aktualisiert: Android, iOS und Web. Auf der Startseite für die Amazon-IVS-Dokumentation wurden die Broadcast-SDK-Referenzlinks mit Verweisen auf die neue Version aktualisiert. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version. Für den Android-Leitfaden haben wir ein neues bekanntes Problem hinzugefügt (Videogröße unter als 176 × 176). Für den Web-Leitfaden haben wir ein neues bekanntes Problem hinzugefügt. Das Problem lässt sich umgehen, indem die Videoauflösung beim Aufrufen von `getUserMedia` oder `getDisplayMedia` auf 720p beschränkt wird. In Optimierungen für IVS-Echtzeit-Streaming haben wir Konfiguration der Codierung von Layern mit Simulcast aktualisiert. Diese Option ist jetzt standardmäßig deaktiviert.	1. Februar 2024

Broadcast-SDK: Android 1.13.4, iOS 1.13.4, Web 1.7.0	Versionsnummer und Artefakt-Links für das neue Release in den Broadcast-SDK-Leitfäden zu Echtzeit-Streaming aktualisiert: Android, iOS und Web. Auf der Startseite für die Amazon-IVS-Dokumentation wurden die Broadcast-SDK-Referenzlinks mit Verweisen auf die neue Version aktualisiert. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version.	3. Januar 2024
IVS-Glossar	Das Glossar wurde um IVS-Begriffe in Echtzeit, niedriger Latenz und Chat erweitert.	20. Dezember 2023
Stage-Zustand: Neue CloudWatch-Metriken	Die Metrik PacketLoss (Stage) wurde in DownloadPacketLoss (Stage) umbenannt und zusätzliche CloudWatch-Metriken für IVS-Echtzeit-Streaming veröffentlicht: • DownloadPacketLoss (Stage, Teilnehmer) • DroppedFrames (Stage, Teilnehmer) • SubscribeBitrate (Stage, Teilnehmer, Medientyp) Siehe Überwachen von IVS-Echtzeit-Streaming.	07. Dezember 2023

IAM-verwaltete Richtlinien	Es wurden zwei verwaltete Richtlinien hinzugefügt: IVSReadOnlyAccess und IVSFullAccess. Siehe: • Der neue Abschnitt zu Verwaltete Richtlinien für Amazon IVS auf der Seite Sicherheit. • Änderungen an Schritt 3: Einrichten von IAM-Berechtigungen in Erste Schritte mit IVS-Streaming mit niedriger Latenz.	05. Dezember 2023
Broadcast-SDK: Android 1.13.2, iOS 1.13.2	Aktualisierte Versionsnummern und Artefakt-Links für die neue Version in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming: Android und iOS. Auf der Startseite für die Amazon-IVS-Dokumentation wurden die Broadcast-SDK-Referenzlinks mit Verweisen auf die neue Version aktualisiert. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version.	4. Dezember 2023

Broadcast-SDK: Android 1.13.1	Aktualisierte Versionsnummern und Artefakt-Links für die neue Version in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming: Android. Auf der Startseite für die Amazon-IVS-Dokumentation wurden die Broadcast-SDK-Referenzlinks mit Verweisen auf die neue Version aktualisiert. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version.	21. November 2023
Service Quotas	Die „Auflösung der Veröffentlichung durch Teilnehmer“ wurde von 1080 p auf 720 p geändert.	18. November 2023

Broadcast-SDK: Android 1.13.0, iOS 1.13.0	Aktualisierte Versionsnummern und Artefakt-Links für die neue Version in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming: Android und iOS. Auf der Startseite für die Amazon-IVS-Dokumentation wurden die Broadcast-SDK-Referenzlinks mit Verweisen auf die neue Version aktualisiert. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version. Es wurden außerdem verschiedene Aktualisierungen der Streaming-Optimierungen vorgenommen. Unter anderem erfordert das Feature „Adaptives Streaming: Ebenen-Codierung mit Simulcast“ jetzt eine ausdrückliche Zustimmung und wird nur in neueren Versionen des SDK unterstützt.	17. November 2023

Serverseitige Zusammensetzung (SSC)	Mit der serverseitigen IVS-Zusammensetzung können Clients die Zusammensetzung und Übertragung einer IVS-Stage an einen von IVS verwalteten Service verlagern. SSC- und RTMP-Übertragungen an einen Kanal werden über IVS-Steuerebenen-Endpunkte in der Heimatregion der Stage aufgerufen. Siehe: • Erste Schritte – Der Richtlinie wurden unter „IAM-Berechtigungen einrichten“ SSC-Endpunkte hinzugefügt. • Verwendung von Amazon EventBridge mit IVS – Es wurden neue Metriken hinzugefügt. • Serverseitige Zusammensetzung – Dieses neue Dokument enthält eine Übersicht und Anweisungen zur Einrichtung. • Service Quotas – Es wurden neue Anrufratenlimits und andere Kontingente hinzugefügt. Lesen Sie auch: • Unten aufgelistete Änderungen in der API-	16. November 2023

	Referenz-Änderungen zu IVS-Echtzeit-Streaming. • Aufgelistete Änderungen im Dokumentverlauf (Streaming mit niedriger Latenz).	
Zusammengesetzte Aufzeichnung	Durchführung der folgenden Änderungen: • Für dieses neue Feature wurde eine Seite für zusammengesetzte Aufzeichnungen hinzugefügt. • Aktualisierte Erste Schritte mit IVS-Echtzeit-Streaming mit S3-Endpunkten in der Richtlinie unter „IAM-Berechtigungen einrichten“. • Aktualisierte Service Quotas mit Anrufratenkontingenten für die neuen Endpunkte.	16. November 2023
IVS-Broadcast-SDK	In der Broadcast-SDK-Übersicht wurde „Plattformanforderungen“ > „Native Plattformen“ aktualisiert, um klarzustellen, welche SDK-Versionen unterstützt werden. Außerdem wurden „Mobile Browser (iOS und Android)“ hinzugefügt. Im Broadcast-Web-Leitfaden wurden „Einschränkungen für das mobile Web“ hinzugefügt.	9. November 2023

IVS-Broadcast-SDK	Eine neue Seite zu Kamerafiltern von Drittanbietern wurde hinzugefügt.	9. November 2023
Erste Schritte mit IVS-Echtzeit-Streaming	Wir haben die Verfahren im Abschnitt IAM-Berechtigungen einrichten aktualisiert.	20. Oktober 2023
Überwachen von Echtzeit-Streaming	In CloudWatch-Metriken: IVS-Echtzeit-Streaming haben wir Beispielwerte für Dimensionen hinzugefügt.	17. Oktober 2023
Broadcast-SDK: Web-Leitfaden	Wir haben mehrere Änderungen an Überwachen des Medien-Stummschaltungsstatus von Remote-Teilnehmern vorgenommen.	17. Oktober 2023

Broadcast-SDK: Web 1.6.0	Aktualisierte Versionsnummern und Artefakt-Links für die neue Version in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming: Web. Die Startseite der Amazon-IVS-Dokumentation verweist auf die aktuelle Version von Broadcast-SDK-Referenzen. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version. Im Web Guide haben wir unter „Einen MediaStream von einem Gerät abrufen“ die beiden `max`-Zeilen ebenfalls gelöscht. Es hat sich bewährt, nur `ideal` anzugeben. Unter Echtzeit-Streaming -Optimierungen haben wir einen neuen Abschnitt hinzugefügt: Optimieren der Audio-Bitrate und der Stereo-Unterstützung.	16. Oktober 2023
Stage-Zustand: Neue CloudWatch-Metriken	CloudWatch-Metriken für IVS-Echtzeit-Streaming veröffentlicht. Siehe Überwachen von IVS-Echtzeit-Streaming.	12. Oktober 2023

Broadcast-SDK: Android 1.12.1	Aktualisierte Versionsnummern und Artefakt-Links für die neue Version in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming: Android. Außerdem wurde ein neuer Abschnitt hinzugefügt: Verwenden von Bluetooth-Mikrofonen. Die Startseite der Amazon-IVS-Dokumentation verweist auf die aktuelle Version von Broadcast-SDK-Referenzen. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version.	12. Oktober 2023
Broadcast-SDK: Web 1.5.2	Aktualisierte Versionsnummern und Artefakt-Links für die neue Version in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming: Web. Die Startseite der Amazon-IVS-Dokumentation verweist auf die aktuelle Version von Broadcast-SDK-Referenzen. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version.	14. September 2023
Erste Schritte mit IVS-Echtzeit-Streaming	Unter Android > Broadcast-SDK installieren wurde Datenbindung hinzugefügt.	12. September 2023

Broadcast-SDK-Fehlerbehandlung	Die Abschnitte „Fehlerbehandlung“ wurden den Broadcast-SDK-Handbüchern hinzugefügt: Web, Android und iOS.	12. September 2023
Erste Schritte mit IVS-Echtzeit-Streaming	In Verteilen von Teilnehmertoken, wurde ein Wichtig-Hinweis hinzugefügt, der darüber informiert, dass Funktionen nicht auf dem aktuellen Tokenformat basieren.	1. September 2023
Erste Schritte mit IVS-Echtzeit-Streaming	In IAM-Berechtigungen einrichten wurde der Satz von Berechtigungen aktualisiert.	31. August 2023
Broadcast-SDK: Web 1.5.1, Android 1.12.0, und iOS 1.12.0	Aktualisierte Versionsnummern und Artefakt-Links für die neue Version in den Broadcast-SDK-Anleitungen zu Echtzeit-Streaming: Web, Android und iOS. Auf der Startseite für die Amazon-IVS-Dokumentation wurden die Broadcast-SDK-Referenzlinks mit Verweisen auf die neue Version aktualisiert. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version.	23. August 2023

Streaming-Start in Echtzeit	Diese Version enthält wichtige Änderungen an der Dokumentation. Wir haben die vorherige Dokumentation in IVS-Streaming mit niedriger Latenz umbenannt und eine neue Dokumentation zu IVS-Echtzeit-Streaming veröffentlicht. Die Landingpage zur IVS-Dokumentation hat jetzt separate Abschnitte für Echtzeit-Streaming und Streaming mit niedriger Latenz. Jeder Abschnitt hat sein eigenes Benutzerhandbuch und eine eigene API-Referenz. Weitere Änderungen an der Dokumentation finden Sie unter Dokumentenverlauf (Streaming mit niedriger Latenz).	7. August 2023

Broadcast-SDK: Web 1.5.0, Android 1.11.0, und iOS 1.11.0	Aktualisierte Versionsnummern und Artefakt-Links für die neue Version in den Broadcast-SDK-Anleitungen: Web, Android und iOS. Auf der Startseite für die Amazon-IVS-Dokumentation wurden die Broadcast-SDK-Referenzlinks mit Verweisen auf die neue Version aktualisiert. Sehen Sie auch die Versionshinweise von Amazon IVS für diese Version.	7. August 2023

Referenzänderungen an der API von IVS-Echtzeit-Streaming

API-Änderungen	Beschreibung	Datum
Vorgang „StartParticipantReplication“ aktualisieren	In „StartParticipantReplication“ wurde der Maximalwert für das Feld `reconnectWindowSeconds` von 60 auf 300 geändert.	16. März 2026
Teilnehmer-Token-Austausch	Wir haben das Objekt ExchangedParticipantToken hinzugefügt. Dem Event-Objekt wurden die Felder `newToken` und `previousToken` hinzugefügt. `TOKEN_EXCHANGED` als gültigen Wert für das Feld `name` im Event-Objekt hinzugefügt. Die Änderungen im Benutzerhandbuch werden in der Tabelle des Benutzerhandbuchs beschrieben.	9. Dezember 2025

API-Änderungen	Beschreibung	Datum
Benutzerdefinierte Teilnehmeranordnung für serverseitige Zusammensetzung	Die Datentypen GridConfiguration und PipConfiguration wurden geändert (das Feld `participantOrderAttribute` wurde hinzugefügt), was sich wiederum auf das Composition-Objekt auswirkt. Dies wirkt sich auf die StartComposition-Anfrage und -Antwort sowie auf die GetComposition-Antwort aus. Die Änderungen im Benutzerhandbuch werden in der Tabelle des Benutzerhandbuchs beschrieben.	16. September 2025

API-Änderungen	Beschreibung	Datum
Teilnehmerreplikation	Erste Veröffentlichung dieser neuen Funktionalität. Sehen Sie sich diese Dokumentationsänderungen an: • Unter „Wichtige Konzepte“ wurde spezifisc he Terminologie für die Teilnehmerreplikation hinzugefügt. • Drei neue Vorgänge wurden hinzugefügt: ListParti cipantReplicas, StartParticipantReplication und StopParticipantReplication. • Ein neues Objekt wurde hinzugefügt: Participa ntReplica. • Das Feld `recordParticipantReplicas` wurde dem Objekt AutoParticipantRecordingCon figuration hinzugefügt. Dies betrifft die CreateSta ge-Anforderung und -Antwort, die GetStage-Antwort sowie die UpdateStage-Anforderung und -Antwort. • Dem Event-Objekt wurden drei Felder hinzugefü gt: `destinationStageArn` , `destinati onSessionId` und `replica`. Dies wirkt sich auf die ListParticipantEvents-Antwort aus. • Den Objekten Participant und ParticipantSummary wurden vier Felder hinzugefügt: `replicati onState` , `replicationType` , `sourceSta geArn` und `sourceSessionId` . Dies wirkt sich auf die GetParticipant- und ListParticipants-A ntworten aus.	29. Mai 2025

API-Änderungen	Beschreibung	Datum
	• Unter „Ereignis“ wurden dem Feld `name` zwei gültige Werte hinzugefügt: `REPLICATION_STARTED` und `REPLICATION_STOPPED` . Die Änderungen im Benutzerhandbuch werden in der Tabelle des Benutzerhandbuchs beschrieben.	
Dauer des Zielsegments	Das Objekt AutoParticipantRecordingConfiguration wurde geändert (das Feld `hlsConfiguration` wurde hinzugefügt). Das wirkt sich wiederum auf das Stage-Objekt aus. Dies betrifft die CreateStage-Anforderung und -Antwort, die GetStage-Antwort sowie die UpdateStage-Anforderung und -Antwort. Das Objekt RecordingConfiguration wurde geändert (das Feld `hlsConfiguration` wurde hinzugefügt). Dies wirkt sich auf die GetComposition-Antwort sowie die StartComposition-Anfrage und -Antwort aus. Zwei Objekte wurden hinzugefügt: CompositionRecordingHlsConfiguration und ParticipantRecordingHlsConfiguration.	13. März 2025
Zusammenfügen der Aufzeichnungen einzelner Teilnehmer	Das Objekt AutoParticipantRecordingConfiguration wurde geändert (das Feld `recordingReconnectWindowSeconds` wurde hinzugefügt). Das wirkt sich wiederum auf das Stage-Objekt aus. Dies betrifft die CreateStage-Anforderung und -Antwort, die GetStage-Antwort sowie die UpdateStage-Anforderung und -Antwort. Die Beschreibung von `Participant.recordingS3Prefix` wurde aktualisiert.	6. März 2025

API-Änderungen	Beschreibung	Datum
Thumbnail-Konfiguration in Echtzeit	Das Objekt S3DestinationConfiguration wurde geändert: `thumbnailConfigurations` hinzugefügt. Dies wirkt sich auf die GetComposition-Antwort sowie die StartComposition-Anfrage und -Antwort aus. Das Objekt AutoParticipantRecordingConfiguration wurde geändert: `thumbnailConfiguration` hinzugefügt und `NONE` als gültiger Wert für `mediaTypes` hinzugefügt. Dies betrifft die CreateStage-Anforderung und -Antwort, die GetStage-Antwort sowie die UpdateStage-Anforderung und -Antwort. Zwei Objekte wurden hinzugefügt: CompositionThumbnailConfiguration und ParticipantThumbnailConfiguration.	10. Dezember 2024
Aktualisieren von Ereignis- und Videoobjekten	Im Ereignisobjekt haben wir weitere gültige Werte für `errorCode` hinzugefügt. Im Video-Objekt haben wir klargestellt, dass `height` und `width` gerade Zahlen sein müssen.	2. Oktober 2024

API-Änderungen	Beschreibung	Datum
RTMP-Erfassung	Wir haben zwei Objekte hinzugefügt: IngestConfiguration und IngestConfigurationSummary. Wir haben fünf IngestConfiguration-Endpunkte (Erstellen, Löschen, Abrufen, Auflisten und Aktualisieren) hinzugefügt. Wir haben DeleteStage (Beschreibung des Vorgangs) und DisconnectParticipant (Beschreibungen des Vorgangs und `participantId`) aktualisiert. Wir haben das Participant-Objekt geändert (das Feld `protocol` hinzugefügt). Dies wirkt sich auf die GetParticipant-Antwort aus. Wir haben das StageEndpoints-Objekt geändert (die Felder `rtmp` und `rtmps` hinzugefügt). Dies wirkt sich auf die CreateStage-, GetStage- und UpdateStage-Antworten aus. Wir haben auch die Beschreibung dieses Objekts aktualisiert (eine Caching-Empfehlung hinzugefügt).	9. September 2024
Generieren von Teilnehmer-Token mit einem Schlüsselpaar	Wir haben drei Objekte (PublicKey, PublicKeySummary, StageEndpoints) und vier Endpunkte hinzugefügt: (DeletePublicKey, GetPublicKey, ImportPublicKey, ListPublicKeys). Wir haben das Stage-Objekt geändert (das Feld `endpoints` hinzugefügt). Dies wirkt sich auf die CreateStage-, GetStage- und UpdateStage-Antworten aus.	26. Juni 2024
Aufzeichnung einzelner Teilnehmer	Wir haben ein Objekt hinzugefügt (AutoParticipantRecordingConfiguration) und drei Objekte geändert (Participant, ParticipantSummary, Stage). Dies betrifft fünf Endpunkte: CreateStage-Anforderung und -Antwort, GetParticipant-Antwort, GetStage-Antwort, ListParticipants-Anforderung und -Antwort sowie UpdateStage-Anforderung und -Antwort.	20. Juni 2024

API-Änderungen	Beschreibung	Datum
Entfernen von `svs` aus ARN-Mustern	ARN-Muster, die `[is]vs` angegeben haben, wurden aktualisiert, um `ivs` anzugeben. Dies betrifft alle drei Tag-Endpunkte und das Feld `ChannelDestinationConfiguration$channelArn` .	25. April 2024
Aktualisierungen der serverseitigen Zusammensetzung	Wir haben ein Objekt hinzugefügt: PipConfiguration. Wir haben zwei Objekte geändert (LayoutConfiguration, GridConfiguration). Dies wirkt sich auf die GetComposition-Antwort sowie die StartComposition-Anfrage und -Antwort aus.	13. März 2024
Zusammengesetzte Aufzeichnung	Es wurden 4 StorageConfiguration-Endpunkte und 7 Objekte hinzugefügt (DestinationDetail, RecordingConfiguration, S3DestinationConfiguration, S3Detail, S3StorageConfiguration, StorageConfiguration, StorageConfigurationSummary). Es wurden 3 Objekte geändert (Composition, Destination, DestinationConfiguration). Dies wirkt sich auf die GetComposition-Antwort sowie die StartComposition-Anfrage und -Antwort aus.	16. November 2023
Serverseitige Zusammensetzung	Es wurden 8 Composition- und EncoderConfiguration-Endpunkte sowie 11 Objekte hinzugefügt (ChannelDestinationConfiguration, Composition, CompositionSummary, Destination, DestinationConfiguration, DestinationSummary, EncoderConfiguration, EncoderConfigurationSummary, GridConfiguration, LayoutConfiguration und Video).	16. November 2023
Stage-Zustand: Neue Teilnehmerdaten	Dem Objekt Teilnehmer wurden sechs Felder hinzugefügt: `browserName` , `browserVersion` , `ispName`, `osName`, `osVersion` , und `sdkVersion` . Dies wirkt sich auf die GetParticipant-Antwort aus.	12. Oktober 2023

API-Änderungen	Beschreibung	Datum
Teilnehmertoken	Es wurde ein Wichtig-Hinweis hinzugefügt, der darüber informiert, dass Funktionen nicht auf dem aktuellen Tokenformat basieren.	1. September 2023
Start von IVS-Echtzeit-Streaming	Diese Version enthält wichtige Änderungen an der Dokumentation. Wir haben die vorherige Dokumentation in IVS-Streaming mit niedriger Latenz umbenannt und eine neue Dokumentation zu IVS-Echtzeit-Streaming veröffentlicht. Die Landingpage zur IVS-Dokumentation hat jetzt separate Abschnitte für Echtzeit-Streaming und Streaming mit niedriger Latenz. Jeder Abschnitt hat sein eigenes Benutzerhandbuch und eine eigene API-Referenz. Die Referenz zur API von IVS-Echtzeit-Streaming- ist Teil der IVS-Echtzeit-Streaming-Dokumentation. Zuvor trug sie den Titel Stage-API-Referenz für IVS. Ihre Vorgeschichte ist beschrieben in Dokumentenverlauf (Streaming mit niedriger Latenz).	7. August 2023

IVS-Versionshinweise | Echtzeit-Streaming

Dieses Dokument enthält alle Versionshinweise zu Amazon-IVS-Streaming in Echtzeit, beginnend mit den neuesten, geordnet nach dem Datum ihrer Veröffentlichung.

12. März 2026

IVS-Broadcast-SDK: Web 1.33.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.33.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Die `exchangeToken` -Methode für den Token-Austausch in Echtzeit wurde implementiert. • Das Ereignis „STAGE_PARTICIPANT_METADATA_CHANGED“ wurde implementiert. Es wird ausgelöst, wenn sich `attributes` und/oder `userId` nach dem Token-Austausch ändern. • Feld `encoderImplementation` wurde zur Methode `requestQualityStats()` des lokalen Stufen-Streams der Anfrage hinzugefügt.

12. März 2026

Amazon IVS Broadcast SDK: Android 1.40.0, iOS 1.40.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.40.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.40.0/android/ • Fehlermeldungen im Zusammenhang mit Fehlern bei der Validierung von TLS-Zertifikaten wurden verbessert und Fehlerauf zählungscodes wurden erweitert.
iOS-Broadcast-SDK 1.40.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.40.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.40.0/ios/ • Fehlermeldungen im Zusammenhang mit Fehlern bei der Validierung von TLS-Zertifikaten wurden verbessert und Fehlerauf zählungscodes wurden erweitert.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,823 MB	14,139 MB
armeabi-v7a	5,046 MB	9,798 MB
x86_64	5,935 MB	14,702 MB

Architektur	Komprimierte Größe	Unkomprimierte Größe
86 x	6,190 MB	15,265 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,939 MB	8,013 MB

13. Februar 2026

Amazon IVS Broadcast SDK: Android 1.39.0, iOS 1.39.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.39.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.39.0/android/ • Es wurden kleinere Fehlerbehebungen für die erneute Verbindung von Bluetooth -Headsets im STUDIO-Audiomodus vorgenommen. • Wichtige Entwicklungstools für Android und die NDK-Version aktualisiert. • Seltener Deadlock beim Anhalten eines `MixedImageDevice` behoben.
iOS-Broadcast-SDK 1.39.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.39.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.39.0/ios/

Plattform	Downloads und Änderungen
	• Xcode wurde auf Version 26.2 aktualisiert. • Mit dieser Version werden die IVS-SDKs nicht mehr über CocoaPods verteilt. CocoaPods kündigte seine Einstellung im Jahr 2024 an und wird im Laufe dieses Jahres in den schreibgeschützten Status übergehen. Swift Package Manager (SPM) ersetzt CocoaPods als von Apple unterstützte Lösung für das Abhängigkeitsmanagement und dient als Standardmethode zur Integration von SDKs in moderne Xcode-Projekte. Wir empfehlen Ihnen, auf SPM umzusteigen oder die IVS-SDK-Frameworks direkt in Ihr Projekt zu integrieren. IVS SDKs werden durch beide Ansätze vollständig unterstützt. Es wurden entsprechende Änderungen an der Dokumentation vorgenommen unter: • Erste Schritte mit IVS-Streaming in Echtzeit – „Schritt 4: Integration des IVS-Broadcast-SDK“ > „iOS“ • Handbuch zum iOS Broadcast SDK – „Installieren der Bibliothek“

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,788 MB	14,059 MB
armeabi-v7a	5,016 MB	9,740 MB

Architektur	Komprimierte Größe	Unkomprimierte Größe
x86_64	5,898 MB	14,615 MB
86 x	6,154 MB	15,184 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,932 MB	7,996 MB

12. Februar 2026

IVS-Broadcast-SDK: Web 1.32.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.32.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Fehlerbehebungen und Stabilitätsverbesserungen.

13. Januar 2026

Amazon IVS Broadcast SDK: Android 1.38.0, iOS 1.38.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.38.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.38.0/android/

Plattform	Downloads und Änderungen
	• Funktion zur Priorisierung von Schichten – Das Enum `Priority` wurde `StageVideoConfiguration.Layer` mit folgenden Werten hinzugefügt: `VERY_LOW`, `LOW`, `MEDIUM` und `HIGH`. Dadurch wird bestimmt, welche Schicht bei Einschränkungen der Netzwerkbandbreite zuerst entfällt. • Schnellere Wiederverbindung von Stages nach Wiederherstellung der Netzwerkkonnektivität. • Die mit einigen Fehlern verbundenen Codes wurden geändert. Weitere Informationen finden Sie weiter unten im Leitfaden zur Fehlermigration in Mobile-Broadcast-SDKs.

Plattform	Downloads und Änderungen
iOS-Broadcast-SDK 1.38.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.38.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.38.0/ios/ • Funktion zur Priorisierung von Schichten – Das Enum `IVSLocalStageStrea mLayerPriority` wurde mit folgenden Werten hinzugefügt: `VeryLow`, `Low`, `Medium` und `High`. Dadurch wird bestimmt, welche Schicht bei Einschränkungen der Netzwerkb andbreite zuerst entfällt. • Schnellere Wiederverbindung von Stages nach Wiederherstellung der Netzwerkk onnektivität. • Die mit einigen Fehlern verbundenen Codes wurden geändert. Weitere Informationen finden Sie weiter unten im Leitfaden zur Fehlermigration in Mobile-Broadcast-SDKs.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,795 MB	14,070 MB
armeabi-v7a	5,021 MB	9,746 MB
x86_64	5,904 MB	14,630 MB
86 x	6,161 MB	15,198 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,609 MB	8,078 MB

Leitfaden zur Fehlermigration in Mobile-Broadcast-SDKs

In Version 1.38.0 der iOS- und Android-Broadcast-SDKs haben sich die Codes für einige Fehler geändert. Bislang gab es keine einzelne Eigenschaft, mit der von SDKs ausgelöste Fehler eindeutig bestimmt werden konnten. Zum Verständnis der Bedeutung eines Fehlers musste stattdessen eine Kombination der folgenden Eigenschaften untersucht werden:

Android	iOS

`BroadcastException.getCode()`
 	
`BroadcastException.getUid()`
 	
`BroadcastException.getError()`
 	
`BroadcastException.getSource()`
 	
`BroadcastException.getDetail()`
	`NSError.code`
	
	`NSError.userInfo[IVSBroadcastUidDescriptionErrorKey]`
	
	`NSError.userInfo[IVSBroadcastResultDescriptionErrorKey]`
	
	`NSError.userInfo[IVSBroadcastSourceDescriptionErrorKey]`
	
	`NSError.userInfo[NSLocalizedDescriptionKey]`

Ab Version 1.38.0 und höher geben `BroadcastException.getCode()` (Android) und `NSError.code` (iOS) eine eindeutige ID zurück, die in den öffentlichen Enumerationen von `BroadcastErrorCode` (Android) und `IVSBroadcastErrorCode` (iOS) nachgeschlagen werden kann.

Zusätzlich dazu, dass `code` zur eindeutigen ID für alle Fehler gemacht wurde, wurde ein zusätzliches Feld hinzugefügt: `BroadcastException.getPlatformCode()` (Android) und `NSError.userInfo[IVSBroadcastPlatformCodeDescriptionErrorKey]` (iOS). Wenn

ein Fehler durch die zugrunde liegende Plattform verursacht wird (z. B. ein Netzwerkfehler oder ein Fehler bei der Codierung oder Decodierung von Videos), ist dieses Feld ungleich null. Mit dessen Hilfe können zusätzliche Informationen aus der Dokumentation der Plattform gesammelt werden.

Migrieren von SDK 1.37.0 und früher

Damit alle Fehler der neuen Strategie entsprechen, mussten die Werte einiger vorhandener Fehler geändert werden. Nachfolgend finden Sie eine Anleitung dazu, wie Sie die vorhandene Logik der neuen Logik zuordnen können:

- Bei Fehlern, bei denen `code` ungleich null war, wird der gleiche Wert für den Code beibehalten. Die Referenzierung des Codes über die neuen Enum-Konstanten kann jedoch zu mehr Klarheit führen. Zum Beispiel ist der Vergleich eines Fehlers mit `BroadcastErrorCode.Broadcast.LatencyThresholdReached` klarer als der Vergleich mit `20401`.
- Bei Fehlern, bei denen `UID` ein Wert hatte (d. h. nicht `-1` in Android oder `"-1"` in iOS war), wird das Feld `code` jetzt auf den vorhandenen `UID`-Wert gesetzt. Bei Bedingungen, die das Feld `UID` vergleichen, können Sie die Konstanten beibehalten, sie aber künftig mit dem Feld `code` vergleichen.
- Einige ältere Fehler enthielten keinen `code`- oder `UID`-Wert. Diese wurden in der Regel auf Grundlage des `message` (Android) oder der `description` (iOS) des Fehlers abgeglichen, was wegen der dynamischen Natur von Fehlermeldungen keine zuverlässige Methode zur Bestimmung von Fehlern ist. Da diese Fehler keine eindeutigen Identifizierungsmerkmale aufwiesen, können keine Eins-zu-Eins-Zuordnungen bereitgestellt werden. Bei den meisten Fehlern wurde jedoch die gleiche Beschreibung beibehalten. Deshalb ist es möglich, die gleiche Abgleichlogik weiter zu verwenden und zugleich den neuen `code`-Wert für künftige App-Versionen zu erfassen und zu melden.

Als konkretes Beispiel sollte die Fehlerprüfung in der folgenden Tabelle wie folgt migriert werden:

Vor	Nach
`error.code == 20401`	`error.code == BroadcastErrorCode .Broadcast.LatencyThreshold Reached`

Vor	Nach
	Keine Änderung, aber den Vergleich mit dem Enum-Wert bevorzugen.
`error.uid == 207`	`error.code == BroadcastErrorCode.Net.SocketRemoteHangup` Mit `code` statt mit `uid` vergleichen.
`error.message.contains("Ice ConnectionFailed")`	`error.code == BroadcastErrorCode.RealTime.PeerConnectionIce ConnectionFailed` Nicht mit `message` (bzw. `source` oder `result/detail`) vergleichen. Stattdessen den entsprechenden Enum-Code suchen, mit dem verglichen werden soll.

Der wichtigste Teil eines Fehlers ist nach wie vor `BroadcastException.getPlatformCode()` (Android) und `NSError.userInfo[IVSBroadcastPlatformCodeDescriptionErrorKey]` (iOS). Doch in Version 1.38.0 und höher identifiziert das Feld `code` Fehler eindeutig und ermöglicht das sofortige Nachschlagen des Fehlernamens und der Fehlerbeschreibung in den Enumerationen von `BroadcastErrorCode` (Android) und `IVSBroadcastErrorCode` (iOS). Daher sollten andere Felder wie `UID`, `source` und `detail` nicht in der Suchlogik verwendet werden; sie dienen nur der ergänzenden Information.

11. Dezember 2025

Amazon IVS Broadcast SDK: Android 1.37.1 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.37.1	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.37.1/android/

Plattform	Downloads und Änderungen
	• Probleme im Zusammenhang mit dem Teardown der Teilnehmervorschau wurden behoben.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,754 MB	13,965 MB
armeabi-v7a	4,991 MB	9,683 MB
x86_64	5,858 MB	14,529 MB
86 x	6,128 MB	15,120 MB

9. Dezember 2025

Teilnehmer-Token-Austausch

Die neue Unterstützung für den Austausch von Teilnehmer-Token ermöglicht es Ihnen, die Token-Funktionen zu aktualisieren oder herunterzustufen und die Token-Attribute innerhalb des IVS-Client-SDK zu aktualisieren, ohne dass Clients die Verbindung trennen und erneut herstellen müssen. Dies ist nützlich für Szenarien wie Co-Hosting, bei denen Teilnehmer zunächst nur über Subscribe-Funktionen verfügen und später Veröffentlichungsfunktionen benötigen.

Weitere Informationen finden Sie auf der neuen Seite auf Token-Austausch.

5. Dezember 2025

IVS-Broadcast-SDK: Web 1.31.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.31.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Fehlerbehebungen und Stabilitätsverbesserungen.

5. Dezember 2025

Amazon IVS Broadcast SDK: Android 1.37.0, iOS 1.37.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.37.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.37.0/android/ • Fehlerbehebungen und Stabilitätsverbesserungen. • Unterstützung für den Austausch von Teilnehmer-Token hinzugefügt.
iOS-Broadcast-SDK 1.37.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.37.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.37.0/ios/ • Fehlerbehebungen und Stabilitätsverbesserungen.

Plattform	Downloads und Änderungen
	• Unterstützung für den Austausch von Teilnehmer-Token hinzugefügt.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,753 MB	13,961 MB
armeabi-v7a	4,990 MB	9,680 MB
x86_64	5,857 MB	14,525 MB
86 x	6,127 MB	15,116 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,588 MB	8,028 MB

7. November 2025

Synchronisation der Aufzeichnungen einzelner Teilnehmer

Die neue Unterstützung für `EXT-X-PROGRAM-DATE-TIME`-Tags in HLS-Wiedergabelisten, die einzelne Teilnehmer aufnehmen, ermöglicht die präzise Synchronisation von Aufzeichnungen mehrerer Teilnehmer während der Nachbearbeitung. Dieses Feature bietet millisekundengenaue UTC-Zeitstempel zu Beginn und zu Unterbrechungen der Aufzeichnung, sodass Sie synchronisierte Zusammensetzungen (z. B. nebeneinander oder Bild-in-Bild-Layouts) erstellen können, selbst wenn es zu Netzwerkunterbrechungen kommt oder die Teilnehmer zu unterschiedlichen Zeiten beitreten. In der Aufzeichnung einzelner Teilnehmer haben wir die Option Aufzeichnungen mehrerer Teilnehmer synchronisieren hinzugefügt.

30. Oktober 2025

IVS-Broadcast-SDK: Web 1.30.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.30.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Fehlerbehebungen und Stabilitätsverbesserungen.

30. Oktober 2025

Amazon IVS Broadcast SDK: Android 1.36.0, iOS 1.36.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.36.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.36.0/android/ • Die Kamerafunktion wurde verbessert, wenn sie nach einer längeren Zeit im Hintergrund wieder in den Vordergrund zurückkehrt. • Es wurde eine `embedMessage` -Methode in `ImageDevice` hinzugefügt, um das Einfügen von Metadaten-Nutzdaten in einen veröffentlichten Videostream zu ermöglichen. Weitere Informationen finden Sie unter Nachrichten einbetten im Android Broadcast SDK-Handbuch.
iOS-Broadcast-SDK 1.36.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.36.0/AmazonIVSBroadcast-Stages.xcframework.zip

Plattform	Downloads und Änderungen
	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.36.0/ios/ • Es wurde eine `embedMessage` -Methode in `IVSImageDevice` hinzugefügt, um das Einfügen von Metadaten-Nutzdaten in einen veröffentlichten Videostream zu ermöglichen. Weitere Informationen finden Sie unter Nachrichten einbetten im iOS Broadcast SDK-Handbuch.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,736 MB	13,898 MB
armeabi-v7a	4,974 MB	9,638 MB
x86_64	5,839 MB	14,456 MB
86 x	6,109 MB	15,047 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,569 MB	7,962 MB

14. Oktober 2025

Neues Echtzeitlimit: Zusammensetzungen

Wir haben das Kontingent für die „maximale Anzahl gleichzeitiger Zusammensetzungsressourcen pro Konto“ von 5 auf 20 aktualisiert. Sie ist unter Service Quotas > Andere Quotas dokumentiert.

2. Oktober 2025

IVS-Broadcast-SDK: Web 1.29.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.29.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Fehlerbehebungen und Stabilitätsverbesserungen.

2. Oktober 2025

Amazon IVS Broadcast SDK: Android 1.35.0, iOS 1.35.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.35.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.35.0/android/ • Fehlerbehebungen und Stabilitätsverbesserungen.
iOS-Broadcast-SDK 1.35.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.35.0/AmazonIVSBroadcast-Stages.xcframework.zip

Plattform	Downloads und Änderungen
	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.35.0/ios/ • `IVSImageDevice.setOnFrameCallback` kann jetzt mit einer `DispatchQueue` angepasst werden. Außerdem kann es optional den mit dem Bild verknüpften `CVPixelBuffer` enthalten.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,730 MB	13,900 MB
armeabi-v7a	4,971 MB	9,639 MB
x86_64	5,835 MB	14,455 MB
86 x	6,104 MB	15,041 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,569 MB	7,963 MB

16. September 2025

Benutzerdefinierte Teilnehmeranordnung für die serverseitige Zusammensetzung

Die neue Unterstützung der benutzerdefinierten Teilnehmeranordnung für die serverseitige Zusammensetzung ermöglicht eine detaillierte Steuerung der Teilnehmerpositionierung im Raster-

und im PiP-Layout (PiP). Weitere Informationen finden Sie unter Serverseitige Zusammensetzung (verschiedene Änderungen, darunter das Hinzufügen von `participantOrderAttribute` und von „Benutzerdefinierte Teilnehmeranordnung“) und in der Referenz zur API von IVS-Echtzeit-Streaming (`participantOrderAttribute` wurde dem Composition-Objekt hinzugefügt).

11. September 2025

Amazon IVS Broadcast SDK: Android 1.34.0, iOS 1.34.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.34.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.34.0/android/ • CPU-Verbesserungen für den Transport von veröffentlichten und abonnierten Medien. • `packetsLost` wurde `LocalVideoStats` und `LocalAudioStats` hinzugefügt.
iOS-Broadcast-SDK 1.34.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.34.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.34.0/ios/ • CPU-Verbesserungen für den Transport von veröffentlichten und abonnierten Medien. • `packetsLost` wurde `IVSLocalVideoStats` und `IVSLocalAudioStats` hinzugefügt. • Es wurde ein Fehler behoben, durch den Geräte nach dem Verlassen einer Stage nicht getrennt wurden, was zur unerwarteten

Plattform	Downloads und Änderungen
	Anzeige von Datenschutzindikatoren führen konnte.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,796 MB	14,089 MB
armeabi-v7a	5,036 MB	9,788 MB
x86_64	5,906 MB	14,653 MB
86 x	6,174 MB	15,240 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,594 MB	8,046 MB

10. September 2025

Schnittstellen-VPC-Endpunkte

Dank der neuen Unterstützung für Schnittstellen-VPC-Endpunkte (Virtual Private Cloud) können Sie für Workloads, die eine sichere Live-Video-Erfassung erfordern, eine sichere private Verbindung zwischen Ihrer Amazon-VPC und IVS herstellen. Dadurch bleibt Ihr IVS-Erfassungsverkehr innerhalb des AWS-Netzwerks und außerhalb des öffentlichen Internets. Schnittstellen-VPC-Endpunkte werden über AWS PrivateLink bereitgestellt, eine AWS-Technologie, die eine private Kommunikation zwischen AWS-Services über eine Elastic-Network-Schnittstelle mit privaten IP-Adressen in Ihrer Amazon-VPC ermöglicht. Weitere Informationen finden Sie unter Private Erfassung im Benutzerhandbuch für IVS-Streaming mit niedriger Latenz und unter Private Erfassung für Stage im Benutzerhandbuch zum ISV-Echtzeit-Streaming.

4. September 2025

IVS-Broadcast-SDK: Web 1.28.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.28.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Bei Beitritt zu einer Stage, die gelöscht wurde, oder mit einem Teilnehmer-Token, dessen Verbindung unterbrochen wurde, werden jetzt `STAGE_DELETED` oder `STAGE_DISCONNECTED` -Fehler statt `TIMEOUT` angezeigt. • Interne Polling-Anfragen im Zusammenhang mit Simulcast wurden optimiert.

7. August 2025

IVS-Broadcast-SDK: Web 1.27.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.27.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • `requestQualityStats` wurde zu `RemoteStageStream` hinzugefügt, womit ein vereinfachtes Objekt mit Video- und Audiostatistiken von `requestRTCStats` zur Verfügung gestellt wird. • Aktualisierungen, um sicherzustellen, dass der `RemoteStageStream` -Stummsch

Plattform	Downloads und Änderungen
	altstatus und der aktivierte Status von `mediaStreamTrack` immer synchron sind.

7. August 2025

Amazon IVS Broadcast SDK: Android 1.33.0, iOS 1.33.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen

Android-Broadcast-SDK 1.33.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.33.0/android/
	• Neue Methoden zur Steuerung des Geräte-Torch:
	  • `CameraSource.Capabilities` implementiert `isTorchSupported` .
	  • `CameraSource.Options.Builder` implementiert `setEnableTorch` .
	• Das Android-Broadcast-SDK erfüllt die Kompatibilitätsanforderungen von Google Play für eine Seitengröße von 16 KB. (Hinweis: Dies wurde ab Version 1.23.0 des SDK implementiert.)

iOS-Broadcast-SDK 1.33.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.33.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.33.0/ios/ • Neue Methode zur Steuerung des Geräte-Torch: `IVSImageDevice` implementiert

Plattform	Downloads und Änderungen
	zwei Eigenschaften, `isTorchSupported` und `torchEnabled` . Überprüfen Sie mit `isTorchSupported` , ob das Gerät Torch unterstützt, und schalten Sie es dann durch Einstellen von `torchEnabled` um. • Es wurde ein Problem unter iOS 18.5+ mit bestimmten VPNs behoben, das zu Timeouts bei der Peer-Verbindung führen konnte. (Hinweis: Dies wurde ab Version 1.32.1 des SDK implementiert.)

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,689 MB	13,829 MB
armeabi-v7a	4,962 MB	9,649 MB
x86_64	5,806 MB	14,413 MB
86 x	6,066 MB	14,983 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,505 MB	7,828 MB

25 Juli 2025

Amazon IVS Broadcast SDK: Android 1.32.2 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.32.2	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.32.2/android/ • IPv6 für `Stage`-Verbindungen deaktiviert.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,693 MB	13,838 MB
armeabi-v7a	4,964 MB	9,653 MB
x86_64	5,810 MB	14,422 MB
86 x	6,067 MB	14,988 MB

23. Juli 2025

Durchsetzung neuer Echtzeit-Metriken und -Grenzwerte: Gleichzeitige Herausgeber und Abonnements

Am 23. Juni haben wir zwei neue einstellbare Servicekontingente eingeführt, welche die maximale Anzahl gleichzeitiger Veröffentlicher und gleichzeitiger Abonnements über alle Stages in einer AWS-Region festlegen. Wir beginnen heute mit der Durchsetzung dieser neuen Kontingente.

15. Juli 2025

Neues Echtzeitlimit: Gleichzeitige Teilnehmerreplikationen

Wir haben eine neue, nicht anpassbare Service Quota für die maximale Anzahl gleichzeitiger Replikationen pro Teilnehmer über alle Stages in einer AWS-Region eingeführt. Sie ist unter Service Quotas > Andere Quotas dokumentiert.

10. Juli 2025

Amazon IVS Broadcast SDK: Android 1.32.1, iOS 1.32.1 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.32.1	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.32.1/android/ • `StageAudioConfiguration.enableEchoCancellation()` wurde entfernt. Aktivieren oder deaktivieren Sie die Echounterdrückung stattdessen mit `StageAudioManager` . • Die Voreinstellungen `STUDIO` und `SUBSCRIBE_ONLY` in `StageAudioManager` wurden geändert, um die Echounterdrückung auszuschalten. Wenn Sie `STUDIO` mit Echounterdrückung verwenden möchten, legen Sie zuerst die Voreinstellung fest und aktivieren Sie dann die Echounterdrückung, um die Standardeinstellung von `STUDIO` außer Kraft zu setzen. • Es wurde eine `MixedDevice` -API-Suite hinzugefügt, mit der mehrere Bild- und Audioquellen in einem einzigen

Plattform	Downloads und Änderungen
	Ausgabe-`Device` zusammengefügt werden können. Damit lassen sich komplexere Audios und Bilder auf einer Stage veröffent lichen.
iOS-Broadcast-SDK 1.32.1	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.32.1/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.32.1/ios/ • Es wurde eine `IVSMixedDevice` - API-Suite hinzugefügt, mit der mehrere Bild- und Audioquellen in einem einzigen Ausgabe-`IVSDevice` zusammengefügt werden können. Damit lassen sich komplexere Audios und Bilder auf einer Stage veröffent lichen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,692 MB	13,840 MB
armeabi-v7a	4,965 MB	9,655 MB
x86_64	5,810 MB	14,424 MB
86 x	6,068 MB	14,990 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,508 MB	7,900 MB

7. Juli 2025

IVS-Broadcast-SDK: Web 1.26.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.26.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • `requestQualityStats` wurde zu LocalStageStream hinzugefügt, womit ein vereinfachtes Objekt mit Video- und Audiostatistiken aus RequestRTCStats zur Verfügung gestellt wird. • Es wurden Websocket-Leaks behoben, die während der Einrichtung auftreten konnten und zu späteren Join-Fehlern führten. • Es wurde ein Problem behoben, bei dem fälschlicherweise 1302-Fehler auftraten, wenn ein fehlgeschlagener Abonnierungs- oder Veröffentlichungsvorgang wiederholt wurde. • Die Stabilität der Wiederholungsversuche für Verbindungen zum Abonnieren und Veröffent lichen wurde verbessert, wenn sich die Join-Verbindung im Status ERRORED oder CONNECTING befindet.

23. Juni 2025

Neue Echtzeit-Metriken und Limits: gleichzeitige Publisher und Abonnements

Es wurden zwei neue anpassbare Servicekontingente für die Höchstzahl gleichzeitiger Publisher und Abonnements für alle Stages in einer AWS-Region eingeführt. Sie sind unter „Service Quotas“ > Andere Kontingente dokumentiert. Diese Kontingente verleihen Ihnen mehr Kontrolle über die Gesamtnutzung Ihres Kontos. Bislang hatte IVS nur Limits für die Anzahl der Publisher und Abonnenten pro Stage durchgesetzt. Dies erschwerte das Treffen von Sicherheitsvorkehrungen auf Kontoebene und konnte zu einer höheren Nutzung und zu mehr Kosten als erwartet führen, vor allem bei Kunden, die viele Stages erstellen.

Hinweis: Die Durchsetzung dieser neuen Kontingente beginnt am 23. Juli. So haben Sie 30 Tage Zeit, Ihre Nutzung zu überprüfen und bei Bedarf eine Erhöhung der Servicekontingente zu beantragen.

Außerdem wurden zwei neue CloudWatch-Metriken hinzugefügt: `ConcurrentPublishers` und `ConcurrentSubscriptions`. Mithilfe dieser Metriken können Sie die Nutzung für alle Stages überwachen und beurteilen, ob Sie sich den Standardlimits nähern. Sie sind im Abschnitt „Überwachen von Echtzeit-Streaming“ > CloudWatch Metrics dokumentiert. Sie sollten CloudWatch-Alarme einrichten, um sich benachrichtigen zu lassen, wenn sich die Nutzung einem Kontingentlimit nähert.

20. Juni 2025

Unterstützung für die Erfassung von E-RTMP-Multitrack-Videos

Sie können E-RTMP-Multitrack-Videos (Enhanced Real-Time Messaging Protocol) verwenden, um Videos unterschiedlicher Qualität an die IVS-Stages zu senden. Dieses Feature ermöglicht das Streaming mit adaptiver Bitrate, sodass Zuschauer die Videos in der für ihre jeweilige Netzwerkverbindung besten Qualität ansehen können. Weitere Informationen finden Sie im Abschnitt E-RTMP-Multitrack-Videos in der Dokumentation zur IVS-RTMP-Veröffentlichung.

16. Juni 2025

IVS-Broadcast-SDK: Web 1.25.1 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.25.1	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Die unbeabsichtigte Durchsetzung der Engine von NPM (Version 22) wurde entfernt. Alle LTS-Knotenversionen werden unterstützt, wenn das Paket transpiliert wird.

12. Juni 2025

Amazon IVS Broadcast SDK: Android 1.31.0, iOS 1.31.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.31.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.31.0/android/ • Fehlerbehebungen und Stabilitätsverbesserungen.
iOS-Broadcast-SDK 1.31.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.31.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.31.0/ios/ • Fehlerbehebungen und Stabilitätsverbesserungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,579 MB	13,594 MB
armeabi-v7a	4,864 MB	9,473 MB
x86_64	5,697 MB	14,173 MB
86 x	5,951 MB	14,724 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,431 MB	7,732 MB

12. Juni 2025

IVS-Broadcast-SDK: Web 1.25.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.25.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Es wurde ein Fehler behoben, bei dem SEI-Nachrichten möglicherweise nicht gesendet wurden, wenn bei einem Remote-Teilnehmer der Status `ERROR` auftrat. • Es wurde ein Fehler behoben, bei dem bei Aufruf des Stageereignisses `STAGE_STREAM_MUTE_CHANGED` möglicherweise mehrere Remote-Stagestreams zurückgegeben wurden.

Plattform	Downloads und Änderungen
	• Es wurde ein Fehler behoben, bei dem `STAGE_PARTICIPANT_STREAMS_REMOVED` bei fehlerhaften Streams nicht aufgerufen wurde.

29. Mai 2025

Teilnehmerreplikation

Mithilfe der Teilnehmerreplikation können Sie einen Teilnehmer von einer Stage in eine andere kopieren. Das ist nützlich, wenn ein und derselbe Teilnehmer auf mehreren Stages gleichzeitig erscheinen soll, um Stageübergreifende Interaktionen zu ermöglichen. Änderungen an der Dokumentation finden Sie im Dokumentverlauf (im Benutzerhandbuch und in den API-Referenztabellen).

26. Mai 2025

Amazon IVS Broadcast SDK: Android 1.30.1 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.30.1	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.30.1/android/ • Es wurde ein Fehler behoben, bei dem die Mikrofonlautstärke auf einigen Android-Geräten zu niedrig war, wenn SDK-verwaltete Mikrofone aus `DeviceDiscovery` mit der Audiovoreinstellung `STUDIO` verwendet wurden.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,579 MB	13,592 MB
armeabi-v7a	4,863 MB	9,472 MB
x86_64	5,696 MB	14,171 MB
86 x	5,950 MB	14,722 MB

15. Mai 2025

IVS-Broadcast-SDK: Web 1.24.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.24.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Speicherlecks beim Verlassen und Wiederbetreten einer Stage wurden behoben.

15. Mai 2025

Amazon IVS Broadcast SDK: Android 1.30.0, iOS 1.30.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.30.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.30.0/android/ • Fehlerbehebungen und Stabilitätsverbesserungen.

Plattform	Downloads und Änderungen
iOS-Broadcast-SDK 1.30.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.30.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.30.0/ios/ • Fehlerbehebungen und Stabilitätsverbesserungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,571 MB	13,577 MB
armeabi-v7a	4,857 MB	9,462 MB
x86_64	5,691 MB	14,156 MB
86 x	5,944 MB	14,708 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,430 MB	7,732 MB

2. Mai 2025

IVS-Broadcast-SDK: Web 1.23.1 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.23.1	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Es wurde ein Problem behoben, bei dem der Teilnehmerbeitritt immer vor der Auflösung von `join()` erfolgte. • Es wurde ein Problem behoben, bei dem lokale Teilnehmer bei Verlassen und Wiederbeitritt in kurzer Folge fälschlic herweise als Remote-Teilnehmer gemeldet wurden.

17. April 2025

Amazon IVS Broadcast SDK: Android 1.29.0, iOS 1.29.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.29.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.29.0/android/ • Ein Feature wurde zur Simulcast-Publishe r-Steuerung hinzugefügt. Weitere Informati onen finden Sie unter „Konfigurieren mehrschichtiger Kodierung (Publisher)“ im Android Broadcast SDK-Handbuch. • Fehlerbehebungen und Stabilitätsverbess erungen.

Plattform	Downloads und Änderungen
iOS-Broadcast-SDK 1.29.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.29.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.29.0/ios/ • Ein Feature wurde zur Simulcast-Publisher-Steuerung hinzugefügt. Weitere Informationen finden Sie unter „Konfigurieren mehrschichtiger Kodierung (Publisher)“ im iOS Broadcast SDK-Handbuch. • Fehlerbehebungen und Stabilitätsverbesserungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,566 MB	13,546 MB
armeabi-v7a	4,853 MB	9,444 MB
x86_64	5,681 MB	14,119 MB
86 x	5,939 MB	14,674 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,429 MB	7,715 MB

17. April 2025

IVS-Broadcast-SDK: Web 1.23.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.23.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Ein Feature wurde zur Simulcast-Publisher-Steuerung hinzugefügt. Weitere Informationen finden Sie unter „Konfigurieren mehrschichtiger Kodierung (Publisher)“ im Web Broadcast SDK-Handbuch. • Die Latenz bis zur Veröffentlichung wurde verbessert. Dies wirkt sich auf den Zeitpunkt des `PUBLISHED` -Ereignisses aus. • Ein Fehler wurde behoben, bei dem das SDK über den ERROR-Callback Fehler beim Beitritt zur Kategorie auslöste, wenn die Verbindung zur Stage unterbrochen wurde, aber möglicherweise wiederhergestellt werden konnte (genauer gesagt, Fehler `FAILED` und `TIMEOUT` für die Kategorie `JOIN_ERROR`). • Ein Fehler im Vorgang `insertSeiMessage` wurde behoben, bei dem das Aktualisieren einer Strategie zu aufeinanderfolgenden Aufrufen von `insertSeiMessage` führen konnte und die SEI-Nachricht nicht gesendet wurde.

2. April 2025

Neues Kontingent: Zusammensetzungen pro Stage

Wir haben ein neues Kontingent für die maximale Anzahl gleichzeitiger Zusammensetzungen pro Stage hinzugefügt. Dies ist unter Service Quotas > Andere Quotas dokumentiert.

20. März 2025

Amazon IVS Broadcast SDK: Android 1.28.1, iOS 1.28.1 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.28.1	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.28.1/android/ • Fehlerbehebungen und Stabilitätsverbesserungen.
iOS-Broadcast-SDK 1.28.1	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.28.1/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.28.1/ios/ • Fehlerbehebungen und Stabilitätsverbesserungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,613 MB	13,760 MB

Architektur	Komprimierte Größe	Unkomprimierte Größe
armeabi-v7a	4,885 MB	9,558 MB
x86_64	5,728 MB	14,342 MB
86 x	5,987 MB	14,923 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,417 MB	7,698 MB

20. März 2025

IVS-Broadcast-SDK: Web 1.22.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.22.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • `null` wurde als gültiger Rückgabetyp für die Strategiemethode preferredLayerForStream hinzugefügt. • Es wurde ein Fehler behoben, bei dem `preferredLayerForStream` nicht erneut aufgerufen wurde, wenn nach dem Start des Streams neue Schichten verfügbar wurden. • Es wurde ein Fehler behoben, bei dem `stream.getHighestQualityLayer` nach dem Start des Streams nicht die Schicht mit der höchsten Qualität auswählte.

19. März 2025

Amazon IVS Broadcast SDK: Android 1.27.2, iOS 1.27.2 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.27.2	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.27.2/android/ • Es wurde eine durch Ressourcenverlust verursachte Regression behoben, die sich auf einige Geräte auswirkte, wenn 50 oder mehr Stages erstellt wurden. • Es wurde eine Regression behoben, die bei der Nutzung von Publishing-Software von Drittanbietern dazu führen konnte, dass Videos häufiger einfrieren.
iOS-Broadcast-SDK 1.27.2	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.27.2/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.27.2/ios/ • Es wurde eine Regression behoben, die bei der Nutzung von Publishing-Software von Drittanbietern dazu führen konnte, dass Videos häufiger einfrieren.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,700 MB	14,197 MB

Architektur	Komprimierte Größe	Unkomprimierte Größe
armeabi-v7a	4,945 MB	9,879 MB
x86_64	5,810 MB	14,802 MB
86 x	6,073 MB	15,412 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,622 MB	8,584 MB

13. März 2025

Dauer des Zielsegments

Diese Version erweitert die IVS-Echtzeit-Streaming-API, damit Sie die Zieldauer für aufgezeichnete Segmente definieren können, die bei Verwendung von zusammengesetzten Aufzeichnungen oder bei der Aufzeichnung eines Stageteilnehmers generiert werden. Spezifische API-Änderungen finden Sie im Dokumentverlauf (im Benutzerhandbuch und in den API-Referenztabellen).

6. März 2025

Zusammenfügen der Aufzeichnungen einzelner Teilnehmer

Dies ist die erste Version der neuen Funktionalität. Wenn Ihre Stage für die Aufzeichnung einzelner Teilnehmer konfiguriert ist, können Sie nun ein Zeitfenster angeben, in dem IVS versucht, im selben S3-Präfix wie bei der vorherigen Sitzung aufzuzeichnen, wenn ein Stage-Publisher die Verbindung zu einer Stage trennt und dann wieder herstellt. Mit anderen Worten: Wenn ein Publisher die Verbindung trennt und dann innerhalb des angegebenen Intervalls wieder herstellt, werden die einzelnen Aufzeichnungen als eine einzige Aufzeichnung betrachtet und zusammengeführt. Änderungen an der Dokumentation finden Sie im Dokumentverlauf (sowohl im Benutzerhandbuch als auch in den API-Referenztabellen).

03. März 2025

Amazon IVS Broadcast SDK: iOS 1.27.1 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
iOS-Broadcast-SDK 1.27.1	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.27.1/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.27.1/ios/ • Objekte, die bei Nutzung des Ultraweit winkelobjektivs an Pro-Geräten nahe an die Kamera gehalten werden, werden schärfer dargestellt.

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,625 MB	8,601 MB

20. Februar 2025

Amazon IVS Broadcast SDK: Android 1.27.0, iOS 1.27.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.27.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.27.0/android/

Plattform	Downloads und Änderungen
	• Fehlerbehebungen und Stabilitätsverbess erungen.
iOS-Broadcast-SDK 1.27.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.27.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.27.0/ios/ • Fehlerbehebungen und Stabilitätsverbess erungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,700 MB	14,197 MB
armeabi-v7a	4,944 MB	9,879 MB
x86_64	5,809 MB	14,802 MB
86 x	6,073 MB	15,412 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,625 MB	8,601 MB

20. Februar 2025

IVS-Broadcast-SDK: Web 1.21.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.21.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Die Strategietypen `preferredLayerForStream` wurden um `null` erweitert, was eine gültige Rückgabe darstellt. • TypeScript-Kompilierungsfehler, die auftreten, wenn TSConfig `skipLibCheck` auf „false“ gesetzt ist, wurden behoben. Hinweis: Im Rahmen dieser Version wurden die Typen in einem einzigen Rollup konsolidiert. Wenn eine Anwendung verschachtelte Typen auf Grundlage eines Pfads importiert, können Fehler auftreten. Treten Fehler auf, ändern Sie den Import einfach in `'amazon-ivs-broadcast'` .

30. Januar 2025

Amazon IVS Broadcast SDK: Android 1.26.0, iOS 1.26.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.26.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.26.0/android/

Plattform	Downloads und Änderungen
	• Fehlerbehebungen und Stabilitätsverbess erungen.
iOS-Broadcast-SDK 1.26.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.26.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.26.0/ios/ • Fehlerbehebungen und Stabilitätsverbess erungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,695 MB	14,186 MB
armeabi-v7a	4,939 MB	9,872 MB
x86_64	5,804 MB	14,790 MB
86 x	6,065 MB	15,398 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,624 MB	8,601 MB

23. Januar 2025

IVS-Broadcast-SDK: Web 1.20.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.20.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Die Methode `insertSeiMessage` auf LocalStageStream wurde hinzugefügt, um das Einfügen von SEI-Nutzdaten (Supplemental Enhancement Information, Ergänzende Informationen zur Verbesserung) in einen Veröffentlichungsvideostream zu ermöglichen. Siehe Abrufen von SEI-Daten (Supplemental Enhancement Information, Ergänzende Informationen zur Verbesserung) im Leitfaden zum IVS-Broadcast-SDK: Handbuch für Web.

12. Dezember 2024

Amazon IVS Broadcast SDK: Android 1.25.0, iOS 1.25.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.25.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.25.0/android/ • Ein Feature zur Simulcast-Steuerung wurde hinzugefügt. Siehe Konfigurieren mehrschichtiger Kodierung mit Simulcast (Subscriber) unter Streaming-Optimierungen.

Plattform	Downloads und Änderungen
	• Es wurden SEI-Nutzdaten (Supplemental Enhanced Information) für Subscriber mit einem neuen Feld für ImageDeviceFrame-Objekte verfügbar gemacht. Siehe Abrufen von SEI-Daten (Supplemental Enhancement Information) im Leitfaden zum IVS-Broadcast-SDK: Android. • Die Methode `SubscribeConfiguration::setInitialGain` wurde hinzugefügt, um die Konfiguration des anfänglichen Verstärkungswerts für eingehende Audiostreams zu ermöglichen. • Fehlerbehebungen und Stabilitätsverbesserungen.

Plattform	Downloads und Änderungen
iOS-Broadcast-SDK 1.25.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.25.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.25.0/ios/ • Ein Feature zur Simulcast-Steuerung wurde hinzugefügt. Siehe Konfigurieren mehrschichtiger Kodierung mit Simulcast (Subscriber) unter Streaming-Optimierungen. • Es wurden SEI-Nutzdaten (Supplemental Enhanced Information) für Subscriber mit einem neuen Feld für IVSImageDeviceFrame-Objekte verfügbar gemacht. Siehe Abrufen von SEI-Daten (Supplemental Enhancement Information) im Leitfaden zum IVS-Broadcast-SDK: iOS. • Die Methode `IVSSubscribeConfiguration.initialGain` wurde hinzugefügt, um die Konfiguration des anfänglichen Verstärkungswerts für eingehende Audiostreams zu ermöglichen. • Fehlerbehebungen und Stabilitätsverbesserungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,677 MB	14,103 MB
armeabi-v7a	4,905 MB	9,791 MB

Architektur	Komprimierte Größe	Unkomprimierte Größe
x86_64	5,786 MB	14,725 MB
86 x	6,030 MB	15,302 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,625 MB	8,585 MB

12. Dezember 2024

IVS-Broadcast-SDK: Web 1.19.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.19.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Ein Feature zur Simulcast-Steuerung wurde hinzugefügt. Siehe Konfigurieren mehrschichtiger Kodierung mit Simulcast (Subscriber) unter Streaming-Optimierungen. • Fehlerbehebungen und Stabilitätsverbesserungen.

10. Dezember 2024

Streaming-Thumbnail-Konfiguration in Echtzeit

In dieser Version können Sie die Aufzeichnung von Miniaturansichten für eine Live-Sitzung aktivieren/deaktivieren und das Intervall ändern, in dem Miniaturansichten für die Live-Sitzung generiert werden. Dies ist die erste Version dieser neuen Funktionalität. Siehe:

- Aufzeichnung einzelner Teilnehmer – Wir haben Beispiele und Informationen zu JSON-Metadaten aktualisiert und Preisinformationen sowie „Nur Thumbnail-Aufzeichnungen“ hinzugefügt.
- Zusammengesetzte Aufzeichnung – Wir haben Beispiele und Informationen zu JSON-Metadaten aktualisiert und Preisinformationen hinzugefügt.
- API-Referenz-RT – Wir haben mehrere Änderungen vorgenommen:
  - Das Objekt S3DestinationConfiguration wurde geändert: `thumbnailConfigurations` hinzugefügt. Dies wirkt sich auf die GetComposition-Antwort sowie die StartComposition-Anfrage und -Antwort aus.
  - Das Objekt AutoParticipantRecordingConfiguration wurde geändert: `thumbnailConfiguration` hinzugefügt und `NONE` als gültiger Wert für `mediaTypes` hinzugefügt. Dies betrifft die CreateStage-Anforderung und -Antwort, die GetStage-Antwort sowie die UpdateStage-Anforderung und -Antwort.
  - Zwei Objekte wurden hinzugefügt: CompositionThumbnailConfiguration und ParticipantThumbnailConfiguration.

13. November 2024

Amazon IVS Broadcast SDK: Android 1.24.0, iOS 1.24.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.24.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.24.0/android/ • Fehlerbehebungen und Stabilitätsverbesserungen.
iOS-Broadcast-SDK 1.24.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.24.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.24.0/ios/

Plattform	Downloads und Änderungen
	• Fehlerbehebungen und Stabilitätsverbess erungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,521 MB	13,791 MB
armeabi-v7a	4,789 MB	9,623 MB
x86_64	5,718 MB	14,709 MB
86 x	5,933 MB	15,163 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,589 MB	8,466 MB

12. November 2024

IVS-Broadcast-SDK: Web 1.18.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.18.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Es wurde ein neues Ereignis hinzugefügt, um SEI-Nutzdaten (Supplemental Enhanced Information) für Subscriber zur Verfügung zu stellen.

Plattform	Downloads und Änderungen
	• Es wurde eine Ausnahme behoben, die bei Anfragen zum Aufheben der Veröffentlichung und des Abonnements auftrat. • Eine Race-Bedingung wurde korrigiert, bei der schnelles Beitreten und Verlassen bei anderen Teilnehmern zu Fehlern führte.

10. Oktober 2024

IVS-Broadcast-SDK: Web 1.17.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.17.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Kleinere Fehlerbehebungen.

10. Oktober 2024

Amazon IVS Broadcast SDK: Android 1.23.0, iOS 1.23.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.23.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.23.0/android/ • Ab diesem Release veröffentlichen wir auch eine Version des Android-Broadcast-SDK, die Debug-Symbole enthält. Siehe Verwenden des SDK mit Debug-Symbolen. • Kleinere Fehlerbehebungen.

Plattform	Downloads und Änderungen
iOS-Broadcast-SDK 1.23.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.23.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.23.0/ios/ • Kleinere Fehlerbehebungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,432 MB	13,560 MB
armeabi-v7a	4,707 MB	9,451 MB
x86_64	5,626 MB	14,459 MB
86 x	5,838 MB	14,908 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,542 MB	8,316 MB

11. September 2024

Amazon IVS Broadcast SDK: Android 1.22.0, iOS 1.22.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.22.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.22.0/android/ • Es wurde ein Fehler behoben, bei dem bestimmte Android-Geräte nach dem Umschalten der Kameraeingabe ein schwarzes Bild in der Vorschau anzeigten. • Kleinere Fehlerbehebungen.
iOS-Broadcast-SDK 1.22.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.22.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.22.0/ios/ • Kleinere Fehlerbehebungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,359 MB	13,392 MB
armeabi-v7a	4,636 MB	9,325 MB
x86_64	5,548 MB	14,268 MB
86 x	5,754 MB	14,710 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,488 MB	8,199 MB

11. September 2024

IVS-Broadcast-SDK: Web 1.16.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.16.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Kleinere Fehlerbehebungen.

9. September 2024

RTMP-Erfassung

Als Alternative zur Verwendung des IVS-Broadcast-SDK können Sie jetzt Videos aus einer RTMP-Quelle auf einer IVS-Stage veröffentlichen (zusätzlich zu WHIP, das bereits unterstützt wurde). Änderungen an der Dokumentation finden Sie im Dokumentverlauf (sowohl im Benutzerhandbuch als auch in den API-Referenztabellen).

19. August 2024

Veröffentlichen/Abonnieren in der Konsole

Sie können jetzt über die IVS-Konsole veröffentlichen und abonnieren. Weitere Informationen finden Sie in Erste Schritte mit IVS-Echtzeit-Streaming unter Video veröffentlichen und abonnieren.

15. August 2024

IVS-Broadcast-SDK: Web 1.15.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.15.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Es wurde eine Race-Bedingung behoben, die sich auf die Qualität der Publisher-Medien auswirkt, wenn `join()` wiederholt aufgerufen wird. Der wiederholte Aufruf von `join()` löst das Ereignis `STAGE_PARTICIPANT_JOINED` zusammen mit den damit einhergehenden Änderungen des Veröffentlichungs- und Streamstatus nicht mehr aus. • Ein Fehler wurde behoben, der Probleme beim Parsen von Teilnehmer-Token verursachte, wenn im Feld `attributes` Nicht-Textzeichen verwendet wurden. • Es wurde eine Methode zum Konfigurieren der Subscriber eines Teilnehmers hinzugefügt. Zunächst können Sie nur die Mindestverzögerung des Jitter-Puffers konfigurieren. Weitere Informationen finden Sie in der SDK-Referenzdokumentation Konfiguration für das Abonnieren von Teilnehmern im Leitfaden zum Web-Broadcast-SDK und Ändern der Mindestverzögerung des Jitter-Puffers für Subscriber in Streaming-Optimierungen.

15. August 2024

Amazon IVS Broadcast SDK: Android 1.21.0, iOS 1.21.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.21.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.21.0/android/ • Ein Fehler wurde behoben, der sich auf Geräte mit MT6765-Chipsätzen auswirkte , bei denen die Subscriber-Vorschau unter bestimmten Umständen schwarze Bilder anzeigte. • Es wurde eine Methode zum Konfigurieren der Subscriber eines Teilnehmers hinzugefü gt. Zunächst können Sie nur die Mindestve rzögerung des Jitter-Puffers konfigurieren. Weitere Informationen finden Sie in der SDK-Referenzdokumentation Konfiguration für das Abonnieren von Teilnehmern im Leitfaden zum Android-Broadcast-SDK und Ändern der Mindestverzögerung des Jitter-Puffers für Subscriber in Streaming-Optimierungen. • Kleinere Fehlerbehebungen.
iOS-Broadcast-SDK 1.21.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.21.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.21.0/ios/ • Es wurde eine Methode zum Konfigurieren der Subscriber eines Teilnehmers hinzugefü gt. Zunächst können Sie nur die Mindestve

Plattform	Downloads und Änderungen
	rzögerung des Jitter-Puffers konfigurieren. Weitere Informationen finden Sie in der SDK-Referenzdokumentation Konfiguration für das Abonnieren von Teilnehmern im Leitfaden zum iOS-Broadcast-SDK und Ändern der Mindestverzögerung des Jitter-Puffers für Subscriber in Streaming-Optimierungen. • Kleinere Fehlerbehebungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,350 MB	13,378 MB
armeabi-v7a	4,628 MB	9,312 MB
x86_64	5,538 MB	14,253 MB
86 x	5,744 MB	14,694 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,485 MB	8,199 MB

18. Juli 2024

IVS-Broadcast-SDK: Web 1.14.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.14.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Verbesserungen an der API-Dokumentation • Ausreißer bei den Video- und Audiostat istiken, die bei Verbindungsneustarts gemeldet wurden, wurden behoben. • Kleinere Aktualisierungen der Abhängigk eiten.

18. Juli 2024

Amazon IVS Broadcast SDK: Android 1.20.0, iOS 1.20.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.20.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.20.0/android • Es wurde ein Fehler behoben, durch den das Broadcast-SDK nicht auf Chromebooks mit Intel-Prozessoren ausgeführt werden konnte. • Kleinere Fehlerbehebungen.
iOS-Broadcast-SDK 1.20.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.20.0/AmazonIVSBroadcast-Stages.xcframework.zip

Plattform	Downloads und Änderungen
	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.20.0/ios • Kleinere Fehlerbehebungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,318 MB	13,299 MB
armeabi-v7a	4,605 MB	9,254 MB
x86_64	5,507 MB	14,168 MB
86 x	5,715 MB	14,608 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,465 MB	8,164 MB

26. Juni 2024

Generieren von Teilnehmer-Token mit einem Schlüsselpaar

Sie können jetzt Teilnehmer-Token in Ihrer eigenen Serveranwendung mithilfe eines Schlüsselpaars generieren. So müssen Sie nicht jedes Mal CreateParticipantToken aufrufen, wenn Sie ein Teilnehmer-Token benötigen. Änderungen an der Dokumentation finden Sie im Dokumentverlauf (sowohl im Benutzerhandbuch als auch in den API-Referenztabellen).

20. Juni 2024

Aufzeichnung einzelner Teilnehmer

Die Aufzeichnung einzelner Teilnehmer ermöglicht es IVS-Echtzeit-Streaming-Kunden, IVS-Stage-Publisher einzeln in S3-Buckets aufzuzeichnen. Siehe Aufzeichnung, Aufzeichnung einzelner Teilnehmer und Änderungen in der Referenz zur API für IVS-Echtzeit-Streaming. (Bestimmte Änderungen an der Dokumentation finden Sie im Dokumentverlauf.)

13. Juni 2024

Amazon IVS Broadcast SDK: Android 1.19.0, iOS 1.19.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.19.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.19.0/android • Neuere Android-Versionen erfordern ein Symbol in der Benachrichtigung, die bei der Bildschirmerfassung angezeigt wird. Falls gewünscht, können Sie das Symbol jetzt anpassen, indem Sie `setSmallIcon` für den `Notification.Builder` aufrufen, der von `Session # createServiceNotificationBuilder` zurückgegeben wird. • Die Zeit für die Wiederherstellung der Verbindung bei Geräten, die von einer WLAN-Verbindung auf eine Mobilfunkverbindung umschalten, wurde verbessert. Diese Änderung erfordert die Berechtigung `CHANGE_NETWORK_STATE` .

Plattform	Downloads und Änderungen
iOS-Broadcast-SDK 1.19.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.19.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.19.0/ios • Kleinere Fehlerbehebungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,304 MB	13,340 MB
armeabi-v7a	4,598 MB	9,299 MB
x86_64	5,495 MB	14,207 MB
86 x	5,694 MB	14,625 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,393 MB	7,949 MB

13. Juni 2024

IVS-Broadcast-SDK: Web 1.13.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.13.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Die Dauer des Verhaltens bei Ereignisänderungen für `StageEvents.STAGE_PARTICIPANT_SUBSCRIBE_STATE_CHANGED` und `StageEvents.STAGE_PARTICIPANT_PUBLISH_STATE_CHANGED` wurde aktualisiert. Teilnehmer bleiben nun länger im Status `ATTEMPTING_SUBSCRIBE` oder `ATTEMPTING_PUBLISH`, bis das `ERRORED`-Ereignis ausgelöst wird. • Das `StageEvents.ERROR`-Ereignis wurde hinzugefügt, um Fehler zu überwachen, die im SDK auftreten. Weitere Informationen finden Sie unter Fehlerbehandlung im Web-Leitfaden des Echtzeit-Broadcast-SDK.

20. Mai 2024

IVS-Broadcast-SDK: Web 1.12.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.12.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference

Plattform	Downloads und Änderungen
	• Die Behandlung von Wiederholungsversu chen bei Veröffentlichungs- und Abonnemen tvorgängen wurde verbessert. • Verbesserte Analytik, insbesondere Messung der Latenz und der Audioqualität.

16. Mai 2024

Amazon IVS Broadcast SDK: Android 1.18.0, iOS 1.18.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.18.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.18.0/android • Das SDK sendet jetzt spezifische Fehlercod es, wenn eine verbundene Stage von der AWS-Steuerebene gelöscht oder wenn das verwendete Token widerrufen wird. • Kleinere Fehlerbehebungen.
iOS-Broadcast-SDK 1.18.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.18.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.18.0/ios • Das SDK sendet jetzt spezifische Fehlercod es, wenn eine verbundene Stage von der AWS-Steuerebene gelöscht oder wenn das verwendete Token widerrufen wird.

Plattform	Downloads und Änderungen
	• Die IVSCamera-Methode `setVideoZoomFactor` und die zugehörigen `IVSCameraDelegate` -Methoden wurden hinzugefügt.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,275 MB	13,279 MB
armeabi-v7a	4,573 MB	9,254 MB
x86_64	5,472 MB	14,142 MB
86 x	5,664 MB	14,554 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,393 MB	7,916 MB

6. Mai 2024

IVS-Broadcast-SDK: Web 1.11.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.11.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Es wurde ein Grenzfall behoben, bei dem das SDK bei einem Stage-`DISCONNECT`

Plattform	Downloads und Änderungen
	nicht versuchte, eine Wiederherstellung durchzuführen. • Die Fehlermeldung für einen `join()`-Timeout-Fehler wurde aktualisiert. Statt „InitialConnectTimedOut nach 10 Sekunden“ gibt das SDK jetzt „Zeitüberschreitung beim Vorgang“ zurück.

30. April 2024

IVS-Broadcast-SDK: Web 1.10.1 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.10.1	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Kleinere Fehlerbehebungen.

30. April 2024

Amazon IVS Broadcast SDK: Android 1.15.2, iOS 1.15.2 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.15.2	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.2/android • Kleinere Fehlerbehebungen. Führen Sie nur dann ein Upgrade auf diese Version durch, wenn Sie einen bestimmten Grund

Plattform	Downloads und Änderungen
	dafür haben. Verwenden Sie andernfalls die höchste veröffentlichte Version.
iOS-Broadcast-SDK 1.15.2	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.15.2/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.2/ios • Kleinere Fehlerbehebungen. Führen Sie nur dann ein Upgrade auf diese Version durch, wenn Sie einen bestimmten Grund dafür haben. Verwenden Sie andernfalls die höchste veröffentlichte Version.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,244 MB	13,198 MB
armeabi-v7a	4,543 MB	9,192 MB
x86_64	5,437 MB	14,051 MB
86 x	5,631 MB	14,461 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,359 MB	7,836 MB

22. April 2024

Amazon IVS Broadcast SDK: Android 1.17.0, iOS 1.17.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.17.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.17.0/android • Ein seltener Absturz, der beim Veröffent lichen auftreten kann, wurde behoben.
iOS-Broadcast-SDK 1.17.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.17.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.17.0/ios • Das `AmazonIVSBroadcast` -Framewor k enthält jetzt ein von Apple gefordertes Datenschutzmanifest.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,273 MB	13,275 MB
armeabi-v7a	4,571 MB	9,251 MB
x86_64	5,468 MB	14,137 MB
86 x	5,662 MB	14,549 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,388 MB	7,916 MB

21. März 2024

Amazon IVS Broadcast SDK: Android 1.16.0, iOS 1.16.0, Web 1.10.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.10.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Ein sporadisch auftretender Fehler beim Bereinigen von Verbindungen nach dem Abbestellen oder Verlassen einer Stage wurde behoben.
Android-Broadcast-SDK 1.16.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.16.0/android • Das Einfrieren der Vorschau auf der Exynos-Variante von Samsung-Geräten mit Android 14 wurde behoben. • Eine Funktion zum Abfragen der Zoomfähig keiten der Kamera und zum Einstellen des Zoomfaktors wurde hinzugefügt.
iOS-Broadcast-SDK 1.16.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.16.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.16.0/ios

Plattform	Downloads und Änderungen
	• Kleinere Fehlerbehebungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,253 MB	13,21 MB
armeabi-v7a	4,551 MB	9,204 MB
x86_64	5,447 MB	14,070 MB
86 x	5,640 MB	14,480 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,361 MB	7,836 MB

13. März 2024

Amazon IVS Broadcast SDK: Android 1.15.1, iOS 1.15.1 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.15.1	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.1/android • Ein seltener Absturz beim Abonnieren eines Remote-Teilnehmers wurde behoben.

Plattform	Downloads und Änderungen
iOS-Broadcast-SDK 1.15.1	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.15.1/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.1/ios • Ein seltener Absturz beim Abonnieren eines Remote-Teilnehmers wurde behoben.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,243 MB	13,194 MB
armeabi-v7a	4,541 MB	9,188 MB
x86_64	5,628 MB	14,455 MB
86 x	5,434 MB	14,046 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,358 MB	7,820 MB

13. März 2024

Aktualisierungen der API für serverseitige Zusammensetzung

Wir haben der GridConfiguration neue Eigenschaften und ein neues Bild-in-Bild-Layout hinzugefügt, wodurch die Anpassungsoptionen für Zusammensetzungen verbessert wurden. Spezifische

Änderungen an der Dokumentation finden Sie im Dokumentverlauf (siehe Tabelle der API-Referenzänderungen).

Wichtig: Stellen Sie sicher, dass Ihre Anwendung nicht von den spezifischen Features des aktuellen Layouts abhängt, z. B. von Größe und Position der Kacheln. Visuelle Verbesserungen an Layouts können jederzeit vorgenommen werden.

8. März 2024

Aktualisierungen am Layout der serverseitigen Zusammensetzung

Heute haben wir die Änderungen am Standard-Rasterlayout aktiviert, die im Eintrag vom 7. Februar 2024 beschrieben sind.

22. Februar 2024

Amazon IVS Broadcast SDK: Android 1.15.0, iOS 1.15.0, Web 1.9.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.9.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Interne Fehlerbehandlung wurde verbessert.
Android-Broadcast-SDK 1.15.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.0/android • Kleinere Fehlerbehebungen.
iOS-Broadcast-SDK 1.15.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.15.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.15.0/ios

Plattform	Downloads und Änderungen
	• Eine `AVPictureInPictureController` -Erweiterung wurde hinzugefügt, um das Erstellen einer neuen Instance mit einer `IVSImagePreviewView` zu ermöglichen. • Es wurde eine neue API für `IVSImageDevice` hinzugefügt, um einen `AVSampleBufferDisplayLayer` zu erstellen, auf den das Gerät rendert. • Ein Problem mit niedriger Bitrate auf Geräten mit iOS 17 und höher wurde behoben. • Kleinere Fehlerbehebungen.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,243 MB	13,194 MB
armeabi-v7a	4,541 MB	9,188 MB
x86_64	5,628 MB	14,455 MB
86 x	5,434 MB	14,046 MB

Broadcast-SDK-Größe: iOS

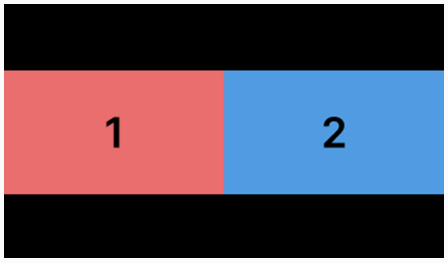
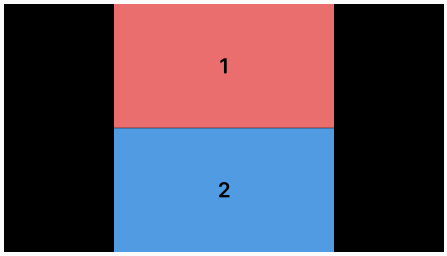
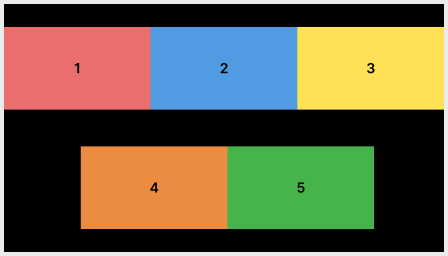
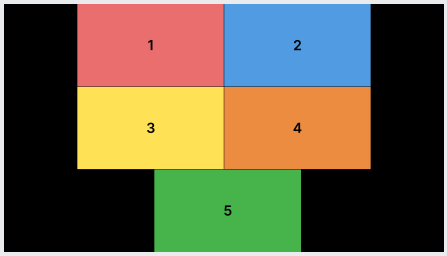
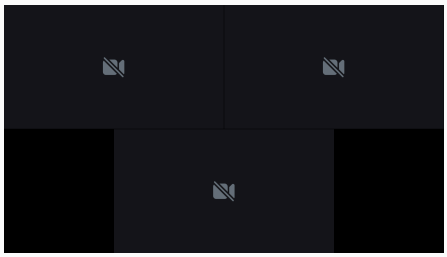
Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,358 MB	7,820 MB

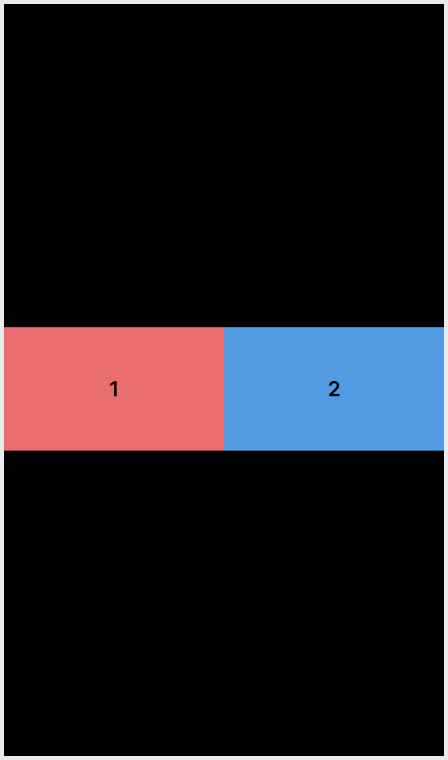
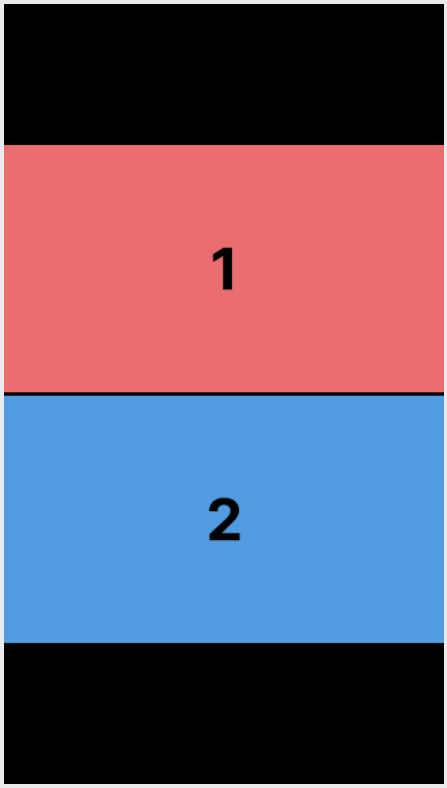
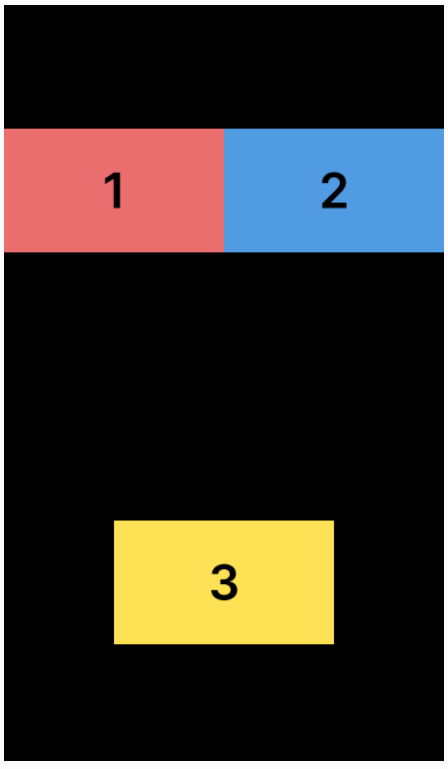
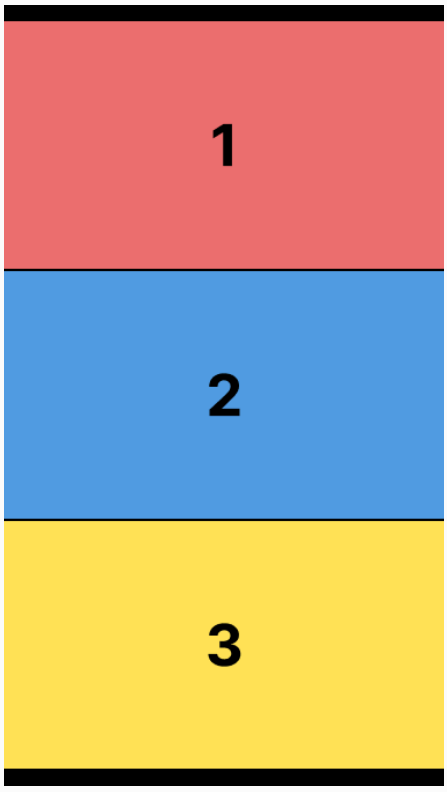
7. Februar 2024

Aktualisierungen am Layout der serverseitigen Zusammensetzung

In diesem Release wurden visuelle Verbesserungen am Standard-Rasterlayout eingeführt. Diese Änderungen werden die Anzeige von Videos optimieren und leere Flächen reduzieren. Diese Änderungen werden am 7. März 2024 aktiviert.

Wichtig: Stellen Sie sicher, dass Ihre Anwendung nicht von den spezifischen Features des aktuellen Layouts abhängt, z. B. von Größe und Position der Kacheln. Visuelle Verbesserungen an Layouts können jederzeit vorgenommen werden.

Beschreibung der Änderung	Alt	Neu
Wählt automatisch die optimale Platzierung der Teilnehmer aus, um die Videogröße zu maximieren.		
Verbessert die Umgebungs nutzung, indem Lücken reduziert und schwarze Balken minimiert werden.		
Fügt einen neuen „Kamera aus“-Indikator hinzu, damit Teilnehmer, die kein Video teilen, deutlich erkennbar sind.		

Beschreibung der Änderung	Alt	Neu
Verbessert die Umgebungs nutzung und die Proportio nen für Anwendungsfälle im Hochformat.		
Verbessert die Umgebungs nutzung für Anwendung sfälle im Hochformat, indem die Abstände zwischen den Teilnehmern minimiert und Letterboxing oder Pillarboxing reduziert werden.		

6. Februar 2024

OBS- und WHIP-Unterstützung

IVS kann mit WHIP-kompatiblen Encodern wie OBS verwendet werden, um in Echtzeit in IVS-Streaming zu veröffentlichen. WHIP (WebRTC-HTTP Ingestion Protocol) ist ein IETF-Entwurf, der zur Standardisierung der WebRTC-Erfassung entwickelt wurde. Siehe neue Seite zu OBS- und WHIP-Unterstützung.

1. Februar 2024

Amazon IVS Broadcast SDK: Android 1.14.1, iOS 1.14.1, Web 1.8.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.8.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Die Codierung von Layern mit Simulcast ist jetzt standardmäßig deaktiviert. • Ein Problem wurde behoben, bei dem eine Stage-Instance nicht sauber getrennt wurde, wenn eine Stage gelöscht wurde oder wenn ein Teilnehmer vom Server getrennt wurde. Das SDK gibt jetzt ein `STAGE_CONNECTION_STATE_CHANGED` -Ereignis mit dem Status `DISCONNECTED` (statt `ERRORED` und dann `CONNECTING`) aus. • Problem behoben, bei dem die Veröffentlichung fehlschlug, wenn die Strategie mit leeren Audio- oder Videospuren aktualisiert wurde.
Android-Broadcast-SDK 1.14.1	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.14.1/android

Plattform	Downloads und Änderungen
	• Die Codierung von Layern mit Simulcast ist jetzt standardmäßig deaktiviert. • `libWebRTC` wurde von M108 auf M119 aktualisiert. • Es wurden mehrere Abstürze behoben, um die allgemeine Stabilität zu verbessern. • Unterstützung für Stereo-Veröffentlichung wurde hinzugefügt. Diese kann über das `StageAudioConfiguration` -Objekt aktiviert werden. • Es wurde ein Fehler behoben, der dazu führte, dass Teilnehmer nach dem Beitritt zu einer Sitzung einen schwarzen Feed erhielten. • Interne `libWebRTC` -Verweise wurden aktualisiert, um Symbolkonflikte zu vermeiden, wenn andere `libWebRTC` -Versionen in derselben Hostanwendung enthalten sind.

Plattform	Downloads und Änderungen
iOS-Broadcast-SDK 1.14.1	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.14.1/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.14.1/ios • Die Codierung von Layern mit Simulcast ist jetzt standardmäßig deaktiviert. • `libWebRTC` wurde von M108 auf M119 aktualisiert. • Es wurden mehrere Abstürze behoben, um die allgemeine Stabilität zu verbessern. • Unterstützung für Stereo-Veröffentlichung wurde hinzugefügt. Diese kann über `IVSLocalStageStreamAudioConfiguration` aktiviert werden. • Ein Absturz beim Aktivieren des Nur-Audio-Modus für andere Teilnehmer wurde behoben. • TTV wurde verbessert und die Binärgröße wurde reduziert.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,223 MB	13,118 MB
armeabi-v7a	4,524 MB	9,134 MB
x86_64	5,418 MB	13,955 MB
86 x	5,61 MB	14,369 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,350 MB	7,790 MB

3. Januar 2024

Amazon IVS Broadcast SDK: Android 1.13.4, iOS 1.13.4, Web 1.7.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.7.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Die Zeit bis zum Video wurde für Abonnenten verbessert, die Stages beitreten. • Die `minAudioBitrateKbps` -Eigenschaft wurde entfernt (sie wurde nicht verwendet). • Die Netzwerkwiederherstellung bei Interneta usfällen oder -änderungen wurde verbessert.
Android-Broadcast-SDK 1.13.4	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.4/android • StageAudioConfiguration unterstützt jetzt die Einstellung, ob Echounterdrückung aktiviert werden soll.
iOS-Broadcast-SDK 1.13.4	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.13.4/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.4/ios

Plattform	Downloads und Änderungen
	• Unter iOS haben wir die Audio-Engine sowohl für die Aufzeichnung als auch für die Wiedergabe verbessert, wobei der Schwerpunkt auf Stabilität und Wiederher stellbarkeit lag. Dies verbessert die Unterstüt zung bei Routenänderungen während der Nutzung, verbessert die Akkuerholung in Grenzfällen und reduziert die Blockierung von Haupt-Threads. • Es wurde ein Problem behoben, bei dem das Mikrofon möglicherweise auch dann aktiv blieb, wenn es von einer Stage getrennt wurde, sodass die iOS-Datenschutzanzeige eingeschaltet blieb. (Das SDK verarbeitete zu diesem Zeitpunkt kein eingehendes Audio.)

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,187 MB	13,025 MB
armeabi-v7a	4,491 MB	9,056 MB
x86_64	5,359 MB	13,829 MB
86 x	5,553 MB	14,214 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,45 MB	7,84 MB

07. Dezember 2023

Neue CloudWatch-Metriken

Wir haben die Metrik PacketLoss (Stage) in DownloadPacketLoss (Stage) umbenannt. Wir haben außerdem zusätzliche CloudWatch-Metriken für IVS-Echtzeit-Streaming veröffentlicht:

- DownloadPacketLoss (Stage, Teilnehmer)
- DroppedFrames (Stage, Teilnehmer)
- SubscribeBitrate (Stage, Teilnehmer, Medientyp)

Weitere Informationen finden Sie unter Überwachen von IVS-Echtzeit-Streaming.

4. Dezember 2023

Amazon IVS Broadcast SDK: Android 1.13.2 und iOS 1.13.2 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Alle Mobilgeräte (Android und iOS)	• Die Konfiguration zur Geräuschunterdrück ung steht Entwicklern zum Aktivieren/Deaktiv ieren für die Veröffentlichung zur Verfügung.
Android Broadcast SDK 1.13.2	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.2/android • Die Ladezeit des Videos (TTV) beim Beitritt zur ersten Stage einer Sitzung wurde verbessert.
iOS-Broadcast-SDK 1.13.2	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.13.2/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.2/ios

Plattform	Downloads und Änderungen
	• Keine Änderungen am Echtzeit-SDK.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,177 MB	13,01 MB
armeabi-v7a	4,485 MB	9,045 MB
x86_64	5,352 MB	13,808 MB
86 x	5,547 MB	14,192 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,45 MB	7,82 MB

21. November 2023

Amazon IVS Broadcast SDK: Android 1.13.1 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android Broadcast SDK 1.13.1	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.1/android • Es wurde ein Problem behoben, das beim schnellen Verlassen, Loslassen und erneuten Betreten derselben Stage zu einem Absturz führte.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,177 MB	13,102 MB
armeabi-v7a	4,485 MB	9,046 MB
x86_64	5,353 MB	13,809 MB
86 x	5,547 MB	14,192 MB

17. November 2023

Amazon IVS Broadcast SDK: Web 1.13.0 und iOS 1.13.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Alle Mobilgeräte (Android und iOS)	• Aktualisierte Streaming-Optimierungen. Unter anderem erfordert das Feature „Adaptives Streaming: Ebenen-Codierung mit Simulcast “ jetzt eine ausdrückliche Zustimmung und wird nur in neueren Versionen des SDK unterstützt. • Die Stabilität der Stages wurde verbesser t, indem das Auftreten seltener Abstürze reduziert wurde. • Die Ladezeit des Videos (TTV) beim Beitritt zu einer Stage wurde verbessert. • Die Erfahrung mit Bluetooth-Geräten wurde verbessert. • Die CPU- und Speichernutzung des SDK wurde optimiert und die Bibliotheksgröße reduziert.

Plattform	Downloads und Änderungen
	• Die `StageAudioManager` -Klasse wurde hinzugefügt, mit der sich Audioaufnahme- und Wiedergabeparameter festlegen lassen, einschließlich Voreinstellungen für Sprachkommunikation, Medienwiedergabe und mehr. Einzelheiten finden Sie auf der neuen Seite IVS Broadcast SDK: Mobile Audiomodi. • Es wurde eine neue `requestQualityStats` -Funktion hinzugefügt, um strukturierte Qualitätsereignisse aus WebRTC-Statistiken anzuzeigen. • Es wurde eine neue Funktion zur Aktualisierung der Audiobitrate hinzugefügt. Es wird wie die Videokonfiguration auf `LocalStageStream` -Objekte festgelegt, jedoch über ein neues Audiokonfigurationsobjekt.

Plattform	Downloads und Änderungen

Android Broadcast SDK 1.13.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.0/android
	• Alle Methoden in der `StageRenderer` -Schnittstelle sind jetzt optional.
	• Unterstützung für die `Surfaceview` -basierte Vorschau hinzugefügt, um eine bessere Leistung zu erzielen. Die vorhanden en `getPreview` -Methoden in `Session` und `StageStream` geben weiterhin eine Unterklasse von `TextureView` zurück, dies kann sich jedoch in einer zukünftigen SDK-Version ändern.
	  • Wenn Ihre Anwendung speziell von `TextureView` abhängt, können Sie ohne Änderungen fortfahren. Sie können auch von `getPreview` zu `getPrevie wTextureView` wechseln, um sich auf die eventuelle Änderung der Rückgabew erte des Standardwerts `getPreview` vorzubereiten.
	  • Wenn Ihre Anwendung `TextureView` nicht speziell erfordert, empfehlen wir für eine geringere CPU- und Speichera uslastung den Wechsel zu `getPrevie wSurfaceView` .
	• Das SDK implementiert jetzt einen neuen Vorschautyp namens `ImagePrev iewSurfaceTarget` , der mit dem von der Anwendung bereitgestellten Android-S urface-Objekt funktioniert. Es handelt sich nicht um eine Unterklasse von Android View, die eine bessere Flexibilität bietet.

Plattform	Downloads und Änderungen
	• Der Fall, bei dem der `onFrame`-Rückruf für entfernte Teilnehmer zur falschen Zeit mit der falschen Größe aufgerufen wurde, wurde behoben. • `SurfaceSource # getInputSurface` ist jetzt mit `@Nullable` annotiert. Ihr Code sollte es vor der Verwendung überprüfen. • `UserId` und `attributes` wurden zu `ParticipantInfo` hinzugefügt. Die Eigenschaften `UserId` und `attributes` sind in das Token eingebettet und können von Anwendungen über `ParticipantInfo` abgerufen werden, wenn ein Teilnehmer beitritt. • Die Kameraaufnahme und das Rendern der Vorschau sind jetzt standardmäßig auf 720 x 1280 oder die Veröffentlichungsauflösung (je nachdem, welcher Wert höher ist) bei 15 FPS eingestellt. Mit können Sie die Auflösung und/oder die Bildfrequenz anpassen `StageVideoConfiguration # setCameraCaptureQuality` . • `IllegalArgumentException` , das beim Festlegen von Konfigurationseigenschaften ausgelöst wird, enthält nun den bereitgestellten Wert in der Ausnahmemeldung.

Plattform	Downloads und Änderungen
iOS Broadcast SDK 1.13.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.13.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.13.0/ios • Es wurde ein Problem behoben, bei dem das SDK die Videokonfiguration nicht ändert, wenn die Videokonfiguration vor der Veröffentlichung aktualisiert wird. • Die Google-Korrektur für eine LibVPX-Schwachstelle (CVE-2023-5217) wurde integriert. (Beachten Sie, dass das Android-SDK für dieses Problem keine Änderungen erfordert.) • Bei Anwendungen, die andere Bibliothe ken verwenden, die `libWebRTC` enthalten , treten keine Konflikte mehr mit dem IVS Broadcast SDK auf. • Alle Methoden im `IVSStageRenderer` - Protokoll sind jetzt als `@optional` markiert. • Von unseren SDKs zurückgegebene Mikrofone und Kameras haben jetzt eine garantierte Sortierreihenfolge, wie in den SDKs selbst dokumentiert. • Mehrere Kameras können jetzt den Wert `true` für ihre `isDefault` -Eigenschaft haben, einen für jede vom Betriebssystem festgelegte Position. • `IVSStageAudioManager` hinzugefügt, das eine präzise Kontrolle über das zugrunde liegende `AVAudioSession` ermöglicht, um

Plattform	Downloads und Änderungen
	eine größere Vielfalt an Anwendungsfällen für die Stages-Funktionalität zu ermöglichen. • Hinzufügung von `UserId` zu `ParticipantInfo` .

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,17 MB	13,00 MB
armeabi-v7a	4,48 MB	9,04 MB
x86_64	5,35 MB	13,80 MB
86 x	5,54 MB	14,18 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	3,45 MB	7,84 MB

16. November 2023

Zusammengesetzte Aufzeichnung

Dieses neue Feature ermöglicht die Aufzeichnung der zusammengesetzten Ansicht einer IVS-Stage in einem Amazon-S3-Bucket. Weitere Informationen finden Sie unter:

- Zusammengesetzte Aufnahme – Dies ist eine neue Seite.
- Erste Schritte mit IVS-Echtzeit-Streaming – Es wurden S3-Endpunkte zur Richtlinie in „IAM-Berechtigungen einrichten“ hinzugefügt.

- Service Quotas – Es wurden Aufruf-Kontingente für die neuen Endpunkte hinzugefügt.
- API-Referenz zu IVS-Echtzeit-Streaming – Es wurden 4 StorageConfiguration-Endpunkte und 7 Objekte hinzugefügt (DestinationDetail, RecordingConfiguration, S3DestinationConfiguration, S3Detail, S3StorageConfiguration, StorageConfiguration, StorageConfigurationSummary). Es wurden außerdem 3 Objekte geändert (Composition, Destination, DestinationConfiguration). Dies wirkt sich auf die GetComposition-Antwort und die StartComposition-Anfrage und -Antwort aus.

16. November 2023

Serverseitige Zusammensetzung

Mit der serverseitigen IVS-Zusammensetzung können Clients die Zusammensetzung und Übertragung einer IVS-Stage an einen von IVS verwalteten Service verlagern. Die serverseitige Zusammensetzung und RTMP-Übertragung an einen Kanal werden über Endpunkte der IVS-Steuerebene in der Heimatregion der Stage aufgerufen. Weitere Informationen finden Sie unter:

- Erste Schritte mit IVS-Echtzeit-Streaming – Es wurden SSC-Endpunkte zur Richtlinie im Abschnitt „IAM-Berechtigungen einrichten“ hinzugefügt.
- Verwendung von Amazon EventBridge mit IVS-Echtzeit-Streaming – Es wurden neue Metriken hinzugefügt.
- Serverseitige Zusammensetzung – Dieses neue Dokument enthält eine Übersicht und Anweisungen zur Einrichtung.
- Service Quotas (Echtzeit-Streaming) – Es wurden neue Anrufratenlimits und andere Kontingente hinzugefügt.
- Referenz zur Echtzeit-Streaming-API – Es wurden 8 Zusammensetzungs- und EncoderConfiguration-Endpunkte sowie 11 Objekte hinzugefügt (ChannelDestinationConfiguration, Composition, CompositionSummary, Destination, DestinationConfiguration, DestinationSummary, EncoderConfiguration, EncoderConfigurationSummary, GridConfiguration, LayoutConfiguration und Video).

Im IVS-Streaming-Benutzerhandbuch mit niedriger Latenz finden Sie Folgendes:

- Aktivierung mehrerer Hosts in einem IVS-Stream – Es wurde „Broadcasting einer Stage: Clientseitige im Vergleich zu serverseitige Zusammensetzung“ hinzugefügt und „4. Übertragung der Stage“ wurde aktualisiert.

16. Oktober 2023

Amazon IVS Broadcast SDK: Web 1.6.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.6.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Verbesserte Time-To-Video (TTV). • `maxAudioBitrate` -Konfiguration hinzugefügt, die bis zu 128 kbit/s an Mono- oder Stereo-Audiokanälen unterstützt.

12. Oktober 2023

Neue CloudWatch-Metriken und Teilnehmerdaten

Wir haben CloudWatch-Metriken für IVS-Echtzeit-Streaming veröffentlicht. Weitere Informationen finden Sie unter Überwachen von IVS-Echtzeit-Streaming.

Dem Teilnehmer-API-Objekt wurden auch sechs Felder hinzugefügt: `browserName`, `browserVersion`, `ispName`, `osName`, `osVersion` und `sdkVersion`. Dies wirkt sich auf die GetParticipant-Antwort aus. Siehe die API-Referenz zu IVS-Echtzeit-Streaming.

12. Oktober 2023

Amazon IVS Broadcast SDK: Android 1.12.1 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Android-Broadcast-SDK 1.12.1	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.12.1/android

Plattform	Downloads und Änderungen
	• Es wurde ein Fehler behoben, bei dem das Aufrufen von `BroadcastSession.setListener` zu einem Fehler führte.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,853 MB	16,375 MB
armeabi-v7a	4,895 MB	10,803 MB
x86_64	6,149 MB	17,318 MB
86 x	6,328 MB	17,186 MB

14. September 2023

Amazon IVS-Broadcast-SDK: Web 1.5.2 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.5.2	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Es wurde ein Fehler behoben, der das erneute Veröffentlichen mit `refreshStrategy` verhinderte, wenn der Status „Veröffentlicht“ in einen `ERRORED`-Status übergeht.

23. August 2023

Amazon IVS Broadcast SDK: Web 1.5.1, Android 1.12.0 und iOS 1.12.0 (Echtzeit-Streaming)

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.5.1	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Ein Fehler mit internen Maybe-Typen in TypeScript 5 wurde behoben. • Bessere Erkennung für Simulcast-Unterstützung hinzugefügt. • Zwei Race-Bedingungen mit `refreshStrategy` beim Versuch zu veröffentlichen wurden behoben. • Eine Race-Bedingung mit `refreshStrategy` beim Versuch, Teilnehmer zu aktualisieren, um sie zu abonnieren, wurde behoben.
Alle Mobilgeräte (Android und iOS)	• Es wurde ein seltenes Problem behoben, bei dem die Veröffentlichungsaktion nie abgeschlossen wurde. • Die Stabilität der Stages wurde verbessert, indem das Auftreten seltener Abstürze reduziert wurde. • Die Stabilität der Stages wurde verbessert, indem Probleme mit Race-Bedingungen behoben wurden, die aufgrund des schnellen Beitritts oder Austritts entstanden sind. • Eine neue `setOnFrameCallback` -Methode wurde für `ImageDevice` hinzugefügt. Auf diese Weise kann

Plattform	Downloads und Änderungen
	beobachtet werden, wie die Frames das Gerät selbst durchlaufen, was einen Einblick in das Seitenverhältnis der aktuellen Bilder ermöglicht. Diese Methode kann auch verwendet werden, um zu erkennen, wann der erste Frame für einen Remote-Te ilnehmer in einer Stage gerendert wird.
Android-Broadcast-SDK 1.12.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.12.0/android • Android 9 wird jetzt unterstützt. • Verbesserte CPU-Auslastung und -Leistung.
iOS-Broadcast-SDK 1.12.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.12.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.12.0/ios • Die Signatur von `IVSDeviceDiscovery.createAudioSourceWithName` wurde korrigiert und gibt jetzt `IVSCustomAudioSource` statt `IVSCustomImageSource` zurück.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,853 MB	16,375 MB
armeabi-v7a	4,895 MB	10,803 MB
x86_64	6,149 MB	17,318 MB

Architektur	Komprimierte Größe	Unkomprimierte Größe
86 x	6,328 MB	17,186 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	5,06 MB	10,92 MB

7. August 2023

Amazon IVS Broadcast SDK: Web 1.5.0, Android 1.11.0, und iOS 1.11.0

Plattform	Downloads und Änderungen
Web-Broadcast-SDK 1.5.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-web-broadcast/docs/sdk-reference • Simulcast hinzugefügt – Wenn aktiviert, ermöglicht dieses Feature dem Publisher, Videoebenen mit hoher und niedriger Qualität zu senden. Subscriber wählen automatisch ihre optimale Qualität auf der Grundlage ihrer Netzwerkbedingungen aus. Siehe Optimieren von Medien.
Alle Mobilgeräte (Android und iOS)	Simulcast hinzugefügt – Wenn aktiviert, ermöglicht dieses Feature dem Publisher, Videoebenen mit hoher und niedriger Qualität zu senden. Subscriber wählen automatisch ihre optimale Qualität auf der Grundlage ihrer Netzwerkbedingungen aus. Siehe „Layered Encoding mit Simulcast aktivieren/deaktivieren“

Plattform	Downloads und Änderungen
	in den Anleitungen zum Android- und iOS-Broadcast-SDK.
Android-Broadcast-SDK 1.11.0	Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.11.0/android • Es wurde ein Problem behoben, bei dem das Erstellen vieler Stages letztendlich zu einem Absturz führte. (Die genaue Anzahl der Stages hängt vom Gerät ab.)
iOS-Broadcast-SDK 1.11.0	Für Echtzeit-Streaming herunterladen: https://broadcast.live-video.net/1.11.0/AmazonIVSBroadcast-Stages.xcframework.zip Referenzdokumentation: https://aws.github.io/amazon-ivs-broadcast-docs/1.11.0/ios • Korrigierte die Signatur von `IVSDeviceDiscovery.createAudioSourceWithName` zur Rückgabe von `IVSCustomAudioSource` statt `IVSCustomImageSource`.

Broadcast-SDK-Größe: Android

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64-v8a	5,811 MB	16,186 MB
armeabi-v7a	4,857 MB	10,646 MB
x86_64	6,108 MB	17,122 MB
86 x	6,289 MB	16,994 MB

Broadcast-SDK-Größe: iOS

Architektur	Komprimierte Größe	Unkomprimierte Größe
arm64	5,030 MB	10,810 MB

7. August 2023

Streaming in Echtzeit

Mit Amazon Interactive Video Service (IVS)-Echtzeit-Streaming können Sie Live-Streams mit einer Latenz bereitstellen, bei denen, vom Host bis zum Zuschauer, die Latenz unter 300 Millisekunden liegen kann.

Diese Version enthält wichtige Änderungen an der Dokumentation. Die Landingpage zur IVS-Dokumentation hat jetzt separate Abschnitte für Echtzeit-Streaming und Streaming mit niedriger Latenz. Jeder Abschnitt hat sein eigenes Benutzerhandbuch und eine eigene API-Referenz. Einzelheiten zur Dokumentation finden Sie in der Dokumentenhistorie (für Änderungen an der Dokumentation sowohl für Echtzeit und niedriger Latenz). Für Echtzeit-Streaming beginnen Sie mit dem Benutzerhandbuch für IVS-Echtzeit-Streaming und der Referenz zur IVS-Echtzeit-Streaming-API.