CHAPTER 1

Cisco StadiumVision Mobile Introduction

First Published: May 26, 2015

Revised: June 12, 2015

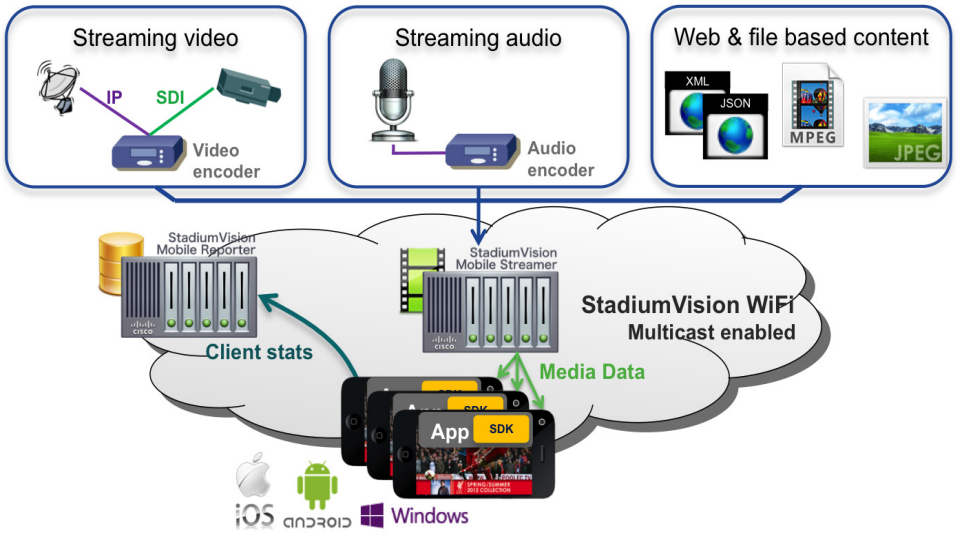
This chapter provides an overview of the Cisco StadiumVision Mobile solution and contains the following sections:

Cisco StadiumVision Mobile Solution Overview

The Cisco StadiumVision Mobile (SVM) solution enables the reliable delivery of low-delay video and data streams to fans' Wi-Fi devices at venues. Figure 1-1 illustrates a high-level view of the Cisco StadiumVision Mobile solution, which has the following attributes:

- Consists of Video Encoder, Streamer and Reporter products
- Requires integration of Cisco Client SDK in the mobile application
- Builds upon Cisco Connected Stadium and Cisco Connected Stadium Wi-Fi solutions

Figure 1-1 Cisco StadiumVision Mobile Architecture



Key Terms and Concepts

The following are key terms and concepts as they apply to the Cisco StadiumVision Mobile solution.

Cisco Demo or Sample App: A standalone mobile application available to a Stadium Operator for testing and evaluating the Cisco StadiumVision Mobile solution.

Repair: In the context of Cisco StadiumVision Mobile, an application-layer mechanism that allows Cisco StadiumVision Mobile Clients to recover lost packets.

Stadium Operator: The entity hosting and configuring the Cisco StadiumVision Mobile solution.

SVM: Cisco StadiumVision Mobile

SVM Reporter: A standalone application used to collect Cisco StadiumVision Mobile Client statistics.

SVM Session: The protocol and associated parameters which define the sender and receiver configuration for the streaming of content.

SVM Session Announcement/Discovery: Methods used by the Cisco StadiumVision Mobile Streamer and SVM Client to allow a mobile device to obtain the list of available sessions and associated session metadata.

SVM Session Triplet Key: A specific combination of **Venue**, **Content Owner**, and **App Developer** used by the SVM Streamer and SVM Client to limit session discovery and content consumption to authorized applications. For additional information, see "Content Access Control–Triplet Key" section on page 1-7. The triplet key components are defined as follows:

- **Venue**: A text string associated with the venue where an Cisco StadiumVision Mobile Streamer is hosted.
- **Content Owner**: A text string associated with an entity that wishes to distribute content over the SVM solution.
- **App Developer**: The text string associated with the Application Developer authorized by a Content Owner to consume the Content Owner's content over the SVM solution.

SVM Streamer: A standalone application used to aggregate and send content to mobile applications with an embedded Cisco StadiumVision Mobile Client.

SVM System: An end-to-end solution for the delivery of digital media content streams consisting of specific products (Video Encoder, Cisco StadiumVision Mobile Streamer, Cisco StadiumVision Mobile Reporter), wireline and wireless infrastructure (Connected Stadium, Connected Stadium Wi-Fi) and mobile apps with an embedded Cisco StadiumVision Mobile Client.

Cisco Stadium Vision Mobile Media Input Types

The Cisco StadiumVision Mobile solution can accept multiple forms of media content input (in-house video feed, IP video feed, and IP data feeds). The media is then routed through the appropriate encoder and into the Streamer. The Cisco StadiumVision Mobile Streamer is a critical component in the Cisco StadiumVision Mobile solution that aggregates video, audio, data, and file content streams and supports the following content types:

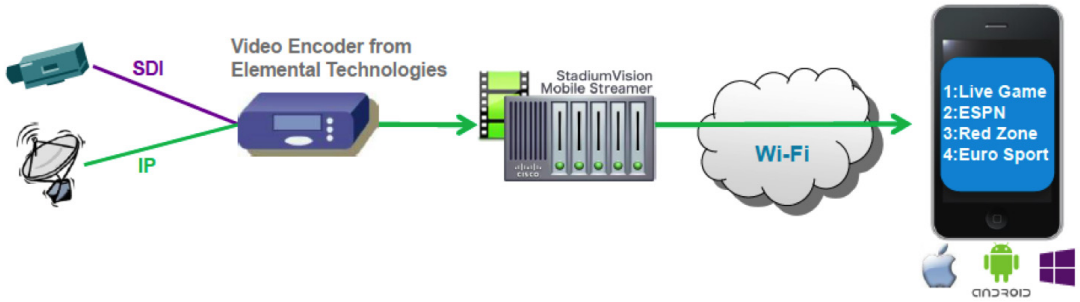
Table 1-1 Channel Types Supported and Use Cases

Channel Type	Maximum Number of Channels	Example Use Cases
Streaming Video	4	• Delivers live real-time venue game feed. • Loops the most recent replays from the live in-venue game. • Provide live out-of-venue game feed taking place at the same time.
Streaming Audio (Available in Android SDK only)	10	• Provides the choice of commentary in multiple languages. • Offers the choice to select between the home team and the away team commentators.
Data	4	• Distributes data through push and pull channels to the client app. **Note** The total number of data channels is 4.
• Data Push		• Triggers all mobile devices to display the same content at the same time, typically used for a sponsored moment of exclusivity.
• Data Pull		• Delivers real-time game statistics overlaid on the video pane.
File	1	• Provides video replays, delivered by EVS C-Cast.

Streaming Video Channels

Figure 1-2 shows the video channel (SDI or IP) inputs in the Cisco StadiumVision Mobile solution. Streaming video can be real-time in-venue game feed, live out of the venue game taking place at the same venue, or loop the most recent replays from the live in-venue game.

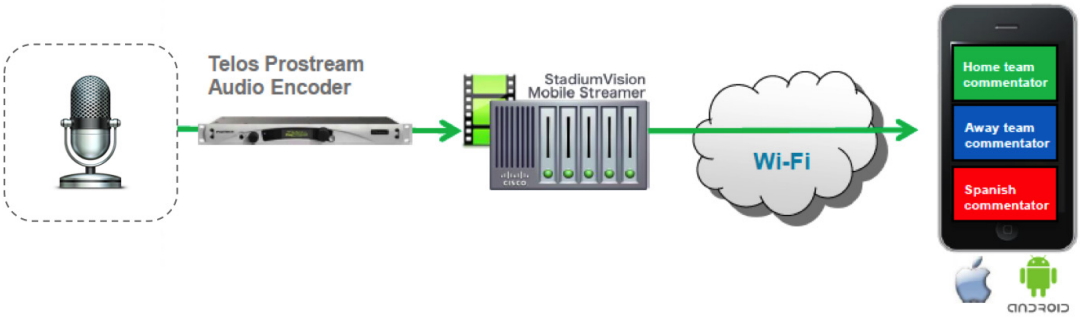
Figure 1-2 Cisco StadiumVision Mobile Streaming Video



Streaming Audio Channels

Figure 1-3 shows how Cisco StadiumVision Mobile allows audio channels to compliment the live game experience. Audio channels consume less bandwidth than video channels, which allows for more room for channels.

Figure 1-3 Cisco StadiumVision Mobile Streaming Audio



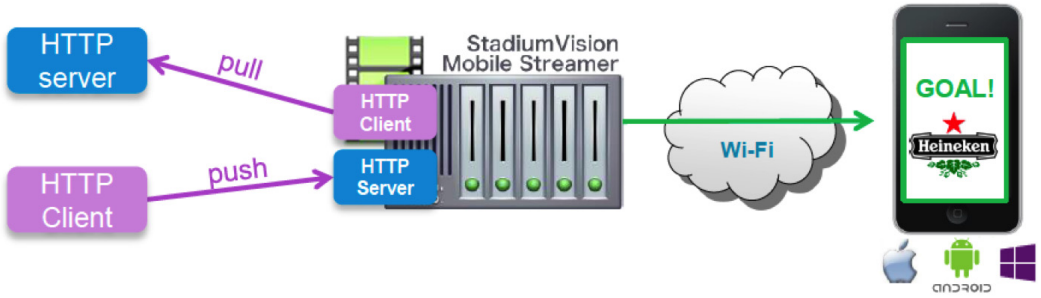
Note Streaming audio is only supported in the Android SDK.

Data Channels

Figure 1-4 shows how Cisco StadiumVision Mobile allows for the data push and pull channels to distribute data of any kind to the client app. The system integrator can decide what types of data is distributed and how it is used by the application. There are two types of data channels:

- Pull: Streamer polls an external web server at regular intervals (default 10 seconds). This channel can be used for data that changes periodically such statistics or thumbnails.
- Push: An external computer pushes data to the Cisco StadiumVision Streamer. This channel can be used for sending on-demand triggers to all mobile devices, for example when a goal is scored.

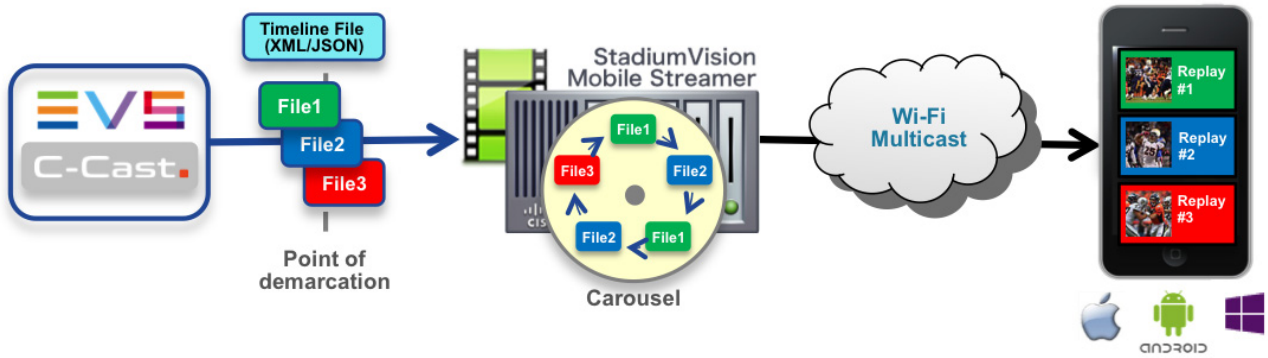
Figure 1-4 Cisco StadiumVision Mobile Data Channels



File Channels

Figure 1-5 shows how Cisco StadiumVision Mobile enables file channels as a way to distribute file-based (video, audio, and data) content to a large number of mobile clients over multicast.

Figure 1-5 Cisco StadiumVision Mobile Files Channels



File Channel Distribution

File channels are often used for replays to mobile devices at a live event where the bottleneck in a stadium is the Wi-Fi network that serves tens of thousands of fans with mobile devices. In order to scale, you can use Cisco StadiumVision Mobile Scalable File Distribution (SFD).

SFD uses multicast over Wi-Fi to scale distribution of the C-Cast video files. Multicast works much like over-the-air broadcast TV where your local TV station sends out a single signal that anyone in the area can receive with an antenna on the roof. From a load perspective it makes no difference to the TV broadcaster if ten subscribers or ten thousand subscribers are watching. Cisco StadiumVision Mobile SFD works in a similar way by sending the files as a single multicast transmission, and any number of mobile devices in the stadium can listen to that signal, receive the file and cache it in local storage for later use.

Generic Ingest

Note Generic ingest is not supported in Cisco StadiumVision Release 2.1.

EVS C-Cast Integration

EVS C-Cast is the platform that Cisco StadiumVision Mobile uses for making replay video clips available to mobile clients over high density Wi-Fi networks. Read more about EVS C-Cast at:

http://www.evs.com/nala/product/c-cast

Note Cisco StadiumVision Mobile is supported with EVS C-Cast version 2.x only. EVS C-Cast version 3.x is not supported.

The traditional way of scaling C-Cast content delivery to a large number of clients is by using a Content Delivery Network (CDN). The CDN caches the content closer to the client, and thus avoids the need for every client to reach back and retrieve the content from the C-Cast Central server. This offloads the C-Cast Central server and reduces the amount of duplicate content that has to traverse the network.

However, the CDN approach does not help in a Wi-Fi environment where the bottleneck is the last 25 meters from an access point to a mobile device. SVM solves this problem by multicasting the C-Cast replays to mobile devices, and therefore avoids sending replays individually to each and every device.

Overview

In a traditional unicast C-Cast deployment (without SVM integration) the client app fetches an XML or JSON formatted C-Cast timeline via HTTP. The timeline describes the sequence of replays (C-Cast calls them events), and the camera angles available for each. Each camera angle consists of a thumbnail graphic and a MP4 video file.

From the perspective of the C-Cast mobile app there is very little difference between the traditional unicast and the Cisco StadiumVision Mobile multicast scenarios. In both cases, the exact same C-Cast timeline provides the app with the info it needs to make replays available to the user. And in both cases, the standard C-Cast media files are used. The only difference between the two scenarios is the transport mechanism used to the deliver the timeline and media files to the mobile devices. And this difference is largely, but not completely, hidden by the Cisco StadiumVision Mobile SDK. The C-Cast timeline is delivered via an SVM data channel and the corresponding media files are delivered via an SVM file channel.

Operation

When an SVM enabled C-Cast app is launched, it starts listening for the timeline on the SVM data channel. By default the timeline is repeated every 10 seconds, so the app may have to wait for up to 10 seconds to receive the latest timeline.

With the timeline in hand, the app extracts the media filenames for each replay and camera angle. The app then queries the SVM SDK to see if the file of interest has been received on the file channel and cached in local storage on the mobile device. If the file is available, the app proceeds to use it.

If a large number of replays and/or camera angles are being distributed, it can take several minutes for all media files to be received by the mobile device. It is therefore entirely possible that the app queries the SVM SDK for a specific file only to find that the file is not yet available. In order to provide a responsive user experience, the app should handle this situation by retrieving the missing file directly from C-Cast via unicast HTTP. The URL for this operation can be constructed from the timeline metadata. Consult the C-Cast API guide for instruction on how to create this URL.

This approach for managing files that are not yet available is referred to as hybrid unicast/multicast. By combining the two we are able to provide a solution that offers scalability and responsiveness.

To obtain the EVS C-Cast API, contact James Stellphlug (j.stellpflug@evs.com) stating that you are developing an app to consume C-Cast clips in a Cisco StadiumVision Mobile venue.

Content Access Control–Triplet Key

An important feature of the Cisco StadiumVision Mobile solution is to limit the consumption of Cisco StadiumVision Mobile encoded content to authorized mobile applications. Consider the following situation:

Content Owner A (e.g., sports team) wishes to use the Cisco StadiumVision Mobile solution to deliver live camera feeds to fans throughout a venue during the team's home games. Content Owner B (e.g., entertainment company) plans to host events at the same venue at a different time and also wishes to deliver live feeds to their fans. The two Content Owners each want to limit content consumption to their chosen and therefore authorized, Application Developer. The reasons for needing to limit content consumption to authorized mobile apps are many. For example, the app might need to be purchased or it may be sponsored by an advertiser. As a result, Cisco StadiumVision Mobile video and data streams configured for Content Owner A's mobile app must not be consumed by Content Owner B's mobile app and vice-versa.

The Cisco StadiumVision Mobile Streamer includes a Triplet (Venue/Content Owner/App Developer) in each announced session. Only mobile apps with the identical Triplet will be able to discover Cisco StadiumVision Mobile sessions and consume the associated content. The Streamer may be configured to support multiple Content Owner and App Developer combinations, though only a single Triplet may be active at any one time.

The Venue, Content Owner, Application Developer (triplet key) settings are critical to enabling content consumption on mobile devices. The Streamer settings must match those used by the App Developer for content to be discovered and consumed by a mobile app. App Developers must be notified of a change in Venue name so that their app may be updated. Conversely, if the App Developer has already deployed the app, App Developers must also be notified if the associated Content Owner/App Developer setting on the Streamer is modified.

Note The Stadium Operator is responsible for correctly configuring the Streamer and working with Content Owner/App Developer to enable content consumption. The Content Owner/App Developer paring must match the values hard coded into the specific SDK for the App Developer contracted for a particular venue.

Additional information regarding the triplet key is available in the *Cisco StadiumVision Mobile Streamer Administration Guide* available on Cisco.com at:

http://www.cisco.com/c/en/us/support/video/stadiumvision/products-maintenance-guides-list.html

Testing Your Cisco StadiumVision Mobile App

The Cisco StadiumVision Mobile SDK includes the ability for developers to test their application without being connected to a Cisco Connected Wi-Fi Network. This capability is provided with a set of files that can emulate the data received over the network. The clean.stream file that comes bundled with the SDK contains just one video channel.

To provide app developers with additional ways to test multiple channels, an additional set of clean.stream files is available for use on Apple iOS and Google Android. This package includes a number of Cisco StadiumVision Mobile stream files for local playback on a mobile device. The stream files enable an SVM app developer to perform some basic testing when access to the Cisco StadiumVision Mobile backend infrastructure is not available.

You can download the set of files at the same location where you obtained the Cisco StadiumVision Mobile SDK tar.bz2 file or contact your Cisco account team for details as to how to become part of the Cisco StadiumVision Mobile SDK partner program by sending an email to:

svm-sdk@external.cisco.com.

We recommend that you get started using these stream files with the included Cisco StadiumVision Mobile demo app before attempting to use it with the app you are developing. For details consult the read-me files included in the root folder of the SDK package.

All stream files are encoded with the default triplet below. The app listening to these stream files must use this exact triplet in order to receive the streams.

```
"license": { "venueName": "VenueName", "contentOwner": "ContentOwner", "appDeveloper":
"AppDeveloper" }
```

The following stream files are included:

- **video.stream:** This stream file contains 4 video channels, as opposed to the one video channel in the stream file that comes bundled with the SDK. The stream file starts off with one channel. Every 10 seconds another channel appears, until all 4 channels are present. The channels then start dropping off one by one, until there are none left.

  Note The video on channel 1 freezes when the channel disappears from the channel lineup. In order to keep the file size reasonable, only one of the channels includes media.

- **audio.stream:** This stream file contains one audio channel.

  Note Only the Android SDK supports audio channels.

- **ccast.stream:** This stream file contains the required streams to test basic EVS C-Cast functionality. The stream file contains these 3 channels:
  - CcastTimeLineXML: Data channel carrying the C-cast timeline in XML format.
  - CcastTimeLineJSON: Data channel carrying the C-cast timeline in JSON format.
  - CcastTimeMedia: File channel carrying the C-Cast mp4 video and jpg thumbnail files.

Note The iOS Sample app 2.1 does not support file channels. The iOS SDK does however have full support file channels.

Please direct your questions to: svm-sdk@external.cisco.com.

Cisco StadiumVision Mobile SDK Best Practices

Apple iOS

Consider the following best practices when developing and delivering an app for Apple iOS:

Correlating Reporter Data to a Specific Device

- Apple does not permit applications to access any device information that can be used to identify that device or its owner. As a result, the iOS SDK is unable to include the MAC address in the periodic stats that it sends to the Cisco StadiumVision Mobile Reporter. As a substitute for the MAC address, the SDK instead includes a SVM Device UUID (universally unique identifier) that is unique for every device. This UUID allows Reporter data to be correlated with a specific device. In order for the correlation to work, the mobile app must display the UUID somewhere in its menu system (for example on the About or Help tabs).

  The app can retrieve the UUID from the SDK via the code sample below. The getDeviceUUID method is documented in the iOS SVM header file.

```
StadiumVisionMobile *svm = [StadiumVisionMobile sharedInstance];
NSString *deviceUUID = [svm getDeviceUUID];
NSLog(@"Device UUID is %@", deviceUUID);
```

Note The Cisco StadiumVision Mobile Device UUID should not be confused with the Unique Device Identifier (UDID) that is displayed in iTunes.

Google Android

Consider the following best practices when developing and delivering an app for Google Android:

Delivering Channel Content

- Internet Group Management Protocol (IGMP) is a prerequisite for Cisco StadiumVision Mobile multicast to function correctly. Most, but not all, Android devices support IGMP. A user will see an empty channel list in the SVM app if they are using a device that does not support IGMP unless another active SVM client is associated to the same Access Point (AP). As a result, the user would experience sporadic channel support without knowing why. We recommend that all SVM-enabled Android apps perform the IGMP capability check as detailed in the example below. If the IGMP capability check returns false, then the app should warn the user when the user attempts to access any part of the app that relies on the SVM SDK.

```
boolean isCapable = false;
File f = new File("/proc/net/igmp");
     if(f.exists()) {
         isCapable = true;
     }
```

Apple iOS, Google Android, and Windows Phone

Consider the following best practices when developing and delivering an app for Apple iOS, Google Android, and Windows Phone:

Delivering Channel Content

- Start the SDK immediately when the app is launched or when the app detects that it is in venue. Do not wait until the user navigates to the SVM features in the UI. This ensures the best possible user experience so that content is then available to present to the user immediately. This is particularly important when using data and file channels because it can take several minutes for a content rotation to complete.
- When a user selects a channel on their mobile device, there could be a 5-10 second delay before the app starts receiving multicast video if the client has to wait for the IGMP/PIM to set up the multicast tree. The same delay could occur when a device resumes from background or sleep mode. Instead of displaying a black screen, communicate to the user that their request is being processed by showing a transition graphic (for example a spinning wheel image) or text that asks them to please wait while the channel is located.
- A channel will disappear from the lineup if it is stopped on the Streamer or if the Streamer detects a loss of input signal. The app should remove the channel immediately, however if a user is already watching the channel when it disappears, terminate the video rendering and return the user to the channel guide where they can select a new channel.
- If the Cisco StadiumVision Mobile administrator starts the Streamer before the video control room has switched content to the encoder inputs, the user will receive an empty channel listing. To avoid a poor user experience, display a message indicating that there aren't any live channels currently available.

Using the Latest Version of the app

- Prevent or warn a user if they are attempting to access SVM services with an older or incompatible version of the app. Set the app to perform a self-check to see if a newer app version is available. If a new version is detected, the app should:
  - Block or warn the user that their app is out-of-date or may not perform as expected
  - Encourage the user to upgrade

Connecting to Wi-Fi

- If the client looses complete Wi-Fi connectivity, the Operating System sends the app a notification. The app should notify the user that the Wi-Fi service is not available and remind them of the Wi-Fi network (SSID) to connect to.
- If a Streamer service announcement has not been received for 30 seconds, the SDK will notify the user that the Cisco StadiumVision Mobile service is down. This could happen because the user exited the venue and is out of range, if their device roamed to a non SVM SSID, or if they entered an area in the venue without Wi-Fi. When this occurs, notify the user that the SVM service is not available and remind them of the name of the Wi-Fi network (SSID) that they must be connected to in order to receive the SVM service.
- The SDK continuously monitors signal quality as a user moves throughout a venue. As a result, the SDK sends the app a 'service down' or 'service up' notification. These notifications can be used by the app for conditional playback based on network conditions. It is recommended that conditional playback is implemented as a passive informational service (as it should never display prompts that require an active response from the user). Refer to Table 1-2 for app guidelines as to how the app should respond depending on the state of the app when the notification is received.

Table 1-2 App Guidelines for Responding to Notifications

app State	Event or Notification	Recommended app Response
The app is rendering a video.	Service down notification, due to poor quality.	Overlay the active video plane with a transparent graphic that contains the text such as, "Video quality degraded due to poor reception". The app continues the video session without interruption regardless of the poor quality and renders whatever video it receives underneath the text overlay.
The app previously received a service down notification with the poor quality reason code. It is currently rendering degraded video with the "Video quality degraded due to poor reception" overlay.	Service down notification, due to poor quality.	Remove the overlay and resume normal video rendering. Do not restart the video session.
The app is currently not rendering video.	Service down notification, due to poor quality.	Delete all channels from the channel list page and replace them with a message such as "Live video is currently unavailable due to poor reception".
The app is currently not rendering video. The app had previously received a service down notification with the poor quality reason code and is currently displaying the message "Live video unavailable due to poor reception" in the channel guide.	Service up notification, due to quality rebounding.	Remove the "Live video unavailable due to poor reception" message and populate the channel page with the current list of channels available.
The app previously received a service down notification with the poor quality reason code and is currently rendering video with the "Video quality degraded due to poor reception" overlay.	The user now stops the video and returns to the channel page.	Upon returning to the channel page, the user sees the message "Live video unavailable due to poor reception" instead of a list of channels.