# Filters and Dynamic Manifests

Starting with 2.11 release, Media Services enables you to define filters for your assets. These filters are server side rules that will allow your customers to choose to do things like: playback only a section of a video (instead of playing the whole video), or specify only a subset of audio and video renditions that your customer’s device can handle (instead of all the renditions that are associated with the asset). This filtering of your assets is archived through **Dynamic Manifest**s that are created upon your customer’s request to stream a video based on specified filter(s).

This topics discusses common scenarios in which using filters would be very beneficial to your customers and links to topics that demonstrate how to create filters programmatically (currently, you can create filters with REST APIs only).

## Overview

When delivering your content to customers (streaming live events or video-on-demand) your goal is to deliver a high quality video to various devices under different network conditions. To achieve this goal do the following:

* encode your stream to multi-bitrate ([adaptive bitrate](http://en.wikipedia.org/wiki/Adaptive_bitrate_streaming)) video stream (this will take care of quality and network conditions) and
* use Media Services [Dynamic Packaging](/documentation/articles/media-services-dynamic-packaging-overview) to dynamically re-package your stream into different protocols (this will take care of streaming on different devices). Media Services supports delivery of the following adaptive bitrate streaming technologies: HTTP Live Streaming (HLS), Smooth Streaming, MPEG DASH, and HDS (for Adobe PrimeTime/Access licensees only).

### Manifest files

When you encode an asset for adaptive bitrate streaming, a **manifest** (playlist) file is created (the file is text-based or XML-based). The **manifest** file includes streaming metadata such as: track type (audio, video, or text), track name, start and end time, bitrate (qualities), track languages, presentation window (sliding window of fixed duration), video codec (FourCC). It also instructs the player to retrieve the next fragment by providing information about the next playable video fragments available and their location. Fragments (or segments) are the actual “chunks” of a video content.

Here is an example of a manifest file:

<?xml version="1.0" encoding="UTF-8"?>   
<SmoothStreamingMedia MajorVersion="2" MinorVersion="0" Duration="330187755" TimeScale="10000000">  
  
<StreamIndex Chunks="17" Type="video" Url="QualityLevels({bitrate})/Fragments(video={start time})" QualityLevels="8">  
<QualityLevel Index="0" Bitrate="5860941" FourCC="H264" MaxWidth="1920" MaxHeight="1080" CodecPrivateData="0000000167640028AC2CA501E0089F97015202020280000003008000001931300016E360000E4E1FF8C7076850A4580000000168E9093525" />  
<QualityLevel Index="1" Bitrate="4602724" FourCC="H264" MaxWidth="1920" MaxHeight="1080" CodecPrivateData="0000000167640028AC2CA501E0089F97015202020280000003008000001931100011EDC00002CD29FF8C7076850A45800000000168E9093525" />  
<QualityLevel Index="2" Bitrate="3319311" FourCC="H264" MaxWidth="1280" MaxHeight="720" CodecPrivateData="000000016764001FAC2CA5014016EC054808080A00000300020000030064C0800067C28000103667F8C7076850A4580000000168E9093525" />  
<QualityLevel Index="3" Bitrate="2195119" FourCC="H264" MaxWidth="960" MaxHeight="540" CodecPrivateData="000000016764001FAC2CA503C045FBC054808080A000000300200000064C1000044AA0000ABA9FE31C1DA14291600000000168E9093525" />  
<QualityLevel Index="4" Bitrate="1469881" FourCC="H264" MaxWidth="960" MaxHeight="540" CodecPrivateData="000000016764001FAC2CA503C045FBC054808080A000000300200000064C04000B71A0000E4E1FF8C7076850A4580000000168E9093525" />  
<QualityLevel Index="5" Bitrate="978815" FourCC="H264" MaxWidth="640" MaxHeight="360" CodecPrivateData="000000016764001EAC2CA50280BFE5C0548303032000000300200000064C08001E8480004C4B7F8C7076850A45800000000168E9093525" />  
<QualityLevel Index="6" Bitrate="638374" FourCC="H264" MaxWidth="640" MaxHeight="360" CodecPrivateData="000000016764001EAC2CA50280BFE5C0548303032000000300200000064C080013D60000C65DFE31C1DA1429160000000168E9093525" />  
<QualityLevel Index="7" Bitrate="388851" FourCC="H264" MaxWidth="320" MaxHeight="180" CodecPrivateData="000000016764000DAC2CA505067E7C054830303200000300020000030064C040030D40003D093F8C7076850A45800000000168E9093525" />  
  
<c t="0" d="20000000" /><c d="20000000" /><c d="20000000" /><c d="20000000" /><c d="20000000" /><c d="20000000" /><c d="20000000" /><c d="20000000" /><c d="20000000" /><c d="20000000" /><c d="20000000" /><c d="20000000" /><c d="20000000" /><c d="20000000" /><c d="20000000" /><c d="20000000" /><c d="9600000"/>  
</StreamIndex>  
  
  
<StreamIndex Chunks="17" Type="audio" Url="QualityLevels({bitrate})/Fragments(AAC\_und\_ch2\_128kbps={start time})" QualityLevels="1" Name="AAC\_und\_ch2\_128kbps">  
<QualityLevel AudioTag="255" Index="0" BitsPerSample="16" Bitrate="125658" FourCC="AACL" CodecPrivateData="1210" Channels="2" PacketSize="4" SamplingRate="44100" />  
  
<c t="0" d="20201360" /><c d="20201361" /><c d="20201360" /><c d="20201361" /><c d="20201360" /><c d="20201361" /><c d="20201360" /><c d="20201361" /><c d="20201360" /><c d="20201361" /><c d="20201360" /><c d="20201361" /><c d="20201361" /><c d="20201360" /><c d="20201361" /><c d="20201360" /><c d="6965987" /></StreamIndex>  
  
  
<StreamIndex Chunks="17" Type="audio" Url="QualityLevels({bitrate})/Fragments(AAC\_und\_ch2\_56kbps={start time})" QualityLevels="1" Name="AAC\_und\_ch2\_56kbps">  
<QualityLevel AudioTag="255" Index="0" BitsPerSample="16" Bitrate="53655" FourCC="AACL" CodecPrivateData="1210" Channels="2" PacketSize="4" SamplingRate="44100" />  
  
<c t="0" d="20201360" /><c d="20201361" /><c d="20201360" /><c d="20201361" /><c d="20201360" /><c d="20201361" /><c d="20201360" /><c d="20201361" /><c d="20201360" /><c d="20201361" /><c d="20201360" /><c d="20201361" /><c d="20201361" /><c d="20201360" /><c d="20201361" /><c d="20201360" /><c d="6965987" /></StreamIndex>  
  
</SmoothStreamingMedia>

### Dynamic manifests

There are [scenarios](/documentation/articles/media-services-dynamic-manifest-overview#scenarios) when your client needs more flexibility than what’s described in the default asset’s manifest file. For example:

* Device specific: deliver only the specified renditions andspecified language tracks that are supported by the device that is used to playback the content (“rendition filtering”).
* Reduce the manifest to show a sub-clip of a live event (“sub-clip filtering”).
* Trim the start of a video (“trimming a video”).
* Adjust Presentation Window (DVR) in order to provide a limited length of the DVR window in the player (“adjusting presentation window”) .

To achieve this flexibility, Media Services offers **Dynamic Manifests** based on pre-defined [filters](/documentation/articles/media-services-dynamic-manifest-overview#filters). Once you define the filters, your clients could use them to stream a specific rendition or sub-clips of your video. They would specify filter(s) in the streaming URL. Filters could be applied to adaptive bitrate streaming protocols supported by [Dynamic Packaging](/documentation/articles/media-services-dynamic-packaging-overview): HLS, MPEG-DASH, Smooth Streaming, and HDS. For example:

MPEG DASH URL with filter

http://testendpoint-testaccount.streaming.mediaservices.chinacloudapi.cn/fecebb23-46f6-490d-8b70-203e86b0df58/BigBuckBunny.ism/Manifest(format=mpd-time-csf,filter=MyLocalFilter)

Smooth Streaming URL with filter

http://testendpoint-testaccount.streaming.mediaservices.chinacloudapi.cn/fecebb23-46f6-490d-8b70-203e86b0df58/BigBuckBunny.ism/Manifest(filter=MyLocalFilter)

For more information about how to deliver your content and build streaming URLs, see [Delivering content overview](/documentation/articles/media-services-deliver-content-overview).

[AZURE.NOTE]Note that Dynamic Manifests do not change the asset and the default manifest for that asset. Your client can choose to request a stream with or without filters.

### Filters

There are two types of asset filters:

* Global filters (can be applied to any asset in the Azure Media Services account, have a lifetime of the account) and
* Local filters (can only be applied to an asset with which the filter was associated upon creation, have a lifetime of the asset).

Global and local filter types have exactly the same properties. The main difference between the two is for which scenarios what type of a filer is more suitable. Global filters are generally suitable for device profiles (rendition filtering) where local filters could be used to trim a specific asset.

## Common scenarios

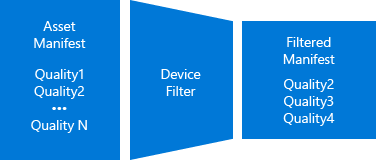
As was mentioned before, when delivering your content to customers (streaming live events or video-on-demand) your goal is to deliver a high quality video to various devices under different network conditions. In addition, your might have other requirements that involve filtering your assets and using of **Dynamic Manifest**s. The following sections give a short overview of different filtering scenarios.

* Specify only a subset of audio and video renditions that certain devices can handle (instead of all the renditions that are associated with the asset).
* Playing back only a section of a video (instead of playing the whole video).
* Adjust DVR presentation window.

## Rendition filtering

You may choose to encode your asset to multiple encoding profiles (H.264 Baseline, H.264 High, AACL, AACH, Dolby Digital Plus) and multiple quality bitrates. However, not all client devices will support all your asset’s profiles and bitrates. For example, older Android devices only supports H.264 Baseline+AACL. Sending higher bitrates to a device which cannot get the benefits, wastes bandwidth and device computation. Such device must decode all the given information, only to scale it down for display.

With Dynamic Manifest, you can create device profiles such as mobile, console, HD/SD, etc. and include the tracks and qualities which you want to be a part of each profile.



Rendition filtering example

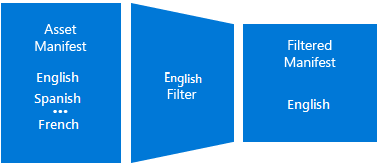
In the following example, Azure Media Encoder was used to encode a mezzanine asset into seven ISO MP4s video renditions (from 180p to 1080p). The encoded asset can be dynamically packaged into any of the following streaming protocols: HLS, Smooth, MPEG DASH, and HDS. At the top of the diagram, the HLS manifest for the asset with no filters is shown (it contains all seven renditions). In the bottom left, the HLS manifest to which a filter named “ott” was applied is shown. The “ott” filter specifies to remove all bitrates below 1Mbps, which resulted in the bottom two quality levels being stripped off in the response. In the bottom right, the HLS manifest to which a filter named “mobile” was applied is shown. The “mobile” filter specifies to remove renditions where the resolution is larger than 720p, which resulted in the two 1080p renditions being stripped off.



Rendition filtering

## Removing language tracks

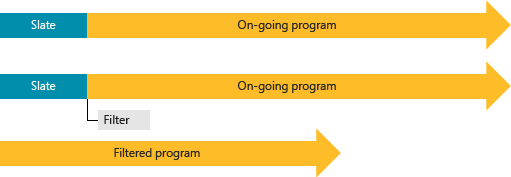
Your assets might include multiple audio languages such as English, Spanish, French, etc. Usually, the Player SDK managers default audio track selection and available audio tracks per user selection. It is challenging to develop such Player SDKs, it requires different implementations across device-specific player-frameworks. Also, on some platforms, Player APIs are limited and do not include audio selection feature where users cannot select or change the default audio track. With asset filters, you can control the behavior by creating filters that only include desired audio languages.



Language tracks filtering

## Trimming start of an asset

In most live streaming events, operators run some tests before the actual event. For example, they might include a slate like this before the start of the event: “Program will begin momentarily”. If the program is archiving, the test and slate data are also archived and will be included in the presentation. However, this information should not be shown to the clients. With Dynamic Manifest, you can create a start time filter and remove the unwanted data from the manifest.



Trimming start

## Creating sub-clips (views) from a live archive

Many live events are long running and live archive might include multiple events. After the live event ends broadcasters may want to break up the live archive into logical program start and stop sequences. Then, publish these virtual programs separately without post processing the live archive and not creating separate assets (which will not get benefit of the existing cached fragments in the CDNs). Examples of such virtual programs (sub-clips) are the quarters of a football or basketball game, the innings in baseball, or individual events of an afternoon of Olympics program.

With Dynamic Manifest, you can create filters using start/end times and create virtual views over the top of your live archive.



Subclip filter

Filtered Asset:



Skiing

## Adjusting Presentation Window (DVR)

Currently, Azure Media Services offers circular archive where the duration can be configured between 5 minutes - 25 hours. Manifest filtering can be used to create a rolling DVR window over the top of the archive, without deleting media. There are many scenarios where broadcasters want to provide a limited DVR window which moves with the live edge and at the same time keep a bigger archiving window. A broadcaster may want to use the data that is out of the DVR window to highlight clips, or hemay want to provide different DVR windows for different devices. For example, most of the mobile devices don’t handle big DVR windows (you can have a 2 minute DVR window for mobile devices and 1 hour for desktop clients).



DVR window

## Adjusting LiveBackoff (live position)

Manifest filtering can be used to remove several seconds from the live edge of a live program. This allows broadcasters to watch the presentation on the preview publication point and create advertisement insertion points before the viewers receive the stream (usually backed-off by 30 seconds). Broadcasters can then push these advertisements to their client frameworks in time for them to received and process the information before the advertisement opportunity.

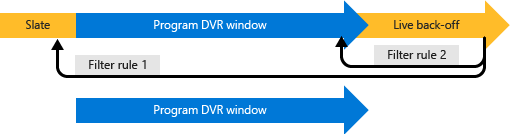
In addition to the advertisement support, LiveBackoff can be used for adjusting client live download position so that when clients drift and hit the live edge they can still get fragments from server instead of getting 404 or 412 HTTP errors.



livebackoff\_filter

## Combining multiple rules in a single filter

You can combine multiple filtering rules in a single filter. As an example you can define a range rule to remove slate from a live archive and also filter available bitrates. For multiple filtering rules the end result is the composition (intersection only) of these rules.



multiple-rules

## Create filters programmatically

The following topic discusses Media Services entities that are related to filters. The topic also shows how to programmatically create filters.

[Create filters with REST APIs](/documentation/articles/media-services-rest-dynamic-manifest).

## Combining multiple filters (filter composition)

You can also combine multiple filters in a single URL.

The following scenario demonstrates why you might want to combine filters:

1. You need to filter your video qualities for mobile devices such as Android or iPAD (in order to limit video qualities). To remove the unwanted qualities, you would create a global filter which is suitable for device profiles. As mentioned above, global filters can be used for all your assets under the same media services account without any further association.
2. You also want to trim the start and end time of an asset. To achieve this, you would create a local filter and set the start/end time.
3. You want to combine both of these filters (without combination you would need to add quality filtering to the trimming filter which will make filter usage difficult).

To combine filters, you need to set the filter names to the manifest/playlist URL with semicolon delimited. Let’s assume you have a filter named *MyMobileDevice* that filters qualities and you have another named *MyStartTime* to set a specific start time. You can combine them like this:

http://teststreaming.streaming.mediaservices.chinacloudapi.cn/3d56a4d-b71d-489b-854f-1d67c0596966/64ff1f89-b430-43f8-87dd-56c87b7bd9e2.ism/Manifest(filter=MyMobileDevice;MyStartTime)

You can combine up to 3 filters.

For more information see [this](http://azure.microsoft.com/blog/azure-media-services-release-dynamic-manifest-composition-remove-hls-audio-only-track-and-hls-i-frame-track-support) blog.

## Know issues and limitations

* Dynamic manifest operates in GOP boundaries (Key Frames) hence trimming has GOP accuracy.
* You can use same filter name for local and global filters. Note that local filter have higher precedence and will override global filters.
* If you update a filter, it can take up to 2 minutes for streaming endpoint to refresh the rules. If the content was served using some filters (and cached in proxies and CDN caches), updating these filters can result in player failures. It is recommend to clear the cache after updating the filter. If this option is not possible, consider using a different filter.

## See Also

[Delivering Content to Customers Overview](/documentation/articles/media-services-deliver-content-overview)