# How to generate thumbnails using Media Encoder Standard with .NET

This topic shows how to use Media Services .NET SDK to encode an asset and generate thumbnails using Media Encoder Standard. The topic defines the XML and JSON thumbnail presets that you can use to create a task that does encoding and generates thumbnails at the same time. [This](https://msdn.microsoft.com/zh-cn/library/mt269962.aspx) document contains descriptions of elements that are used by these presets.

Make sure to review the [Considerations](/documentation/articles/media-services-dotnet-generate-thumbnail-with-mes#considerations) section.

## Example

The following code example uses Media Services .NET SDK to perform the following tasks:

* Create an encoding job.
* Get a reference to the Media Encoder Standard encoder.
* Load the preset [XML](/documentation/articles/media-services-dotnet-generate-thumbnail-with-mes#xml) or [JSON](/documentation/articles/media-services-dotnet-generate-thumbnail-with-mes#json) that contain the encoding preset as well as information needed to generate thumbnails. You can save this [XML](/documentation/articles/media-services-dotnet-generate-thumbnail-with-mes#xml) or [JSON](/documentation/articles/media-services-dotnet-generate-thumbnail-with-mes#json) in a file and use the following code to load the file.
* // Load the XML (or JSON) from the local file.  
   string configuration = File.ReadAllText(fileName);
* Add a single encoding task to the job.
* Specify the input asset to be encoded.
* Create an output asset that will contain the encoded asset.
* Add an event handler to check the job progress.
* Submit the job.
* using System;  
  using System.Collections.Generic;  
  using System.Configuration;  
  using System.IO;  
  using System.Linq;  
  using System.Net;  
  using System.Security.Cryptography;  
  using System.Text;  
  using System.Threading.Tasks;  
  using Microsoft.WindowsAzure.MediaServices.Client;  
  using Newtonsoft.Json.Linq;  
  using System.Threading;  
  using Microsoft.WindowsAzure.MediaServices.Client.ContentKeyAuthorization;  
  using Microsoft.WindowsAzure.MediaServices.Client.DynamicEncryption;  
  using System.Web;  
  using System.Globalization;  
    
  namespace EncodeAndGenerateThumbnails  
  {  
   class Program  
   {  
   // Read values from the App.config file.  
   private static readonly string \_mediaServicesAccountName =  
   ConfigurationManager.AppSettings["MediaServicesAccountName"];  
   private static readonly string \_mediaServicesAccountKey =  
   ConfigurationManager.AppSettings["MediaServicesAccountKey"];  
    
   // Field for service context.  
   private static CloudMediaContext \_context = null;  
   private static MediaServicesCredentials \_cachedCredentials = null;  
    
   private static readonly string \_mediaFiles =  
   Path.GetFullPath(@"../..\Media");  
    
   private static readonly string \_singleMP4File =  
   Path.Combine(\_mediaFiles, @"BigBuckBunny.mp4");  
    
   static void Main(string[] args)  
   {  
   // Create and cache the Media Services credentials in a static class variable.  
   \_cachedCredentials = new MediaServicesCredentials(  
   \_mediaServicesAccountName,  
   \_mediaServicesAccountKey);  
   // Used the chached credentials to create CloudMediaContext.  
   \_context = new CloudMediaContext(\_cachedCredentials);  
    
   // Get an uploaded asset.  
   var asset = \_context.Assets.FirstOrDefault();  
    
   // Encode and generate the thumbnails.  
   EncodeToAdaptiveBitrateMP4Set(asset);  
    
   Console.ReadLine();  
   }  
    
   static public IAsset EncodeToAdaptiveBitrateMP4Set(IAsset asset)  
   {  
   // Declare a new job.  
   IJob job = \_context.Jobs.Create("Media Encoder Standard Job");  
   // Get a media processor reference, and pass to it the name of the   
   // processor to use for the specific task.  
   IMediaProcessor processor = GetLatestMediaProcessorByName("Media Encoder Standard");  
    
    
   // Load the XML (or JSON) from the local file.  
   string configuration = File.ReadAllText("ThumbnailPreset\_JSON.json");  
    
   // Create a task  
   ITask task = job.Tasks.AddNew("Media Encoder Standard encoding task",  
   processor,  
   configuration,  
   TaskOptions.None);  
    
   // Specify the input asset to be encoded.  
   task.InputAssets.Add(asset);  
   // Add an output asset to contain the results of the job.   
   // This output is specified as AssetCreationOptions.None, which   
   // means the output asset is not encrypted.   
   task.OutputAssets.AddNew("Output asset",  
   AssetCreationOptions.None);  
    
   job.StateChanged += new EventHandler<JobStateChangedEventArgs>(JobStateChanged);  
   job.Submit();  
   job.GetExecutionProgressTask(CancellationToken.None).Wait();  
    
   return job.OutputMediaAssets[0];  
   }  
    
   private static void JobStateChanged(object sender, JobStateChangedEventArgs e)  
   {  
   Console.WriteLine("Job state changed event:");  
   Console.WriteLine(" Previous state: " + e.PreviousState);  
   Console.WriteLine(" Current state: " + e.CurrentState);  
   switch (e.CurrentState)  
   {  
   case JobState.Finished:  
   Console.WriteLine();  
   Console.WriteLine("Job is finished. Please wait while local tasks or downloads complete...");  
   break;  
   case JobState.Canceling:  
   case JobState.Queued:  
   case JobState.Scheduled:  
   case JobState.Processing:  
   Console.WriteLine("Please wait...\n");  
   break;  
   case JobState.Canceled:  
   case JobState.Error:  
    
   // Cast sender as a job.  
   IJob job = (IJob)sender;  
    
   // Display or log error details as needed.  
   break;  
   default:  
   break;  
   }  
   }  
    
    
   private static IMediaProcessor GetLatestMediaProcessorByName(string mediaProcessorName)  
   {  
   var processor = \_context.MediaProcessors.Where(p => p.Name == mediaProcessorName).  
   ToList().OrderBy(p => new Version(p.Version)).LastOrDefault();  
    
   if (processor == null)  
   throw new ArgumentException(string.Format("Unknown media processor", mediaProcessorName));  
    
   return processor;  
   }  
    
   }  
  }

## Thumbnail JSON preset

For information about schema, see [this](https://msdn.microsoft.com/zh-cn/library/mt269962.aspx) topic.

{  
 "Version": 1.0,  
 "Codecs": [  
 {  
 "KeyFrameInterval": "00:00:02",  
 "SceneChangeDetection": "true",  
 "H264Layers": [  
 {  
 "Profile": "Auto",  
 "Level": "auto",  
 "Bitrate": 4500,  
 "MaxBitrate": 4500,  
 "BufferWindow": "00:00:05",  
 "Width": 1280,  
 "Height": 720,  
 "ReferenceFrames": 3,  
 "EntropyMode": "Cabac",  
 "AdaptiveBFrame": true,  
 "Type": "H264Layer",  
 "FrameRate": "0/1"  
   
 }  
 ],  
 "Type": "H264Video"  
 },  
 {  
 "JpgLayers": [  
 {  
 "Quality": 90,  
 "Type": "JpgLayer",  
 "Width": 640,  
 "Height": 360  
 }  
 ],  
 "Start": "{Best}",  
 "Type": "JpgImage"  
 },  
 {  
 "PngLayers": [  
 {  
 "Type": "PngLayer",  
 "Width": 640,  
 "Height": 360,  
 }  
 ],  
 "Start": "00:00:01",  
 "Step": "00:00:10",  
 "Range": "00:00:58",  
 "Type": "PngImage"  
 },  
 {  
 "BmpLayers": [  
 {  
 "Type": "BmpLayer",  
 "Width": 640,  
 "Height": 360  
 }  
 ],  
 "Start": "10%",  
 "Step": "10%",  
 "Range": "90%",  
 "Type": "BmpImage"  
 },  
 {  
 "Channels": 2,  
 "SamplingRate": 48000,  
 "Bitrate": 128,  
 "Type": "AACAudio"  
 }  
 ],  
 "Outputs": [  
 {  
 "FileName": "{Basename}\_{Index}{Extension}",  
 "Format": {  
 "Type": "JpgFormat"  
 }  
 },  
 {  
 "FileName": "{Basename}\_{Index}{Extension}",  
 "Format": {  
 "Type": "PngFormat"  
 }  
 },  
 {  
 "FileName": "{Basename}\_{Index}{Extension}",  
 "Format": {  
 "Type": "BmpFormat"  
 }  
 },  
 {  
 "FileName": "{Basename}\_{Width}x{Height}\_{VideoBitrate}.mp4",  
 "Format": {  
 "Type": "MP4Format"  
 }  
 }  
 ]  
}

## Thumbnail XML preset

For information about schema, see [this](https://msdn.microsoft.com/zh-cn/library/mt269962.aspx) topic.

<?xml version="1.0" encoding="utf-16"?>  
<Preset xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" Version="1.0" xmlns="http://azure.microsoft.com/media/encoding/Preset/2014/03">  
 <Encoding>  
 <H264Video>  
 <KeyFrameInterval>00:00:02</KeyFrameInterval>  
 <SceneChangeDetection>true</SceneChangeDetection>  
 <H264Layers>  
 <H264Layer>  
 <Bitrate>4500</Bitrate>  
 <Width>1280</Width>  
 <Height>720</Height>  
 <FrameRate>0/1</FrameRate>  
 <Profile>Auto</Profile>  
 <Level>auto</Level>  
 <BFrames>3</BFrames>  
 <ReferenceFrames>3</ReferenceFrames>  
 <Slices>0</Slices>  
 <AdaptiveBFrame>true</AdaptiveBFrame>  
 <EntropyMode>Cabac</EntropyMode>  
 <BufferWindow>00:00:05</BufferWindow>  
 <MaxBitrate>4500</MaxBitrate>  
 </H264Layer>  
 </H264Layers>  
 <Chapters />  
 </H264Video>  
 <AACAudio>  
 <Profile>AACLC</Profile>  
 <Channels>2</Channels>  
 <SamplingRate>48000</SamplingRate>  
 <Bitrate>128</Bitrate>  
 </AACAudio>  
 <JpgImage Start="{Best}">  
 <JpgLayers>  
 <JpgLayer>  
 <Width>640</Width>  
 <Height>360</Height>  
 <Quality>90</Quality>  
 </JpgLayer>  
 </JpgLayers>  
 </JpgImage>  
 <BmpImage Start="10%" Step="10%" Range="90%">  
 <BmpLayers>  
 <BmpLayer>  
 <Width>640</Width>  
 <Height>360</Height>  
 </BmpLayer>  
 </BmpLayers>  
 </BmpImage>  
 <PngImage Start="00:00:01" Step="00:00:10" Range="00:00:58">  
 <PngLayers>  
 <PngLayer>  
 <Width>640</Width>  
 <Height>360</Height>  
 </PngLayer>  
 </PngLayers>  
 </PngImage>  
 </Encoding>  
 <Outputs>  
 <Output FileName="{Basename}\_{Width}x{Height}\_{VideoBitrate}.mp4">  
 <MP4Format />  
 </Output>  
 <Output FileName="{Basename}\_{Index}{Extension}">  
 <JpgFormat />  
 </Output>  
 <Output FileName="{Basename}\_{Index}{Extension}">  
 <BmpFormat />  
 </Output>  
 <Output FileName="{Basename}\_{Index}{Extension}">  
 <PngFormat />  
 </Output>  
 </Outputs>  
</Preset>

## Considerations

The following considerations apply:

* The use of explicit timestamps for Start/Step/Range assumes that the input source is at least 1 minute long.
* Jpg/Png/BmpVideo elements have Start, Step and Range string attributes – these can be interpreted as:
  + Frame Number if they are non-negative integers, eg. “Start”: “120”,
  + Relative to source duration if expressed as %-suffixed, eg. “Start”: “15%”, OR
  + Timestamp if expressed as HH:MM:SS… format. Eg. “Start” : “00:01:00”
* You can mix and match notations as you please.
* Additionally, Start also supports a special Macro:{Best}, which attempts to determine the first “interesting” frame of the content NOTE: (Step and Range are ignored when Start is set to {Best})
  + Defaults: Start:{Best}
* Output format needs to be explicitly provided for each Image format: Jpg/Png/BmpFormat. When present, AMS will match JpgVideo to JpgFormat and so on. OutputFormat introduces a new image-codec specific Macro: {Index}, which needs to be present (once and only once) for image output formats.

## See Also

[Media Services Encoding Overview](/documentation/articles/media-services-encode-asset)