title: Media Services v2 to v3 migration checklist | Microsoft Docs description: This article is a checklist that will assist you min migrating from Azure Media Services v2 to v3. services: media-services documentationcenter: na author: IngridAtMicrosoft manager: femila editor: '' tags: '' keywords: azure media services, stream, broadcast, live, offline

## ms.service: media-services ms.devlang: multiple ms.topic: conceptual ms.tgt\_pltfrm: multiple ms.workload: media ms.date: 11/07/2020 ms.author: inhenkel

# Media Services migration checklist

This article is intended as step-by-step guide for migrating from Media Services V2 to V3. Although it is somewhat prescriptive, you should use your best judgement and determine what best fits your scenario. The recommendations are based on a migration that takes advantage of the new features and functions now available in the V3 API.

We encourage you to start using version 2020-05-01 of the Azure Media Services v3 API now to gain the benefits of the new functionality, such as encryption at rest with customer managed keys, trusted storage with support for vNet and firewalls, system managed identities, private link, live transcription, 24x7 live streaming, and much more. Also, new features and functionality and performance optimizations are only available in the current v3 API.

**NOTE**: After February 29, 2024, Azure Media Services will no longer support the v2 API versions. If you prefer not to upgrade, requests sent from earlier versions will continue to be served by the Azure Media Services until February 29, 2024.

# Migration Steps

## Getting Started with migration to the v3 API

**Step 1:** First migrate to the latest v3 REST API, or client SDK of your choice. After upgrading to the latest client SDK, customers should next go into the portal for their media service accounts, select API Access and Service Principal authentication to generate a new AAD application ID and secrets for use with the v3 SDKs. Then, understand the differences between the Media Services V2 and V3 APIs.

**Step 2:**Review the scenario-based migration guidance in this article for details on moving your encoding, live streaming, on-demand streaming, or content protection use case scenarios from the v2 API to the new v3 API.

* Live Streaming
* Encoding
* On-demand streaming
* Content protection
* Audio Indexing
* Face Redaction

## Understand the [differences](file:///C:/Users/inhenkel/Documents/GitHub/azure-docs-pr/articles/media-services/latest/migrate-v-2-v-3-differences-description.md) between the Media Services V2 and V3 APIs.

1. The V3 API is now ARM based and uses AAD Service Principal IDs and Keys to connect to the API. Developers will need to create Service Principals or Managed Identities to connect to the API.
2. There are more official client languages available for the v3 API, however, there is no longer a community PHP SDK available for Azure Media Services on v3. Customer using PHP on v2 should migrate to the REST API directly in their code.

### How to move from the v2 .NET SDK to the v3 .NET SDK

In the V2 API, there were two different client SDKs available. One for the management API which allowed programmatic creation of accounts, and one for the resource management.   
Previously developers would work with an Azure Service Principal Client ID and Client Secret, along with a specific v2 REST API endpoint for their AMS account.

The following code shows how the v2 API was previously accessed using the v2 .NET SDK. In the v2 API developers would begin with creating a **CloudMediaContext** and create an instance with an **AzureAdTokenCredentials** object.

You can search your code base for instances of **CloudMediaContext** usage to begin the upgrade process to the V3 API.

###### Legacy v2 method of connecting to Azure Media Services using the .NET SDK

class Program

{

// Read values from the App.config file.

private static readonly string \_AADTenantDomain =

ConfigurationManager.AppSettings["AMSAADTenantDomain"];

private static readonly string \_RESTAPIEndpoint =

ConfigurationManager.AppSettings["AMSRESTAPIEndpoint"];

private static readonly string \_AMSClientId =

ConfigurationManager.AppSettings["AMSClientId"];

private static readonly string \_AMSClientSecret =

ConfigurationManager.AppSettings["AMSClientSecret"];

private static CloudMediaContext \_context = null;

static void Main(string[] args)

{

try

{

AzureAdTokenCredentials tokenCredentials =

new AzureAdTokenCredentials(\_AADTenantDomain,

new AzureAdClientSymmetricKey(\_AMSClientId, \_AMSClientSecret),

AzureEnvironments.AzureCloudEnvironment);

var tokenProvider = new AzureAdTokenProvider(tokenCredentials);

\_context = new CloudMediaContext(new Uri(\_RESTAPIEndpoint), tokenProvider);

...

}

In the new v3 API, the API now uses the standard Azure Resource Management endpoints and the API uses a similar and consistent model to all other Azure services.

The latest Open API specifications are maintained publicly in Github, with the 2020-05-01 version being the latest stable release of the v3 API.  
[azure-rest-api-specs/specification/mediaservices/resource-manager/Microsoft.Media/stable/2020-05-01 at master · Azure/azure-rest-api-specs (github.com)](https://github.com/Azure/azure-rest-api-specs/tree/master/specification/mediaservices/resource-manager/Microsoft.Media/stable/2020-05-01)

All client SDKs are derived and generated from the Open API specification published on Github. To connect to the v3 APIs, you fist need to update to the latest .NET SDK for the v3 API [here](https://www.nuget.org/packages/Microsoft.Azure.Management.Media/). Version 3.0.2 and higher support the 2020-05-01 version of the API referenced in the Open API above.

To install the latest SDK using PackageManager, use the following command:

Install-Package Microsoft.Azure.Management.Media

To install the latest SDK using the .NET CLI use the following command:

dotnet add package Microsoft.Azure.Management.Media

A full tutorial walkthrough on how to connect to the new v3 .NET API is available here.

[Connect to Azure Media Services v3 API - .NET - Azure Media Services v3 | Microsoft Docs](https://docs.microsoft.com/en-us/azure/media-services/latest/configure-connect-dotnet-howto)

In addition, full .NET samples are available in the following GitHub for various scenarios.

[Azure-Samples/media-services-v3-dotnet: The projects in this repository show how to implement different Azure Media Services scenarios using the v3 version. (github.com)](https://github.com/Azure-Samples/media-services-v3-dotnet)

### Inventory your architecture

Compare what you are currently using and compare it what is now available in V3.

* + 1. Features available in V3
       1. Azure AD integration
          1. Accounts and storage: Encryption at rest
          2. Content protection
          3. Live events

24x7 live streaming on pass-through and encoding live events

Premium 1080p encoding live events

Support for standby mode to help build pools of quicker starting live events at reduced cost.

Live transcription support

Encoding live events will output constant fragment sized outputs, allowing inputs with variable key frame intervals

Inputs of various aspect ratios will not result in stretching in the output

Additional ingest metrics

Updated Portal experience

* + - * 1. Monitoring: Use Azure Monitor

Monitor Live Events

Monitor Streaming Endpoints

Monitor Key Delivery

* + - * 1. Azure Event Grid

Webhooks are no longer supported and have been replaced by Event Grid integration. Customers should move their code from Web hooks to Event Grid for notifications.

* + - * 1. Use ARM templates
  1. Make some decisions about which V3 features you want to use and when.

1. Download the appropriate SDK and follow the How To articles to get started. (Note: The PHP client library is no longer supported in v3)
   1. .NET - [Connect to Media Services v3 API with .NET](https://docs.microsoft.com/en-us/azure/media-services/latest/configure-connect-dotnet-howto)
   2. Java – [Connect to Media Services v3 API with Java](https://docs.microsoft.com/en-us/azure/media-services/latest/configure-connect-java-howto)
   3. Python - [Connect to Azure Media Services v3 API - Python - Azure Media Services v3 | Microsoft Docs](https://docs.microsoft.com/en-us/azure/media-services/latest/configure-connect-python-howto)
   4. Node.js - [Connect to Azure Media Services v3 API - Node.js - Azure Media Services v3 | Microsoft Docs](https://docs.microsoft.com/en-us/azure/media-services/latest/configure-connect-nodejs-howto)
   5. Ruby client
   6. Go client
2. Do you use the REST API? Determine if you can switch from using V2 REST API directly to using the new V3, ARM-based REST API.
   1. [REST API](https://docs.microsoft.com/rest/api/media/) reference pages
   2. The Postman Collection for REST in v3

[Configure Postman for Azure Media Services v3 REST API calls - Azure Media Services v3 | Microsoft Docs](https://docs.microsoft.com/en-us/azure/media-services/latest/media-rest-apis-with-postman)

1. Inventory the events and alerts you are currently using and migrate them to Event Grid.
2. Test
3. Set up a test V3 Media Services account to try the new API.
   1. All Media Services accounts will have access to the v3 API, but it is advised to start development on a fresh account before applying updated code to an existing v2 account. This is because v3 entities are not backwards compatible with v2. Some v2 entities like Assets are forward compatible with V3.
   2. V2 Assets are forward compatible, but V3 Assets and features are not backwards compatible.
4. Model your revised architecture with test accounts. Once you have tested and are happy with your new architecture...
5. Use [Event Grid](file:///C:/Users/inhenkel/Documents/GitHub/azure-docs-pr/articles/media-services/latest/reacting-to-media-services-events.md) if you are using webhooks through NotificationEndpoints for job notifications.
6. Streaming endpoints are forward compatible with the v3 API.

## Encoding Scenarios

#### How to move from v2 Encoding with system presets to v3 Encoding NOTE: Jobs and Tasks created in v2 do not show up in v3 as they are not associated with a Transform. The recommendation is to switch to v3 Transforms and Jobs. There will be a relatively short time of needing to monitor the inflight v2 Jobs during the switchover.

* 1. You now need to create a Transform prior to submitting a Job.
  2. If your v2 code called the Standard Encoder with a preset, you first will create a new Transform with the Standard Encoder preset prior to submitting a Job.
  3. Link to example here – or basic code here.

#### How to submit jobs with inputs that are on HTTP(S) hosted URLs

* 1. You can now submit jobs in v3 from files stored either in Azure storage, or external web servers using the HTTP(S) Job input support.
  2. If you previously used workflows to copy files from Azure blob files into empty Assets before submitting jobs, you may be able to now simplify your workflow by passing a SAS URL for the file in Azure blob storage directly into the Job
  3. << TODO>> Link to sample for HTTP ingest jobs.

#### How to move from v2 Encoding with custom presets to v3 Encoding

* 1. You now need to create a Transform prior to submitting a custom encoding Job.
  2. If your v2 code called the Standard Encoder with a custom preset, you first need to create a new Transform with the custom Standard Encoder preset prior to submitting a Job.
     1. The new custom preset format is defined here
     2. Custom presets are now JSON and no longer XML based. Recreate your preset in JSON following the custom preset schema as defined in the Transform Open API (Swagger) documentation.
     3. Common custom encoding scenarios:
        1. Create a custom Single Bitrate MP4 encode
        2. Create a custom Adaptive Bitrate Encoding Ladder
        3. Creating Sprite Thumbnails
        4. Creating Thumbnails
        5. Sub Clipping
        6. Cropping
  3. Link to example here – or basic code here.

#### How to get the input and output metadata files from an Encoding job

* 1. In v2, XML [input](https://docs.microsoft.com/en-us/azure/media-services/previous/media-services-input-metadata-schema) and [output](https://docs.microsoft.com/en-us/azure/media-services/previous/media-services-output-metadata-schema) metadata files get generated as the result of an encoding job. In v3, the metadata format changed from XML to JSON.
  2. Link to article on metadata in v3

#### How to move from Premium Encoding to v3 Standard Encoder or partner-based solutions

* 1. The v2 API no longer supports the Premium Encoder workflows. Customers that previously used the workflow based Premium Encoder for HEVC encoding should migrate to the new v3 Standard Encoder with HEVC (8-bit) encoding support. 10-bit HEVC encoding support is on the roadmap and will available soon.
  2. Customers requiring the advanced workflow features of the Premium Encoder that we do not support in the Standard Encoder are encouraged to migrate their solutions to an Azure advanced encoding partner solution from Imagine Communications, Telestream or [Bitmovin](https://go.microsoft.com/fwlink/?linkid=2151333&clcid=0x409). Please contact technical support for available partner options if required for your migration.

## Audio Indexing (Indexer v1) and Transcription Scenarios

#### How to move from Indexer v1 audio transcription to the new AudioAnalyzer “basic mode”.

* 1. For customers using the Indexer v1 processor in the v2 API, you need to create a Transform that invokes the new AudioAnalyzer in “basic mode” prior to submitting a Job.
  2. << TODO>> Link to example of using Audio Analyzer in basic mode here – or a basic code sample inline as well.

## Media Reserved Units in the v3 API

* For new v3 accounts created in the Azure Portal, or with the 2020-05-01 version of the v3 API, you no longer are required to set Media Reserved Units (MRUs). The system will now automatically scale up and down based on load.
* If you have a v3 or v2 account that was created prior to the 2020-05-01 version of the API, you can still control the concurrency and performance of your Jobs using Media Reserved Units. For more information, see [Scaling Media Processing](https://docs.microsoft.com/en-us/azure/media-services/previous/media-services-scale-media-processing-overview). You can manage the MRUs using [CLI 2.0 for Media Services v3](https://docs.microsoft.com/en-us/azure/media-services/latest/media-reserved-units-cli-how-to), or using the [Azure portal](https://docs.microsoft.com/en-us/azure/media-services/previous/media-services-portal-scale-media-processing).
* If you do not see the option to manage MRUs in the Azure portal, you are running an account that was created with the 2020-05-01 API or later.

## Live Streaming Scenarios

#### How to move from v2 pass-through Channels to v3 pass-through Live Events NOTE: Channels and Programs created with v2 (which are mapped to Live Events and Live Outputs in v3) cannot be managed with the v3 API. The guidance is to switch over to v3 Live Events and Live Outputs at a convenient time that you can stop your existing v2 Channels. There is currently no support for seamlessly migrating a continuously running (24x7) live Channel to the new v3 Live Event, so maintenance downtime needs to be coordinated at the best convenience for your audience and business.

* 1. Stop and delete all existing v2 Channels and Programs at a convenient maintenance window for your customers and audience
  2. Create a new [v3 Live Event](https://docs.microsoft.com/en-us/azure/media-services/latest/live-streaming-overview)
     1. Sample .NET and REST code here
  3. Use the new [Live Output](https://docs.microsoft.com/en-us/azure/media-services/latest/live-events-outputs-concept) entity instead of Programs
  4. Create new [streaming locators](https://docs.microsoft.com/en-us/azure/media-services/latest/live-streaming-overview) for live streaming of HLS and DASH
  5. Explore the new [Standby](https://docs.microsoft.com/en-us/azure/media-services/latest/live-event-states-billing) features if you require fast-start of live events.
  6. Explore the new [live transcription](https://docs.microsoft.com/en-us/azure/media-services/latest/live-transcription) feature
  7. Explore the new [lower latency](https://docs.microsoft.com/en-us/azure/media-services/latest/live-event-latency) feature
  8. Create [24x7x365 live events](https://docs.microsoft.com/en-us/azure/media-services/latest/live-event-types-comparison) in v3
  9. [Monitor live events](https://docs.microsoft.com/en-us/azure/media-services/latest/media-services-event-schemas" \l "live-event-types) with [Event Grid](https://docs.microsoft.com/en-us/azure/media-services/latest/reacting-to-media-services-events)
  10. Use the new Portal experience

#### How to move from v2 encoding Channels to v3 encoding Live Events

* 1. Stop and delete all existing v2 Channels and Programs
  2. Create a new v3 Live Event for encoding
     1. Sample .NET and REST code here
     2. Show how to enable 1080P and 720P encoding presets
  3. Use the new Live Output entity instead of Programs
  4. How to create streaming locators for live streaming of HLS and DASH
  5. Explore the new Standby features if you require fast-start of live events.
  6. Explore the new live transcription feature
  7. Create 24x7x365 live events in v3
  8. Monitor live events with Event Grid
  9. Use the new Portal experience

## Publishing and Streaming Scenarios

How to publish and stream on-demand content in v3

* 1. Major changes to the way content is published in v3 API
     1. The new publishing model is simplified and uses fewer entities to create a Streaming Locator. The API reduced down to just two entities vs. the previous 4 entities required.
     2. Content Key Policies and Streaming Locators now replace the previous need for ContentKeyAuthoriationPolicy, AssetDeliveryPolicy, ContentKey, and AccessPolicy
  2. Creating Content Key Policies
  3. Creating Streaming Locators
  4. Getting the Streaming paths for use in a DASH or HLS player
  5. Creating Streaming Locators

|  |  |  |
| --- | --- | --- |
| ~~Publish an asset with AES encryption~~ | ~~1. Create ContentKeyAuthorizationPolicyOption 2. Create ContentKeyAuthorizationPolicy 3. Create AssetDeliveryPolicy 4. Create Asset and upload content OR Submit job and use output asset 5. Associate AssetDeliveryPolicy with Asset 6. Create ContentKey 7. Attach ContentKey to Asset 8. Create AccessPolicy 9. Create Locator  [v2 .NET example](https://github.com/Azure-Samples/media-services-dotnet-dynamic-encryption-with-aes/blob/master/DynamicEncryptionWithAES/DynamicEncryptionWithAES/Program.cs" \l "L64)~~ | ~~1. Create Content Key Policy 2. Create Asset 3. Upload content or use Asset as JobOutput 4. Create Streaming Locator  [v3 .NET example](https://github.com/Azure-Samples/media-services-v3-dotnet-tutorials/blob/master/AMSV3Tutorials/EncryptWithAES/Program.cs" \l "L105)~~ |

## Content Protection Scenarios

How to protect content in v3 API

<<TODO>>

Support for Multi-key features in the new v3 API

<<TODO>>

Move from client-side storage encryption for Assets to server-side encryption at REST in v3

<<TODO>>

## Face Redaction Scenarios

How to migrate from Face Redaction in v2 to the new Face Detection and Redaction in v3  
<<TODO>>

## Inventory and Compare architectures

**It would be good to include a V2 architecture diagram to a V3 architecture diagram and discuss new options and benefits.**

## Next steps

[Media Services migration: differences](file:///C:/Users/inhenkel/Documents/GitHub/azure-docs-pr/articles/media-services/latest/migrate-v-2-v-3-differences-description.md)