AMToolkit identifyStream on AWS ECS Implementation Guide

AMToolkit ISEIG v45
rev. 2023.10.31

Introduction

This document shows how to implement the Audible Magic Toolkit `identifyStream` application within AWS Elastic Container Service via a Python wrapper in a Docker container. While the example below showcases the `identifyStream` capabilities in conjunction with streams that are processed by the AWS Interactive Video Service (IVS), `identifyStream` can be used with any standalone media stream.

In this example, when a stream is started within the AWS IVS, it triggers an event that is pushed to an AWS SQS queue, where an AWS ECS task will consume the message and start to run identifications on the stream that was started. Identification responses that are produced by the AWS ECS task will be pushed/saved to an AWS S3 bucket.

Prerequisites

To follow along with this guide, the following items are required:

- Audible Magic Toolkit package for Amazon Linux 2
  - This guide uses the following Toolkit package (Toolkit v45+ will work with this guide): `AudibleMagicToolkit_45.2_AmazonLinux2.tgz`.
- Audible Magic Toolkit configuration file
  - This guide uses the following Toolkit configuration file: `Toolkit.config`
- An AWS account with appropriate permissions
  - AWS region that will be used is `us-east-1`

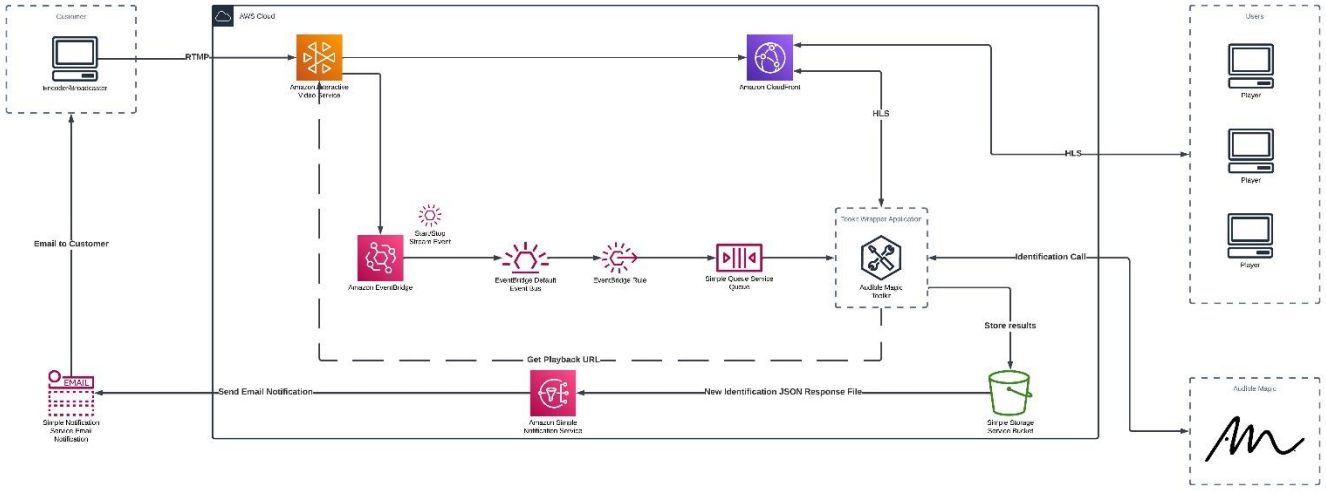
This document assumes familiarity with the following technologies:

- AWS Simple Storage Service (S3)
- AWS Simple Queue Service (SQS)
- AWS Elastic Container Registry (ECR)
- AWS Elastic Container Service (ECS)
- AWS Interactive Video Service (IVS)
- Docker
- Python v3.8

General Setup Steps

Follow the steps to use the application with a Python wrapper in a Docker container:

1. Set up AWS Interactive Video Service (IVS)
2. Create an S3 bucket for output for identification responses
3. Create an SQS queue to receive event notifications
4. Create AWS EventBridge event rule
5. Create the Python wrapper script
6. Create the Toolkit Docker image for use in AWS Elastic Container Service (ECS)
7. Upload the image to AWS Elastic Container Registry (ECR)
8. Create the ECS Task
9. Create the ECS Service
10. Test the AWS ECS Service



A high-level architecture diagram showcasing the integration of the Audible Magic Toolkit within the AWS ecosystem.

1. Set up AWS Interactive Video Service (IVS)

Amazon Interactive Video Service (Amazon IVS) is a managed live streaming service that enables businesses to build interactive experiences and applications to delight their customers with low-latency live video streams.

Refer to the Getting Started with Amazon IVS guide to learn how to create a channel and set up a live stream.

2. Create an AWS S3 Bucket

Create an AWS S3 bucket to hold the identification responses delivered by the Audible Magic toolkit. In this example, we named the AWS S3 bucket “am-stream-identifications”.

```
# Create an AWS S3 Bucket
aws s3api create-bucket --bucket am-stream-identifications --region us-east-1
```

3. Create an AWS SQS Queue to Receive Event Notifications

Create an AWS SQS queue for messages about newly started streams. Your Audible Magic identification service will read these queued messages and begin scanning the streams that have started. In this example, we named our AWS SQS queue “stream-start-queue”. This queue will also be located in the “us-east-1” region.

```
# Create an AWS SQS queue
aws sqs create-queue --queue-name stream-start-queue --region us-east-1 --
attributes
"{\"VisibilityTimeout\":\"30\",\"DelaySeconds\":\"0\",\"ReceiveMessageWaitTimeSeco
nds\":\"20\",\"MessageRetentionPeriod\":\"345600\",\"MaximumMessageSize\":\"262144
\"}"
```

Note: Take note of the AWS SQS queue ARN as it will be needed later

4. Create AWS EventBridge Event Rule

Create an AWS EventBridge event rule to create a message when a new stream starts. This message will get sent to the previously created AWS SQS queue. In this example, we named the event rule “stream-start-event-rule”.

```
# Create an AWS EventBridge event rule
aws events put-rule --name stream-start-event-rule --region us-east-1 --
event-pattern "{\"source\":[\"aws.ivs\"],\"detail-type\":[\"IVS Stream State
Change\"],\"detail\":{\"event_name\":[\"Stream Start\"]}}"
```

More information about AWS EventBridge Event Rules for IVS can be found here:
https://docs.aws.amazon.com/ivs/latest/userguide/eventbridge.html

Attach AWS EventBridge Event Rule to SQS Queue

Now that we have the AWS SQS queue and AWS EventBridge event rule created, we need to link them together, so that the AWS EventBridge knows to send messages to the AWS SQS queue whenever the event rule is triggered.

```
# Attach the event rule to AWS SQS queue as a target
aws events put-targets --rule stream-start-event-rule --region us-east-1 --
targets "Id"="","Arn"=""
```

Note: Be sure to use the ARN of the AWS SQS queue that was created earlier.

Note: The “Id” should be a unique identifier for the newly created stream that can be referenced for troubleshooting, reporting, or other communication about the stream. In this example, we used “12345abcde”.

5. Create Python Wrapper Script

In order to use the Audible Magic Toolkit within your environment, you will need to write an application that interfaces with the `identifyStream` binary, which is the binary within the Audible Magic Toolkit used for live-streamed content This application will also need to be able to do the following:

1. Pull message from AWS SQS queue
    a. Parse message for `arn` of channel
    b. Parse message for `stream_id`
2. Retrieve the stream playback url using `arn` of the channel
3. Launch `identifyStream` binary to start analyzing the stream playback url
4. Monitor for Identification Response files
5. Push Identification Response files to the AWS S3 bucket

Pull Message from AWS SQS Queue

Messages about new streams will be pushed to the AWS SQS queue created earlier. The Toolkit wrapper application will need to pull, parse, and delete these messages while taking the appropriate action on each.

```
import os
import boto3
```

```
STREAM_START_QUEUE_URL = os.environ["STREAM_START_QUEUE_URL"]

sqs = boto3.client('sqs')

def receive_message():
    try:
        response = sqs.receive_message(
            QueueUrl=STREAM_START_QUEUE_URL,
            MaxNumberOfMessages=1
        )
        message = response["Messages"][0]
        receipt_handle = message["ReceiptHandle"]

        channel_arn = message["resources"][0]
        stream_id = message["detail"]["stream_id"]

        print("Stream started for channel: with stream_id: %s: %s", channel_arn,
stream_id)

        sqs.delete_message(
            QueueUrl=STREAM_START_QUEUE_URL,
            ReceiptHandle=receipt_handle
        )
    except ClientError as error:
        print("Couldn't receive messages from queue: %s", queue)
        raise error
    else:
        return (channel_arn, stream_id)
```

Note: The `STREAM_START_QUEUE_URL` will be set as an environment variable when starting the AWS ECS service.

Retrieve Stream Playback URL Using Channel ARN

After pulling the message from the queue, parse the "`channel_arn`" and use that to get the stream playback url.

```
import boto3

client = boto3.client("ivs")

def get_playback_url(channel_arn: str):
    try:
        response = client.get_stream(channelArn=channel_arn)

        playback_url = response["stream"]["playbackUrl"]

        print("Playback URL: %s", playback_url)
    except ClientError as error:
        print("Couldn't get the playback URL for channel arn: %s", channel_arn)
        raise error
    else:
        return playback_url
```

Note: We are using the "`channel_arn`" that we parsed in the message from earlier.

Start Scanning Stream with identifyStream

At this point, we have all the necessary pieces to scan the live stream for copyrighted content using `identifyStream`. We need a function that will run `identifyStream` using the appropriate parameters and place the results in an accessible location.

```
import subprocess
import pathlib
import boto3

# Temporary directory for the Toolkit
tmp_dir = pathlib.Path("/tmp")

# The location of your config file
config_path = pathlib.Path("Toolkit.config")

def run_identifyStream(playback_url: str, stream_id: str):

    # Create local results directory based on the stream_id
    results_directory = pathlib.Path(f'/app/{stream_id}')
    if os.path.exists(results_directory) == False:
        os.mkdir(results_directory)

    cmd = [
        "./identifyStream",
        "-i", str(playback_url),
        "-c", str(config_path),
        "-e", str(stream_id),
        "-O", str(results_directory),
        "-t", str(tmp_dir)]

    cmd_string = " ".join(cmd)

    try:
        print("Executing command: %s",cmd_string)
        subprocess.run(cmd, check=True)
    except Exception as error:
        print("Error running identifyStream")
        raise error
    return
```

Note: Contact Audible Magic for a detailed user guide explaining all mandatory and optional parameters that can be passed to the `identifyStream` binary application.

Monitor for Identification Responses

Since `identifyStream` responds with matches in real-time during a broadcast, it is important to monitor results as they are generated. Identification results are saved locally so you should write a function that will monitor the directory for any new files (results) that are created. In this example, we are saving them in the directory "`/app/{stream_id}`".

```
import os
import time

# Function to return files in a directory
def file_in_directory(my_dir: str):
```

```
    only_files = [f for f in os.listdir(my_dir) if
os.path.isfile(os.path.join(my_dir, f))]
    return only_files

def list_comparison(original_list: list, new_list: list):
    #Note if files get deleted, this will not highlight them
    differences_list = [x for x in new_list if x not in original_list]
    return differences_list

def file_watcher(my_dir: str, poll_time: int):
    while True:
        # Check if this is the first time the function has run
        if "watching" not in locals():
            previous_file_list = file_in_directory(my_dir=my_dir)
            watching = 1
            print("First Time")
            print(previous_file_list)

        time.sleep(poll_time)

        new_file_list = file_in_directory(my_dir=my_dir)
        file_diff = list_comparison(previous_file_list, new_file_list)
        previous_file_list = new_file_list
        if len(file_diff) == 0:
            continue
        push_to_s3(file_diff)
```

Note: The function "`push_to_s3`" will be defined in the next section.

Push Identification Response Files to AWS S3 Bucket

When new files (the identification match results) are detected in the monitored directory, you will want to push these files to the AWS S3 bucket that was created earlier.

```
import os
import boto3

STREAM_IDENTIFICATIONS_BUCKET = os.environ["STREAM_IDENTIFICATIONS_BUCKET"]

def push_to_s3(result_files: list, stream_id: str):
    try:
        for result_file in result_files:
            # The s3 key that the result file will be uploaded to
            upload_key = stream_id + "/" + result_file
            # Upload the result_file to the S3 bucket
            s3_client.upload_file(str(result_file), STREAM_IDENTIFICATIONS_BUCKET,
upload_key)
    except Exception as error:
        print("Error when uploading to S3")
        raise error
    return
```

6. Create Docker Image of the Toolkit Wrapper Application

With the application now written, create a Docker image locally, prior to uploading the Docker image to AWS ECR. To do this, first create a `Dockerfile` with the following contents.

```
FROM public.ecr.aws/docker/library/python:3 as builder

# Install other applications needed for the build
RUN apt install -y gzip tar

# Note: Install dependencies under the WORKDIR alongside
# the function handler, to ensure that the runtime can locate
# them when the function is invoked.
WORKDIR /app

# Install the Toolkit libraries
COPY AudibleMagicToolkit_45.2_AmazonLinux2.tgz .
RUN tar -xzf AudibleMagicToolkit_45.2_AmazonLinux2.tgz
RUN cp -pr AudibleMagicToolkit_45.2_AmazonLinux2/bin/* ./
RUN rm AudibleMagicToolkit_45.2_AmazonLinux2.tgz
RUN rm -fr AudibleMagicToolkit_45.2_AmazonLinux2

# Final stage
FROM public.ecr.aws/docker/library/python:3 as final

# Set the final stage's task directory
WORKDIR /app

# Copy the installed dependencies from the builder
COPY --from=builder /app .

# For build efficiency, isolate these small steps that are most likely to change

# Copy config file
COPY Toolkit.config ./Toolkit.config

# Copy function code
COPY app.py .

# Set the CMD to your handler
# (could also be done as a parameter override outside of the Dockerfile)
CMD [ "app.handler" ]
```

Note: The above `Dockerfile` assumes the same files that are in this guide are being used. If different files are being used, please update the `Dockerfile` to reflect this.

Place the following files into the same directory:

- Toolkit package (`AudibleMagicToolkit_45.2_AmazonLinux2.tgz`)
- `app.py` Python script (which includes all the functionality from the "Create Python Wrapper Script" step)
- Toolkit configuration file (here named `Toolkit.config`)
- `Dockerfile` (here named `Dockerfile`)

With the `Dockerfile` saved, we can run the following Docker command to create the Docker image. In this example, we have named the Docker image "am-stream-identification" with the "latest" Docker tag.

Run the following command to build the Docker image:

```
# Build Docker image
docker build -t am-stream-identification:latest --platform linux/amd64 .
```

Note: Do not forget the trailing dot/period "." at the end of the Docker build command.

7. Upload the Image to AWS Elastic Container Registry

Now that the Docker image is created, create the AWS Elastic Container Registry (ECR) repository for the Docker image to reside.

Note: You need to name the AWS ECR repository the same name that you named your Docker image. In this case, we named our Docker image "am-stream-identification"

Note: You want to create the AWS ECR repository in the same region that you will have your AWS ECS running in. In this example, we are using the "us-east-1" region

Using the AWS CLI, create an AWS ECR repository with the name "am-stream-identification".

```
# Create AWS ECR repository for am-stream-identification
aws ecr create-repository \
--repository-name am-stream-identification \
--region us-east-1
```

Prior to pushing the Docker image to AWS ECR, verify that you are able to connect to the AWS ECR repository via Docker. In this example, we are using the username "AWS".

```
# Ensure you are able to log into AWS ECR with Docker credentials
aws ecr get-login-password --region us-east-1 | docker login --username AWS
--password-stdin example.dkr.ecr.us-east-1.amazonaws.com
```

Next, tag your local Docker image with the appropriate AWS ECR location.

```
# Tag created Docker image with the appropriate AWS ECR location

docker tag am-stream-identification:latest example.dkr.ecr.us-east-
1.amazonaws.com/am-stream-identification:latest
```

After the Docker image has been successfully tagged, you are able to push the Docker image to the AWS ECR repository.

```
# Push Docker image to AWS ECR
docker push example.dkr.ecr.us-east-1.amazonaws.com/am-stream-
identification:latest
```

You can ensure that the Docker image has been successfully pushed to the AWS ECR repository by listing the repositories with the appropriate name. In this example, we used "am-stream-identification".

```
# Ensure that the Docker image has been pushed to the repository
aws ecr list-images --repository-name am-stream-identification
```

8. Create the AWS Elastic Container Service Task

Create an AWS ECS task definition that defines the criteria needed to run the Docker image as a Docker container within AWS ECS. Make sure that the task definition contains information about the AWS SQS queue from which the application will read messages, the AWS S3 Bucket to which the application will save identification results, and the Docker image that the Docker container will use as its base image.

```
# Create AWS ECS task definition
aws ecs register-task-definition \
    --family am-stream-identification-task \
    --container-definitions "[{\"name\":\"am-stream-
identification\",\"image\":\"example.dkr.ecr.us-east-1.amazonaws.com/am-stream-
identification:latest\",\"cpu\":256,\"memoryReservation\":512,\"environment\":[{\"
name\":\"STREAM_START_QUEUE_URL\",\"value\":\"https://sqs.us-east-
1.amazonaws.com/123456789012/stream-start-
queue\"},{\"name\":\"STREAM_IDENTIFICATIONS_BUCKET\""\"stream-
identifications\"}],\"memory\":512,\"essential\":true}]" \
    --requires-compatibilities "FARGATE" \
    --cpu "256" \
    --memory "512" \
    --task-role-arn "arn:aws:iam:123456789012:role/ecsTaskExecutionRole" \
    --execution-role-arn "arn:aws:iam:123456789012:role/ecsTaskExecutionRole"
```

Note: Make sure that the task role and execution role have access to the AWS SQS queue as well as read/write access to the AWS S3 bucket that was created.

9. Create the AWS Elastic Container Service

Now that the AWS ECS task is defined, create an AWS ECS service that will utilize the AWS ECS task. In this example, we named the AWS ECS service “am-stream-identification-service”.

```
# Create AWS ECS service
aws ecs create-service \
    --service-name am-stream-identification-service \
    --launch-type FARGATE \
    --task-definition am-stream-identification-task:1 \
    --desired-count 1
```

Note: With the “`desired-count`” set to “1”, this will automatically start a single instance of the am-stream-identification-task.

10. Adding Email Notification Integration

Without having to check the AWS S3 bucket for new identification JSON response files, you can set up email notifications using AWS Simple Notification Service (SNS) to email you whenever an identification match has been made.

Email notifications are not necessary for Audible Magic identification services or the integration with AWS IVS to function appropriately. If you do not wish to set up email notifications, you may skip to the “Testing the Integration” section.

Create a Simple Notification Service Topic

The first step in setting up email notifications via AWS SNS is to create an AWS SNS topic.

```
# Create an AWS SNS topic
aws sns create-topic \
--region us-east-1 \
--name "am-stream-identification-notifications"
```

Note: you want to use the same region that you have using with your other AWS services thus far.

Note: Take note of the "TopicArn" that is included in the response after you have created the AWS SNS topic, as it will be used in further setup.

Allow S3 to Publish to the AWS SNS Topic

With the AWS SNS topic created, you want to allow your AWS S3 bucket to be able to publish notifications to the AWS SNS topic.

```
# Set the SNS topic to receive notifications from S3 bucket
aws sns set-topic-attributes \
--topic-arn "arn:aws:sns:us-east-1:123456789012:am-stream-identification-
notifications" \
--attribute-name Policy \
--attribute-value '{
    "Version": "2008-10-17",
    "Id": "s3-publish-to-sns",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": { "AWS" : "*" },
            "Action": [ "SNS:Publish" ],
            "Resource": "arn:aws:sns:us-east-1:123456789012:am-stream-
identification-notifications",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:am-stream-identifications"
                }
            }
        }
    ]
}'
```

Note: Be sure to use the AWS SNS topic ARN created in the previous step.

Note: Be sure to use the AWS S3 bucket ARN, otherwise this will not allow for the AWS S3 bucket to be able to push notifications to the AWS SNS topic.

Add Notification Pushing for AWS S3 When Objects have been Created

Since the AWS SNS topic can now receive notifications from the AWS S3 bucket, you need to add the push notification capability to the AWS S3 bucket for whenever a new identification JSON response file is created in the identifications bucket.

```
# Add push AWS SNS notification on ObjectCreated for S3
aws s3api put-bucket-notification \
--region "us-east-1" \
```

```
--bucket "am-stream-identifications" \
--notification-configuration '{
    "TopicConfiguration": {
        "Events": [ "s3:ObjectCreated:*" ],
        "Topic": "arn:aws:sns:us-east-1:123456789012:am-stream-identification-
notifications"
    }
}'
```

Link Email Address to be Sent Notifications

In order to receive an email notification via AWS SNS, you need to link the email address that you want to use to receive the email notification.

```
# Link an email address to receive identification notifications via SNS
aws sns subscribe \
--topic-arn "arn:aws:sns:us-east-1:123456789012:am-stream-identification-
notifications" \
--protocol email \
--notification-endpoint "email@example.com"
```

Now whenever an identification JSON response file has been saved to the AWS S3 bucket, you will receive an email message (formatted in JSON), that shows information about the identification file including the location (key) and size.

The following is an example of a message that would be emailed to you:

```
{
    "Records": [
        {
            "eventVersion": "2.1",
            "eventSource": "aws.s3",
            "awsRegion": "us-east-1",
            "eventTime": "2023-05-23T10:57:55.387Z",
            "eventName": "ObjectCreated:Put",
            // omitted for brevity
            "s3": {
                "bucket": {
                    "name": "am-stream-identifications",
                    "arn": "arn:aws:s3:::am-stream-identifications"
                    // omitted for brevity
                },
                "object": {
                    "key": "st-1DcdEmdn2zkjR1GNFGDjF1W/st-
1DcdEmdn2zkjR1GNFGDjF1W_2023_05_23T10_57_55-0700.json",
                    "size": 5195,
                    // omitted for brevity
                }
            }
        }
    ]
}
```

Note: A confirmation email will be sent to the given email address confirming that you want to receive emails from the AWS SNS topic. You must accept in order to receive the notifications.

11. Test the AWS ECS Service

With the ECS Service fully set up and running, you can test the Audible Magic identification service against a live stream.

Start a stream to your Amazon IVS channel. A message should automatically be sent to the AWS SQS queue that was set up previously.

The Audible Magic identification service will scan the stream as long as it is active. Whenever the service finds a match to copyrighted content that has been registered by rights holders to Audible Magic's databases, a JSON response file with the match results will be produced and saved to the AWS S3 bucket that was created earlier in this guide, within a folder named after the `stream_id` of the active stream.

Note: If you had set up to receive email notifications via SNS, then you should have received an email with information about the identification JSON response file, including the location (key) of the file.

The following example shows response files for three distinct streams:

```
/
|
├──20230523/
|     |
|     └──st-1DcdEmdn2zkjR1GNFGDjF1W
|           |
|           ├──st-1DcdEmdn2zkjR1GNFGDjF1W_2023_05_23T10_57_55-0700.json
|           |
|           └──...
|
├──20230524/
|     |
|     └──st-1AWAZs98VP1HDpzhGQwr3Kp
|           |
|           ├──st-1AWAZs98VP1HDpzhGQwr3Kp_2023_05_24T12_01_45-0700.json
|           |
|           ├──st-1AWAZs98VP1HDpzhGQwr3Kp_2023_05_24T12_01_45-0700.json
|           |
|           └──...
|
└──20230525/
      |
      └──st-1HZbbn0NKqhGYvjBc671UnY
            |
            ├──st-1HZbbn0NKqhGYvjBc671UnY_2023_05_25T09_34_41-0700.json
            |
            └──...
```

The structure of the identification JSON response file is as follows:

```
{
    "statusCode": 2006,
    "statusDescription": "Status 2006 (2006) Match found",
    "unknownAssetID": "8349-4b4589262a1b",
    "matches":
    [
        {
            "ruleCode": "block",
            "matchType": "audio",
            "amItemID": "4514a672110f4a4eb96e507b33c592fd_RGUa00",
            "matchTimeInUnknown": 1659370072.793,
            "matchDurationInUnknown": 20.000,
            "matchTimeInReference": 0.500,
            "metadata":
             {
                "ContentCategory": "Music",
```

```
                "Title": "Song Title",
                "Artist": "Song Artist",
                "ISRC": "USZZ00000001",
                "Label": "Song Label",
                "Vendor": "Song Vendor",
                "AlbumReleaseDate": "01-Jan-1900",
                "AlbumUPC": "012345678900",
                "SongLengthInSeconds": "195"
            }
        }
    ]
}
```


The above identification response indicates that a single identification match was made and includes the associated metadata.

Note: The field "`matchTimeInUnknown`" reflects the start of the matching identification within the stream in Unix epoch time (number of seconds since 1 January 1970).

Note: If you had set up to receive email notifications via AWS SNS, then you should have received an email with information about the notification JSON response file, including the location (key) of the file.

Getting Help

If you require assistance, the best way to contact our support team is to submit a helpdesk ticket in one of the following ways:

- Use the support portal at https://support.audiblemagic.com.
- Send an email to helpdesk@audiblemagic.com.

AM Client Services handles general requests during regular business hours, Monday to Friday 9:00 AM to 5:00 PM Pacific Time.