Implementation Guide

Secure Media Delivery at the Edge on AWS

Secure Media Delivery at the Edge on AWS: Implementation Guide

Table of Contents

Deploy a solution to protect your premium video content from unauthorized access when delivered through Amazon CloudFront

Publication date: *August 2022 (last update: November 2024*)

Premium video content is one of the most valuable assets for media and entertainment companies. Video delivery teams must continue to raise the security bar to ensure that only authorized viewers consume the content over approved delivery channels. For a video streaming distribution of any scale, customers seek a complete, incremental solution that works universally on a variety of video clients without requiring a re-architecture of their workloads.

The Secure Media Delivery at the Edge on AWS solution integrates with Amazon CloudFront to offer a ready-to-use content protection mechanism that allows you to meet licensing obligations from the right holders by improving anti-piracy controls. Video Streaming Engineers and Content Delivery Network (CDN) operators can easily deploy the solution into their environment and incorporate it with a minimal number of steps without needing to rearchitect their video services.

This solution leverages CloudFront Functions to introduce a cookie-less approach that simplifies and automates the process of access token management for media streaming services. By using serverless resources based on a new edge serverless environment, customers can generate an encrypted token, inject it into the media delivery path, and validate the token for every request, without needing to produce and attach the token for the same playback session. The token authorization function at the edge can be associated with specific CloudFront path behavior, pointing to the media origin with original content. Shifting this functionality to the edge simplifies customers' secure video streaming workflows by making it transparent for existing video origins, removing the complexity of manipulating media manifest files.

This implementation guide provides an overview of the Secure Media Delivery at the Edge on AWS solution, its reference architecture and components, considerations for planning the deployment, configuration steps for deploying the solution to the Amazon Web Services (AWS) Cloud.

The intended audience for using this solution's features and capabilities in their environment includes solution architects, DevOps engineers, data scientists, and cloud professionals.

Use this navigation table to quickly find answers to these questions:

If you want to . . .	Read . . .
Know the cost for running this solution. The estimated cost for running this solution in the US East (N. Virginia) Region is **USD $25.65** for the base module **per month** for AWS resources.	Cost
Understand the security considerations for this solution.	Security
Know how to plan for quotas for this solution.	Quotas
Know which AWS Regions support this solution.	Supported AWS Regions
Know which video streaming formats the solution supports.	Supported formats
Know the requirements for using an existing CloudFront distribution.	CloudFront prerequisites
View or download the AWS CloudForm ation template included in this solution to automatically deploy the infrastructure resources (the “stack”) for this solution.	AWS CloudFormation template
Access the source code and optionally use the AWS Cloud Development Kit (AWS CDK) to deploy the solution.	GitHub repository

Features and benefits

The solution provides the following features:

Ease of integration

Easily integrate this solution into your existing workflows or add to new ones in a few configuration steps. Implemented as an incremental component, the solution is ready to use without redesigning the CloudFront architecture.

Live and on demand workloads

The solution supports Live streaming and Video on Demand (VOD) workloads.

Widespread support across video clients

With a wide range of devices and streaming formats, the solution is designed to provide the best possible support coverage. The URL-based token works universally with the clients you use today, and the ones you might need to support tomorrow.

Flexible token structure

Presenting secure tokens in the widely-adopted JSON Web Token (JWT) format offers flexibility in construction. Combine multiple viewer attributes and geolocation details provided by CloudFront to restrict playback to only authorized clients. Viewer attributes are not exposed in the token or URL path, ensuring the privacy of your end-users.

Session revocation

Quickly identify playback sessions with irregular traffic patterns suggesting unauthorized distribution of your content. Block playback sessions by reporting corresponding session identifiers, or leverage the automatic workflow offered by the solution to detect and block suspicious sessions.

Scale and automation

The solution seamlessly scales to the highest traffic events via CloudFront Functions. You can depend on the automated workflows implemented by the solution to handle regular key rotation, and process traffic patterns to detect and block sessions with suspicious traffic patterns.

Integration with Service Catalog AppRegistry and Application Manager, a capability of AWS Systems Manager

This solution includes a Service Catalog AppRegistry resource to register the solution's CloudFormation template and its underlying resources as an application in both Service Catalog AppRegistry and Application Manager. With this integration, you can centrally manage the solution's resources and enable application search, reporting, and management actions.

Use cases

Secure Content Delivery

Building and deploying infrastructure or applications at the edge requires comprehensive security with a reliable cloud infrastructure, and securing your rich media or static content requires extra care. With the Secure Media Delivery at the Edge on AWS solution, you can add additional layers of security to prevent unauthorized access and common web exploits. This allows you to spend more time building applications, and less time monitoring threats. Often, it is also a contractual obligation content distributors need to adhere to with respect to security and access control methods. This solution can be used in combination with digital rights management (DRM) systems or used as a single protection from unauthorized playback.

Streaming Media

As consumer demand for video streaming increases, media and entertainment companies are looking for secure and reliable web-based video streaming alternatives to traditional television. Using this solution, you can avoid inefficient trial-and-error approaches and save on time and costs for your Video on Demand (VOD) and Live streaming media projects. This solution serves customers looking for a robust mechanism with widespread support across variety of clients, as well as more flexibility in adjusting the working parameters (for example, fine-grained geo restrictions, custom headers, source IPs) and logic of securing their video streams.

Concepts and definitions

This section describes key concepts and defines terminology specific to this solution:

application

A logical group of AWS resources that you want to operate as a unit.

Access control list (ACL)

A web ACL gives you fine-grained control over all of the HTTP(S) web requests that your protected resource responds to.

Common Media Application Format (CMAF)

An HTTP-based streaming and packaging standard to improve delivery of media over the internet, compatible with HLS and DASH, and co-developed by Apple and Microsoft.

Digital rights management (DRM)

A technology used to control and manage access to copyrighted material.

Dynamic Adaptive Streaming over HTTP (DASH)

An HTTP-based streaming protocol (also known as MPEG-DASH) to deliver media over the internet and developed under MPEG (Motion Picture Experts Group).

HTTP Live Streaming (HLS)

An HTTP-based streaming protocol to deliver media over the internet and developed by Apple Inc.

WCU

AWS WAF Capacity Units (WCU) are used to calculate and control the operating resources that are required to run your rules, rule groups, and web ACLs. AWS WAF enforces WCU limits when you configure your rule groups and web ACLs. WCUs don't affect how AWS WAF inspects web traffic.

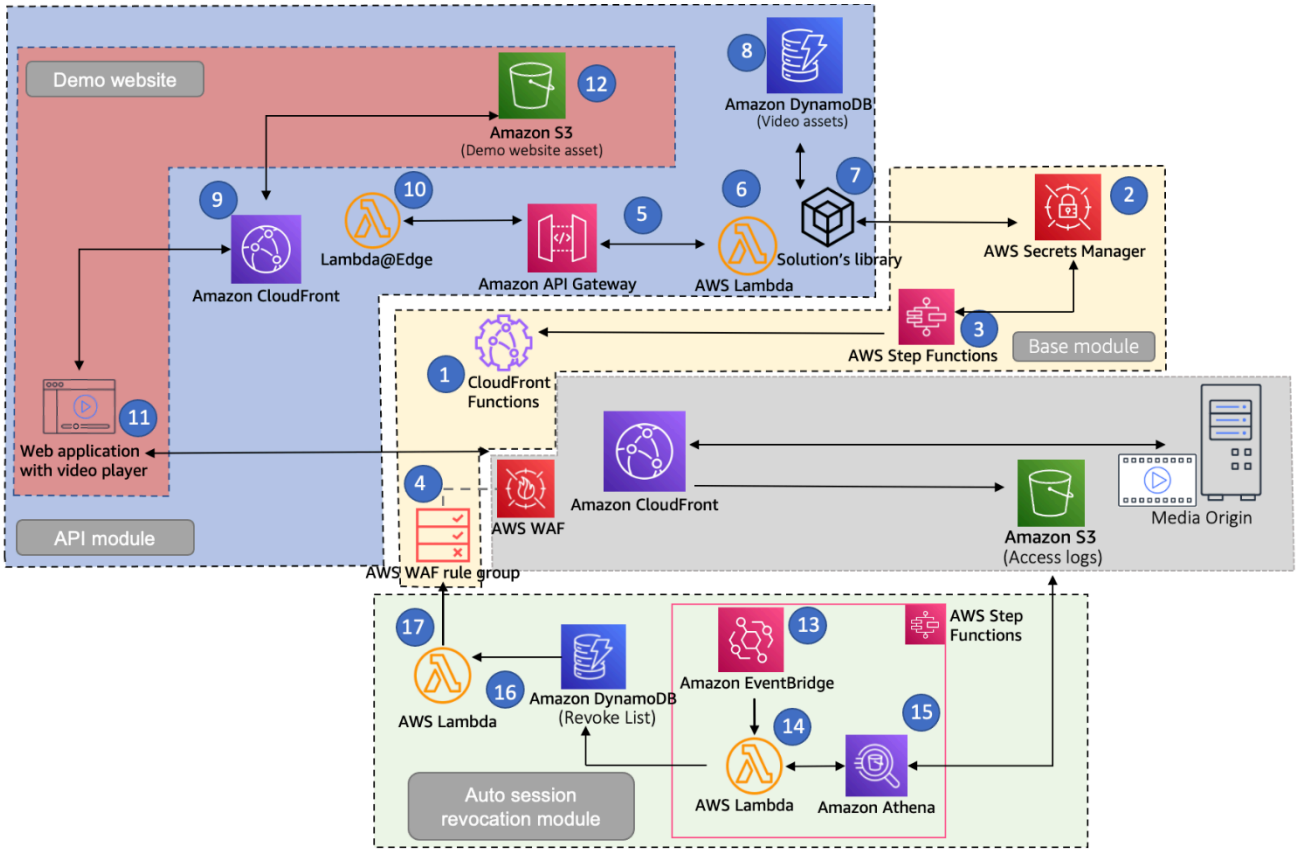
For a general reference of AWS terms, see the AWS Glossary.

Architecture overview

This section provides a reference implementation architecture diagram for the components deployed with this solution.

Architecture diagram

Deploying the Secure Media Delivery at the Edge on AWS solution in an existing environment with Amazon CloudFront and Media Origin service creates a number of resources. These resources play different roles and can be grouped into three functional modules as shown in the following reference architecture diagram.



Secure Media Delivery at the Edge on AWS architecture

Base module

1. An Amazon CloudFront Function that validates secure tokens, permitting or denying access to video content.

2. An AWS Secrets Manager stores secrets holding signing keys for generating and validating viewers' tokens.
3. An AWS Step Functions workflow that coordinates key rotation process.
4. An AWS WAF rule group containing the list of playback sessions that should be blocked as they get identified as compromised.
5. An Amazon API Gateway public API used to process requests to generate the tokens for video playback, and to manually revoke specified playback sessions.
6. An AWS Lambda function associated with API Gateway that generates the token for video playback based on the retrieved metadata about the video assets and token parameters.
7. A solution-provided library that provides the necessary methods to generate the tokens, imported into the AWS Lambda Function.

API Module

8. An Amazon DynamoDB table to store metadata about video assets and corresponding parameters used to generate the tokens.
9. An Amazon CloudFront distribution to deliver the traffic from API Gateway and deliver demo website when activated.
10. A Lambda@Edge function that signs outgoing requests towards API Gateway according to SigV4 specification.
11. A demo website (when activated) with video player embedded in it.
12. An Amazon Simple Storage Service (Amazon S3) bucket that stores static assets for the demo website.

Auto session revocation module

13. An Amazon EventBridge rule that runs periodically to invoke session revocation workflow in AWS Step Functions.
14. Lambda functions invoked in Step Functions workflow that produce SQL query submitted to Amazon Athena, to obtain the results from Amazon Athena, and push moving them forward in the processing pipeline.
15. Amazon Athena running SQL queries against CloudFront access logs to list the suspicious video playback session ids with abnormal traffic characteristics.

16An Amazon DynamoDB table revocation list to store session IDs that have been submitted to be revoked with additional information.

17A Lambda function which compiles a final list of the playback sessions marked to be blocked and updates AWS WAF rule group with the appropriate rules matching selected sessions.

> **Note**
>
> AWS CloudFormation resources are created from AWS Cloud Development Kit (AWS CDK) (AWS CDK) constructs.

Solution workflow

This solution provides code constructs (as a NodeJS library) that makes it simpler to generate a token upon verifying the viewer's permissions when playback URL is requested. Integrating token generation into authorization flow in your application is a key element in the effectiveness of the solution.

> **Important**
>
> The solution-created demo website module creates a simple website, which is meant to provide a proof-of-concept and troubleshooting phases for testing the solution. However, you should eventually replace it with an end-user application and viewer authorization components.

One of the primary objectives of this solution is to provide a uniformly supported mechanism across a variety of client types. The main consideration in that regard is to use a universally supported method to carry the tokens. In this solution, the generated token is embedded in the playback URL to ensure a video player does not drop the token, which would be possible if other methods for carrying the token were used.

Acquiring playback URL with the token

1. When the viewer wishes to play a specific video content, client-side application makes an API call to `Playback` API that is expected to respond with the playback URL including the token.

2. When a request reaches `Playback` API, the request is validated to ensure if the viewer has permissions to watch requested video content (as outlined above, this step needs to be added by users of the solution).
3. Once it is verified that the client is authorized to watch the requested content, an internal call is initiated to construct a playback URL containing the token for that specific viewer. In the solution's implementation, this call is served by API Gateway through Lambda function associated with the `/tokengenerate` path.
4. The service responsible for token generation retrieves information about the requested video asset, and token parameters, to issue a playback URL including the token for requesting viewer. As specified by token policy, token includes certain attributes that are associated to the viewer (user-agent, source IP, etc.) and restrict access only to the requested video asset. If specified in the token policy, provided library will also generate a random playback session ID that will be signed.

The Playback API serves back the playback URL for requested asset, including the token with the session ID.

Client requesting video assets and token validation

1. After the client-side application acquires the playback URL with the token, it is passed on to the video player, which initiates sequential requests for video manifest and video segments.
2. As the request reaches CloudFront distribution, it is first run through attached AWS WAF web ACL (if included in the CloudFront distribution configuration), which includes the rule group with rules to match the sessions that were previously submitted to be blocked.
3. If the session ID corresponding to the request was not blocked by AWS WAF, Amazon CloudFront Function inspects the token retrieved from the URL path. If all signatures match, the token is unexpired, and the token covers requested video asset, the request is allowed and the token is removed from the URL path. After that, the request continues on the normal Amazon CloudFront processing path.

Revoking compromised playback session

Manual path

1. After a solution operator discovers a compromised playback session that should be blocked, an API call can be made to revoke such a session, and start blocking any subsequent requests linked to that session.

2. Amazon API Gateway verifies if the request is coming from an authorized user, and initiates session revocation methods exposed through provided solution library.
3. Session revocation method pushes the session ID to the target DynamoDB table which in turn invokes a Lambda function associated with a DynamoDB data stream.
4. The invoked Lambda function goes through the list of the sessions in the DynamoDB table, prioritizes them, and reduces the list to the size that can be accommodated within the AWS WAF rule group. Eventually, Lambda function updates WAF rule group with the updated list of sessions that must be revoked.

Automatic session revocation path

1. As the traffic flows through the CloudFront distribution, access logs are continuously emitted to an Amazon S3 bucket (logs can be further partitioned as per Analyze your Amazon CloudFront access logs at scale).
2. At a periodic frequency set by the user, an AWS EventBridge rule invokes a Step Functions workflow. The task of that workflow is to analyze the logs in the S3 bucket holding access logs to identify the sessions with unusual traffic composition that suggest the sessions have been compromised (meaning more than one viewer uses the same playback session ID to retrieve the stream).
3. A Lambda function is invoked through the Step Functions workflow to generate a SQL query that incorporates input parameters set by the solution's client and limits the time window for the logs subject to analysis.
4. A SQL query is submitted to Amazon Athena, which performs the job and outputs a list of session IDs that exceed suspicious threshold.
5. Another Lambda function in the Step Functions workflow checks if the job is complete, and pulls the output result for further processing. Athena sourced session IDs are populated into the DynamoDB table that holds the list of sessions that must be revoked.
6. Changes to the DynamoDB table initiate DynamoDB streams that invokes the associated Lambda function as a result.
7. Invoked Lambda function goes through the list of the sessions in DynamoDB table, prioritize them according to the predefined criteria, and reduce the list to the size that can be accommodated within AWS WAF rule group. Eventually, the Lambda function updates WAF rule group with the updated list of sessions that must be revoked.

Key rotation

1. An EventBridge rule is initiated periodically at a time set by the solution's client when the solution is deployed.
2. After initial deployment of the solution and every time EventBridge rule runs, a key rotation Step Functions workflow is started.
3. A Lambda function called by Step Functions workflow generates a random signing key which is used to sign the token through library methods, and validates the token by CloudFront Function logic.
4. Before the newly generated key replaces the existing key stored as the primary secret in AWS Secrets Manager**,** that new key is first updated in CloudFront Function code, and the new code is published.
5. Step Functions workflows check whether that change was successfully updated and propagated through CloudFront Functions. Once confirmed, workflows progress to the next step.
6. The newly created key is promoted as the primary key and replaces the key value of primary secret in Secrets Manager.
7. At this point, when the library method generating token retrieves keys from Secrets Manager it uses a new primary key, which is also available in CloudFront Functions.

AWS Well-Architected design considerations

This solution uses the best practices from the AWS Well-Architected Framework, which helps customers design and operate reliable, secure, efficient, and cost-effective workloads in the cloud.

This section describes how the design principles and best practices of the Well-Architected Framework benefit this solution.

Operational excellence

This section describes how we architected this solution using the principles and best practices of the operational excellence pillar.

Secure Media Delivery at the Edge on AWS automatically creates a CloudWatch dashboard to monitor metrics and indicate if specific components in the solution operate as expected, and if there are any anomalies that must be investigated.

Security

This section describes how we architected this solution using the principles and best practices of the security pillar.

This solution is specifically designed to protect your premium video content from unauthorized access when delivered through Amazon CloudFront. It creates IAM roles associated with resources that need to perform specific actions. The permissions defined in the policies created in the solution align with the principle of least privilege access, granting just those permissions that a specific component needs to fulfil its tasks fully.

Reliability

This section describes how we architected this solution using the principles and best practices of the reliability pillar.

Secure Media Delivery at the Edge on AWS uses AWS serverless services wherever possible, (Lambda, API Gateway, Amazon S3 and DynamoDB) to ensure high availability and quick recovery from service failure.

Performance efficiency

This section describes how we architected this solution using the principles and best practices of the performance efficiency pillar.

Secure Media Delivery at the Edge on AWS uses serverless architecture throughout the solution, and it can be launched in any AWS Region of your choice in which regional resources will be created (Secrets Manager secrets, Step Functions workflows, Lambda functions, and Dynamo DB tables).

This solution is automatically tested and reviewed by solutions architects and subject matter experts for areas to experiment and improve.

Cost optimization

This section describes how we architected this solution using the principles and best practices of the cost optimization pillar.

The cost for running the solution varies based on a number of factors, including the duration of the streaming events and the number of concurrent viewers. We recommend creating a budget through AWS Cost Explorer to help manage costs, and customers can measure the efficiency of the workloads, and the costs associated with delivery, by using Application Manager.

Sustainability

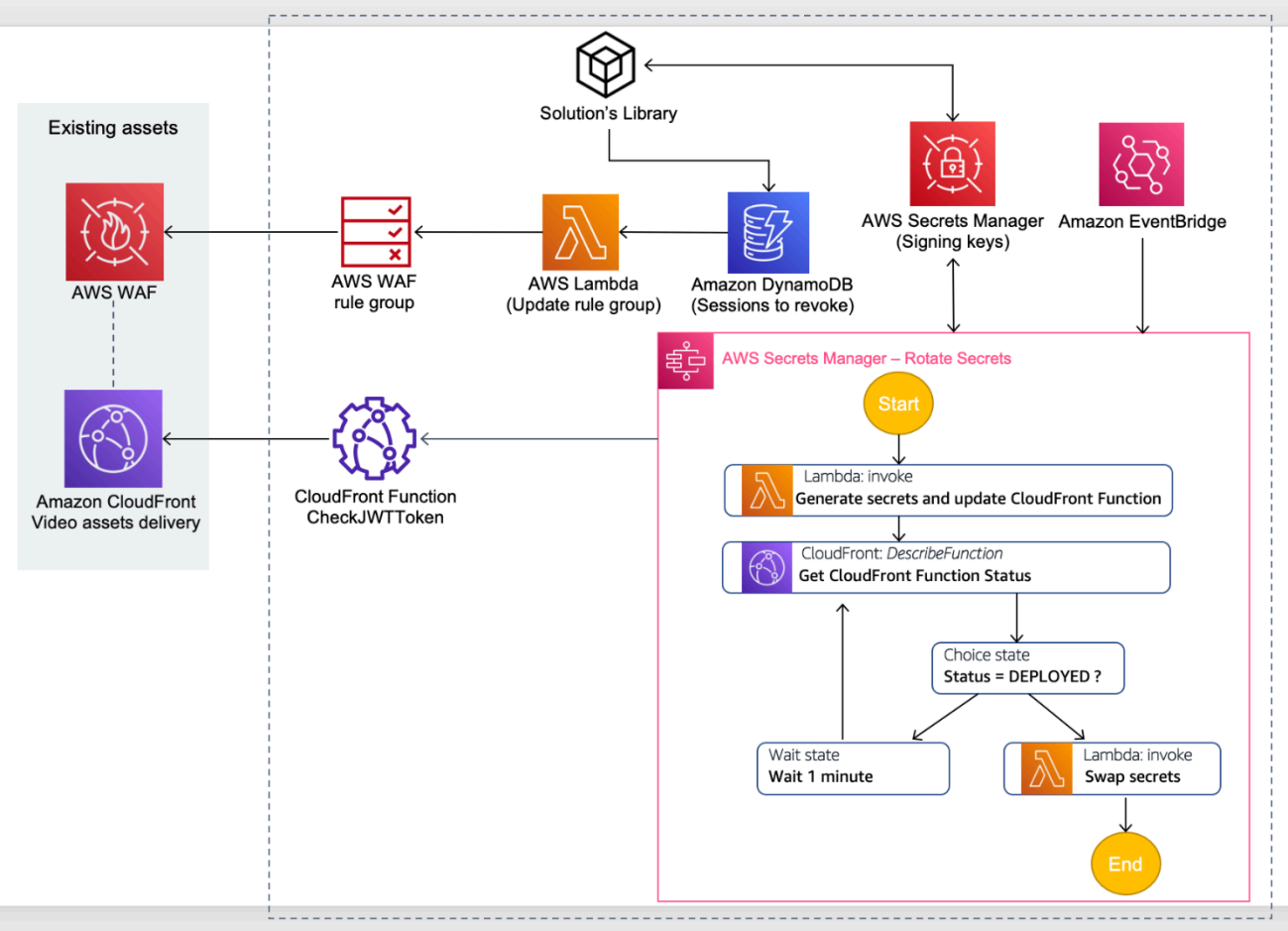
This section describes how we architected this solution using the principles and best practices of the sustainability pillar.

Secure Media Delivery at the Edge on AWS uses managed and serverless services to minimize the environmental impact of the backend services. Customers can run this solution only during the duration of the event and delete the stack after the program ends, reducing the carbon footprint compared to the footprint of continually operating on-premises servers.

Architecture details

This section describes the components and AWS services that make up this solution and the architecture details on how these components work together.

Base module



Secure Media Delivery at the Edge on AWS: Base module

The base module includes solution components that are central to the solution, while the rest of the modules further expands on it. From a functional standpoint, it encompasses some of the key elements for providing the most fundamental outcomes for this solution, which are:

- A validator function implemented as a CloudFront Function code that you attach to your CloudFront distribution. This function inspects requests' attached token and validates conditions.
- AWS Secrets Manager stores the secrets holding signing keys that are used when the token is generated and validated. This part of the solution does not require any user maintenance when

keys are created, updated new keys are distributed to CloudFront Function validator, and when keys are supplied to the service generating tokens through the provided solution library.

- AWS WAF rule group, Lambda function, and DynamoDB connect together to create a session revocation pipeline. The session revocation pipeline allows blocking playback sessions that are determined as compromised. The details of the compromised sessions can be either sent manually through the method that can be found in the solution library, or in an automated way by the means of automatic session revocation. Automatic session revocation automatically runs a log processing pipeline to detect suspicious sessions and push them towards that AWS WAF rule group.

The `Validator` CloudFront function is the same, static code which doesn't change across implementations and use cases. As such, it can be associated with multiple CloudFront distributions delivering different types of video streaming content, as long as it is acceptable to use the same set of signing keys in all cases. Function logic takes the token (subject to validation) from the beginning of the URL path and removes it, before moving the request forward upon successfully token validation. Because CloudFront Function is attached to viewer request trigger, using an individual token does not affect the cache hit rate, as the function code removes the token from the URL path before the cache key is computed. This means that viewers requesting the same video object but using different, individual tokens, will still share the same object from the cache. With regards to scalability, CloudFront Functions have been designed to follow CloudFront scale, and no additional considerations are required in terms of adjusting the service limits for this component. When token validation process is invoked, the input string retrieved from the URL path for further processing has the following format:

```
[SessionID].JWTHeader.JWTPayload.JWTSignagure
```

SessionId parameter is optional. You can generate your token without Session ID associated with it. During token generation step, if you decide to omit the Session ID, it will not be present in the URL path, and only JSON Web Token (JWT) elements will remain. In both cases, there is no need to modify CloudFront Functions as it handles the process of recognizing if Session ID is present.

The process of validating and processing the token can be broadly broken into three discrete stages:

- Verifying JWT signature to validate token integrity. The signing key used here is derived from the unique key identifier that is set in the JWT header UUID. From that UUID, CloudFront Functions determines the corresponding key value.

- Checking access conditions as per claims included in the token. These conditions can be categorized as follows:
  - Time-bounding conditions: to verify if the token is within its validity period
  - Asset attributes conditions: to check if requested object matches with the content the token was issued for
  - Viewer attributes conditions: tokens can be generated for specific viewer attributes for example, source IP, user-agent headers, session ID etc., to improve the uniqueness of a token. In this category, additional signature is calculated from the viewer's attributes available in the function invocation context, and compared with the signature included in the token calculated at playback API stage when token was vended to the viewer.

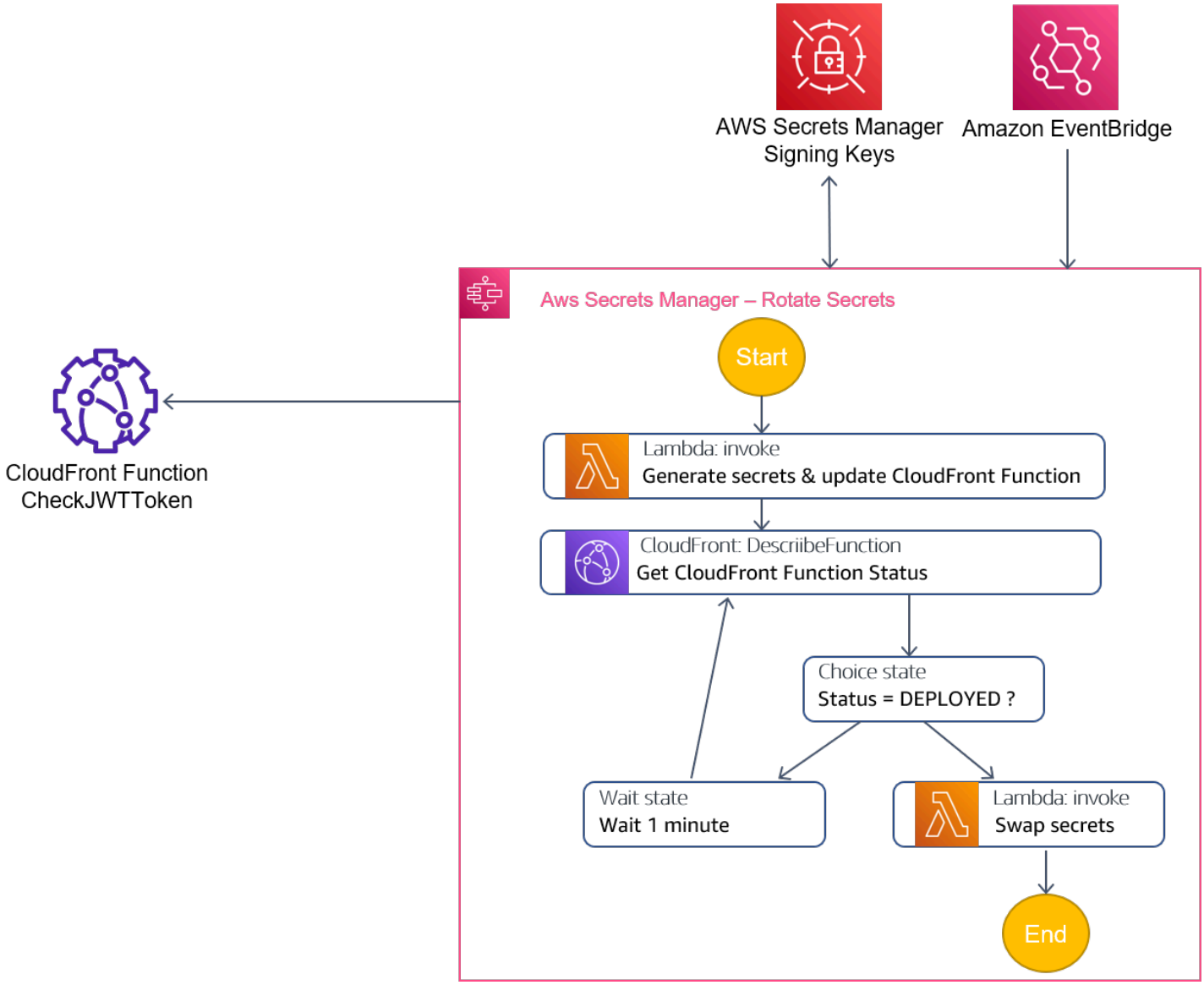
- Token acceptance decision – if token verification failed in any of the previous stages, the request is terminated by the function and a 403 (Forbidden) status code is returned back to the viewer. If the request satisfies all of the access conditions and validation steps, it is permitted. After removing the token from the URL path, the request continues the normal processing path on CloudFront.

It is important to note that the viewer attribute conditions checks, on viewer's request, are based on metadata included in the event object available in CloudFront Functions. While some of the viewer request attributes are always available and present in that object, for example, viewer's IP, headers, and query string parameters, some of the viewer specific attributes require additional configuration steps. Those attributes are location-specific attributes that can be included in the token, for example, viewer's country and region. This information is conveyed through specific CloudFront generated headers: CloudFront-Viewer-Country, CloudFront-Viewer-Country-Region, which are only generated and made available in the event object, when each of these headers is listed either in cache or origin request policies. Configuring these headers in the cache policy split the object in the cache per country the traffic is coming from, and effectively decrease the cache hit ratio. Therefore, we recommend including these headers in the origin request policy as additional headers.

Key rotation workflow

Managing the signing keys used at both token generation and validation steps is a crucial token-based access control mechanism. Any change to the keys, for instance during the key rotation step, must be carefully coordinated to avoid a transient state of inconsistency. In this state, playback API

and CloudFront Functions become out-of-sync, where one of these components uses a signing key that the other does not recognize. To prevent that from happening, as part of the base module an automated key rotation pipeline is built. This pipeline is initiated the first time after the base module stack is deployed, and after that periodically according to the key rotation setting specified when launching the solution. The configuration about when the key rotation process must be initiated is saved as an EventBridge rule. Any time that workflow is initiated, the subsequent steps are controlled via AWS Step Functions workflow as depicted in the following diagram.



Key rotation workflow

A Lambda function automatically generates a new signing key with a corresponding unique UUID value, and stores it as a temporary secret in Secrets Manager. Lambda function also makes an API call to update CloudFront Function with the newly created key and UUID. Previously used key will

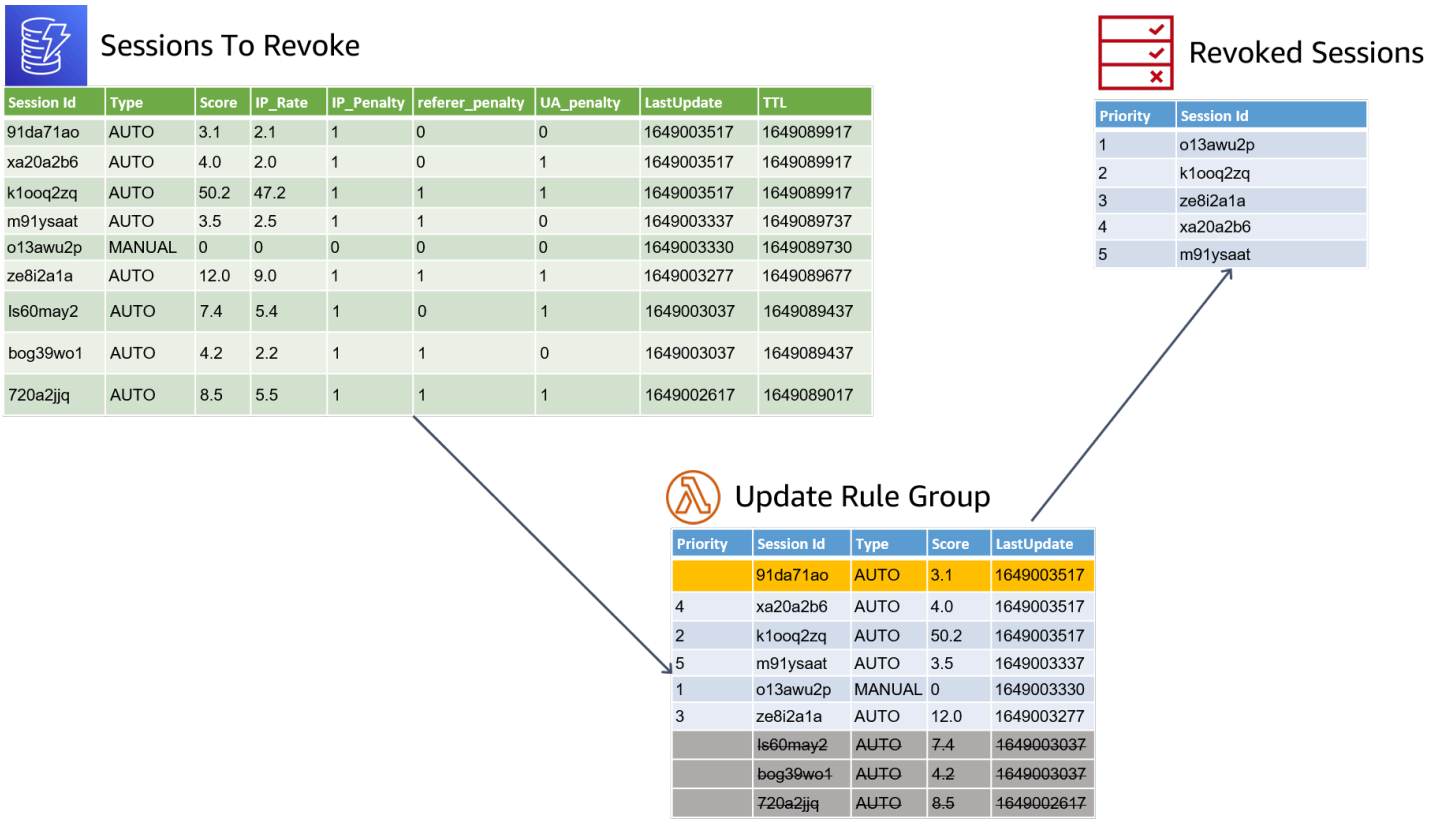
still be available at function runtime, so that CloudFront Function code is able to validate tokens signed both by the old and new key during the transition period when the key is rotated.

The next step in the Step Functions workflow is to check every minute if the new function code, including the new key, has been deployed and that it can be used to generate new keys going forward. Once confirmed, the last Lambda function in the workflow will effectively swap the keys in Secrets Manager by moving the UUID and key value stored in the primary secret to the secondary secret, and replace the primary secret with the keys that have been created and deployed in CloudFront Functions. The next time the primary secret is retrieved by the solution's library method, a new key is used to generate the tokens. With this staged and controlled process, at any time CloudFront Functions is able to validate all incoming requests with the tokens. After new keys are deployed in the function code, there will still be an interim period before rest of Step Functions workflow updates Secrets Manager with the new key, during which time the `Playback` API service relies on the older key. Because the old key and its UUID information is still present in the function code, when a token is generated using an older key by Playback API Gateway API, CloudFront Function can recognize the UUID of the used key specified in the JWT header. When the new key is eventually fetched by `Playback` API, it will be already confirmed that the new key has also been published on CloudFront Functions side, therefore both components can rely on the new key safely.

Base module: Session revocation workflow

The base module also comes with a series of components linked together that allows you to block any ongoing playback session based on the optional **sessionId** attribute that can be associated with the token at the time of token generation. To make use of session revocation, your playback API must include session IDs while creating the token and appended as specified before. There are two paths of emitting sessions IDs that were evaluated to be blocked - manual and automatic. Manual path provides you the flexibility in implementing custom mechanism of detecting compromised sessions while the solution provides you with an interface (in the form of library method or API endpoint) to submit the sessions you identified for blocking. For example, employing A/B watermarking type of traffic mapping and backend analysis to associate leaked stream with the session ID. To push a session ID identified in such a way, you can use a dedicated method defined in secret class, which will push that session into a dedicated DynamoDB table for this purpose, associated with the solution stack. In the automatic approach, to detect and inform about compromised sessions, an automated process is invoked at regular intervals to inspect anomalies in traffic pattern and its composition for each session, and capture the ones that deviate notably from the established norm.

A DynamoDB table holds the list of sessions to be processed to compile the final list of sessions that will be eventually blocked. Session blocking happens at the level of AWS WAF within the scope of the rule group created during solution deployment. You must specify the capacity of that rule group expressed in WAF capacity unit (WCU) only once before it's created, therefore the capacity of that rule group is immutable, which will determine the upper limit of sessions that can be blocked at any time. Rules enclosed within that rule group are string matching rules, one for each blocked session, that check for a matching session ID at the beginning of the request URL path. Role of intermediary between the WAF Rule Group is assumed by a Lambda function which is initiated through DynamoDB data streams every time a change is made to the table. In addition to formulating the WAF rule group and pushing updates, a logic that orders session from the source DynamoDB table to limit the number of output sessions is implemented in a Lambda code to accommodate resulting list within the WCU limit. Refer to the following figure for a description of the logic behind filtering, ordering, and building the output list.



Base module: Session revocation workflow

With every change that occurs in DynamoDB, for example, adding a new session designated to be blocked, a Lambda function is invoked and all entries in the DynamoDB table are evaluated.

In a first step, older session IDs are filtered out before the next step to eliminate the sessions that went past retention period. Retention period is a setting defined when a stack is launched, specifying for how long a session should be kept on the blocking list when WCU limit prevents including all the sessions. After filtering the sessions past their retention timestamp, Lambda function will look into more parameters corresponding to each session to order them accordingly as follows:

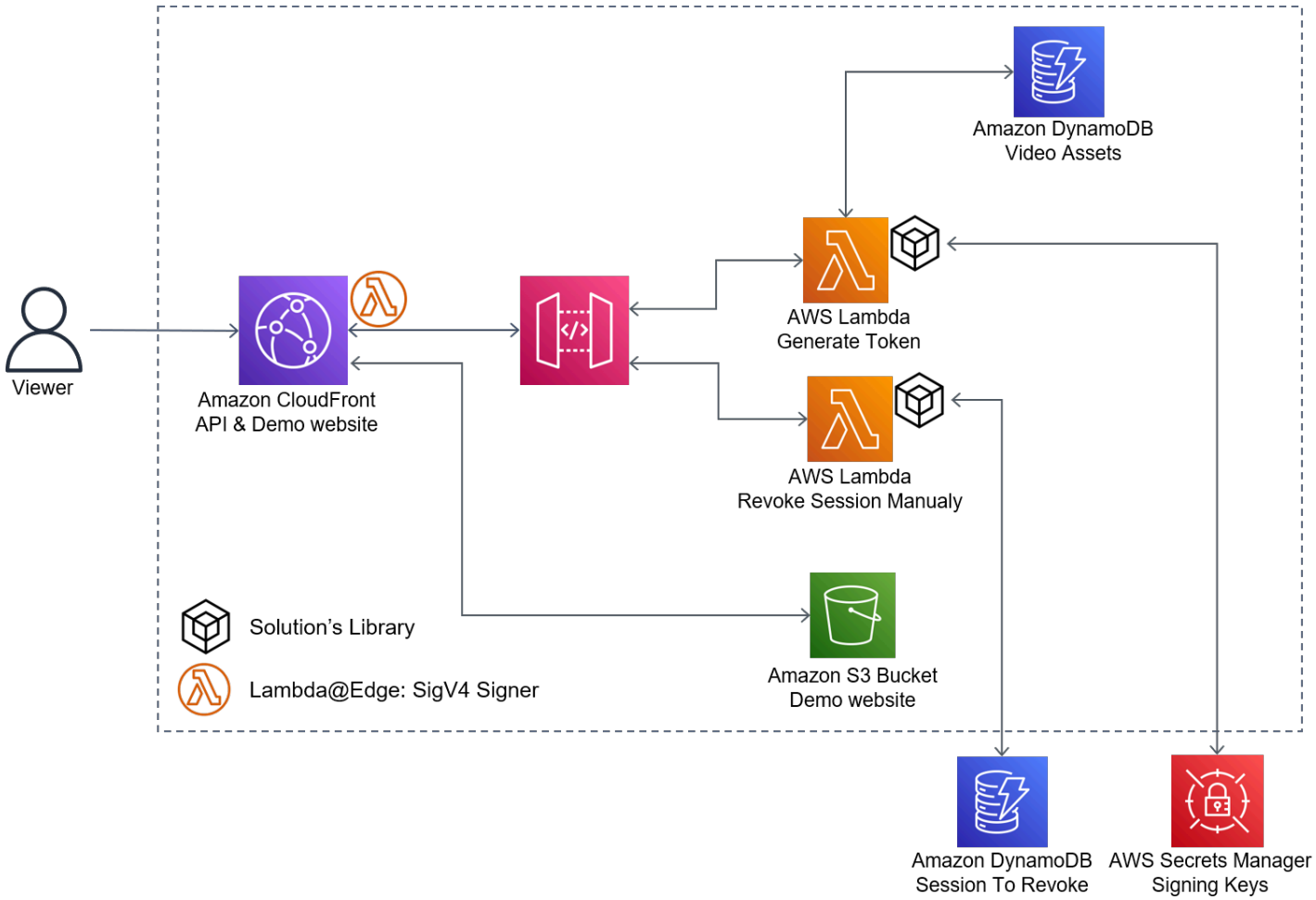
Sessions marked as manually added take higher precedence than the ones added through automatic session revocation module. Within manual sessions, they are ordered by timestamp which equals to the time when session was added to the DynamoDB table – newest sessions have higher priority. If there is any room left (as determined by WAF rule group WCU limit), session IDs populated by the automatic session revocation module are considered and appended to the list, as reminding list size limit allows. In this category of sessions, auto revocation module adds into DynamoDB table additional parameters that allow to assess most offending sessions, that were shared with large number of unauthorized viewers.

This is determined by the suspicion score property, where the higher the score, the bigger the anomaly in traffic levels linked to that session. A Lambda function fills remaining slots in the list with the session IDs with the highest score. As the list is compiled, it is pushed to WAF rule group by an API call which updates entire rule group.

DynamoDB also includes information about the individual components that final suspicion score comprises of – `IP_Rate, IP_Penalty, referrer_penalty, UA_penalty`. This information is included to better understand what contributes to the final score and to facilitate any troubleshooting.

The time to live (TTL) timestamp is calculated as 24 hours from the time, entry is added to the table, determining when DynamoDB will evaluate this item as expired, and eventually deleted by the background process that continuously inspect the timestamps in this column; refer to How it works: DynamoDB Time to Live (TTL). Utilizing this mechanism keeps the size of the list manageable and does not allow for sessions to accumulate in the table indefinitely.

API module



API module

The API module is made available in the solution to represent an example of how to integrate token management process into the `Playback` API Gateway API section of customer architecture. The API Gateway with two Lambda integrations is the central element of this module, responsible for performing token-related operations, that is generating the token and revoking a given session ID associated with the token. The two routes defined in HTTP API Gateway configuration are:

- `/sessionrevoke` (allowed for POST requests)
- `/tokengenerate` (allowed for GET requests)

As incoming request matches any of the two routes, an appropriate Lambda function is invoked. Lambda function associated with `/tokengenerate` path is responsible for issuing the playback URL for a requested video asset (video asset specified as query string parameter – id). This is

possible with the use of NodeJS module provided with the solution which exposes all the methods the Lambda function needs to handle token operations. However, before the token can be issued, the preceding steps must be completed in the Lambda code to collect and format the metadata about requested video asset and parameters that informs how the token must be built. In a typical production environment, customers can run a form of content management system to manage this type of required metadata. In the solution, the element is abstracted as a DynamoDB video assets table to store the necessary information for token generation. After you include the information about your video assets original (without a token) playback URL and fill in token-related parameters for each asset, Lambda code can fetch information by using video asset id that's assigned to the metadata records in DynamoDB table and generate the token accordingly. Last element involved in token generation process is the signing key that also need to be retrieved before token generation method is called. Solution's library has dedicated class and set of functions to obtain the keys from the Secrets Manager's secrets defined for the launched stack.

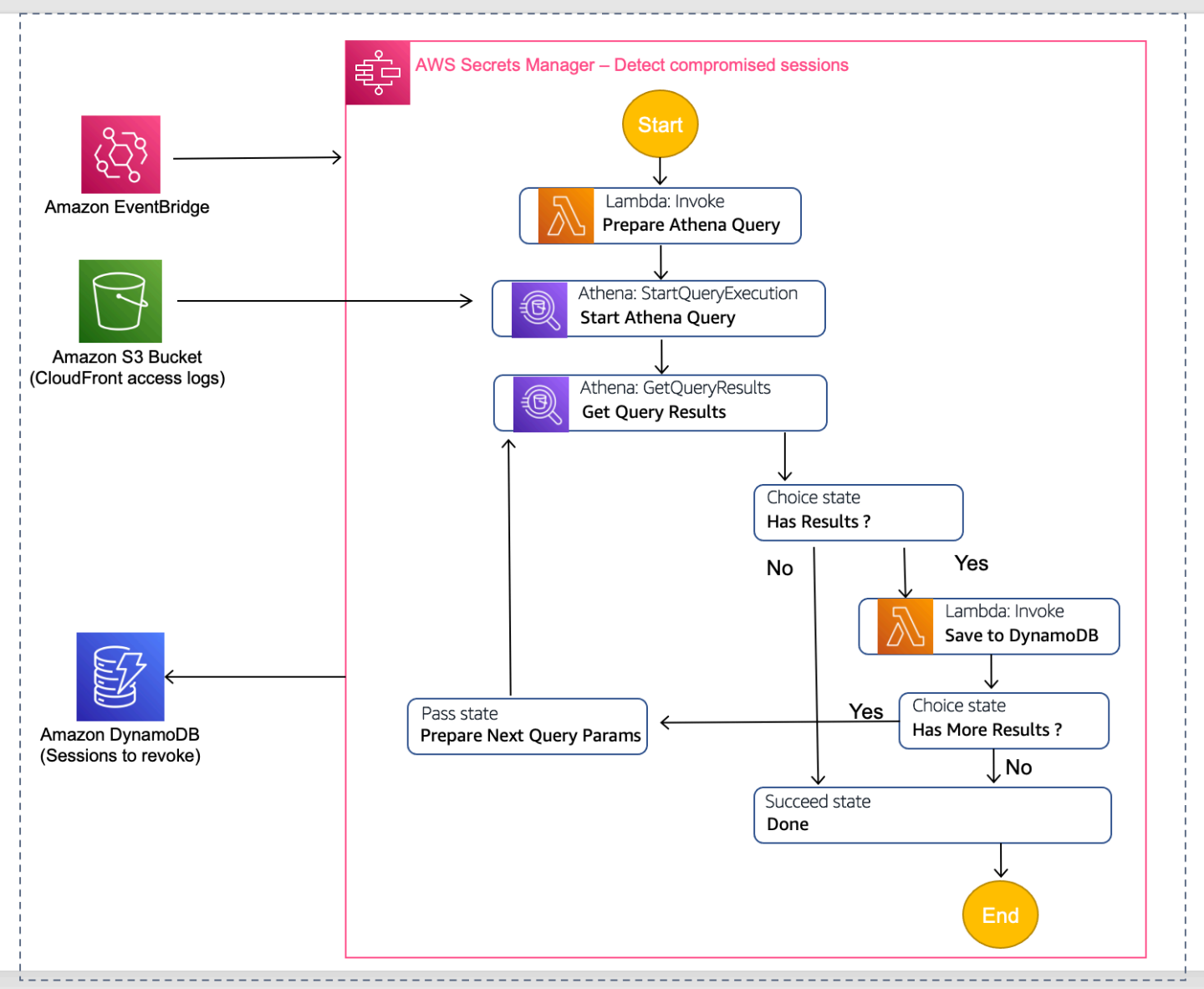
The second route (`/sessionrevoke`) points to another Lambda function which parses the input of the POST request to pull the session ID and put it in the ingest DynamoDB table from the base module as an entry point for revoking the session. Session ID submitted through this path are flagged as MANUAL type in DynamoDB, which guarantees this session will be prioritized over automatically pushed session as explained in previous section.

You can also choose to launch a demo website comprised of few additional elements in the architecture. The demo website is made available solely to test the solution and verify if it can be successfully integrated with your video workloads. However, we strongly recommend turning off the demo website before using the solution in a production environment with sensitive video content that should be protected. This is because the demo website provides a path to interact directly with API Gateway and obtain playback tokens by unauthorized viewers, or revoke legitimate sessions. When a demo website is activated, you can verify if, as a web client, you are able to acquire playback URL with the token that would grant you permission to watch the protected content. The demo website will also decode and expose structure of the token payload making it possible to confirm if the right parameters are included. Finally, it also allows testing manual revocation of the active playback session. The demo website is comprised of static assets stored in Amazon S3 bucket created when the solution is launched.

Both components of API module – API and static website are exposed through the same CloudFront distribution that separates the traffic between API Gateway and S3 bucket by using CloudFront cache behaviors. For the cache behaviors linked with API Gateway: `/sessionrevoke` and `/tokengenerate`, a dedicated Lambda@Edge function is attached and invoked for every API request to sign each of the outgoing requests with SigV4 signature. It is necessary as API Gateway

endpoint has IAM authorization activated to reject any token related requests without sufficient permissions as defined in IAM policies. IAM execution role configured for Lambda@Edge includes the right permissions that allows it to call the API Gateway endpoint, therefore requests signed by this function's invocation role credentials are accepted.

Auto session revocation module



Auto session revocation module: Detect compromised sessions

The base and API module provide the ability to react when, through external analysis, you identified a compromised playback session, if it was confirmed that session was used through unallowed distribution channels. Alternatively, you can also deploy additional and optional

modules which facilitate detecting sessions with a high probability of being compromised. As the process of detecting suspicious session is run regularly, the resulting session list is emitted to DynamoDB created under base module. These sessions are ordered and processed as explained in the Base module: session revocation workflow section.

The auto session revocation module design leverages AWS Step Functions to coordinate this entire multistep process. A predefined workflow is invoked periodically as specified in the created EventBridge rule. For ongoing video delivery streaming, set the period in the range of a few minutes to reduce the time it takes to detect and block suspicious sessions. Once workflow is initiated, the first step is to formulate the right SQL query based on the input parameters configured. The compiled SQL query is submitted as a job to Amazon Athena which initiates SQL query against the CloudFront distribution's access logs. Bear in mind that for this process to work, session IDs must be issued for the viewers together with access tokens. Based on the provided SQL query, Athena job will eventually complete the query run, listing all the sessions that are above the acceptable suspicion score threshold, which is another input controlled by the solution's operator. Multiple factors are taken into consideration when suspicion score level is evaluated, as explained in the Session Revocation Guide section, but put simply it is a measure of by how much request rate deviates from the median value and other signals indicative that multiple unique viewers are reusing the same session ID and token pair to watch the content simultaneously. When Athena job is completed and returned list of session is not empty, a Lambda function is invoked to ingest the session IDs with their suspicion scores and additional information, and put in DynamoDB table from the base module.

Before using the auto revocation module, collection of access logs to Amazon S3 must be configured for each CloudFront distribution, the traffic of which should be analyzed through this process. You must also set up a database and a table in Athena referencing access logs in the S3 bucket.

AWS services in this solution

AWS service	Description
Amazon CloudFront	**Core.** Validates secure tokens, permitting or denying access to video content.
AWS Secrets Manager	**Core.** Stores secrets holding signing keys for generating and validating viewer's tokens

AWS service	Description
AWS Step Functions	**Core.** Coordinates key rotation processes.
Amazon API Gateway	**Core.** Processes requests to generate the tokens for video playback, and to manually revoke specified playback sessions.
AWS Lambda	**Core.** Supports API Gateway to generate token for video playback and signs outgoing requests.
Amazon S3	**Core.** Provides storage of static assets for the demo website and CloudFront access logs.
Amazon DynamoDB	**Core.** Stores metadata about video assets and corresponding parameters used to generate tokens.
Amazon EventBridge	**Core.** Invokes session revocation workflow in AWS Step Functions.
Amazon Athena	**Core.** Runs SQL queries against CloudFron t access logs to list the suspicious video playback session ids with abnormal traffic characteristics.
AWS WAF	**Core.** Provides the list of playback sessions that should be blocked as they get identified as compromised.
AWS Systems Manager	**Supporting.** Provides application-level resource monitoring and visualization of resource operations and cost data.

Plan your deployment

This section describes the cost, security, Regions, and other considerations prior to deploying the solution.

Cost

You are responsible for the cost of the AWS services used while running this solution. The total cost for running this solution depends on the selected modules used in the solution and input parameters.

We recommend creating a budget through AWS Cost Explorer to help manage costs. Prices are subject to change. For full details, see the pricing webpage for each AWS service used in this solution.

Note

- Below cost estimates relate only to the elements created during the deployment of the solution. The charges associated with running the video workload on CloudFront, which must be created independently from this solution, are not included in the cost breakdown.
- Example cost calculations listed below demonstrate incremental costs incurred from newly created components in your account as an output of the solution.
- The total charges will also include the video delivery pipeline that consist of costs of running media content origin (for example, Amazon S3, AWS Elemental MediaPackage, among others) and delivery to the viewers through Amazon CloudFront. These costs are not specified below as this solution is designed to complement existing video streaming workloads implemented on Amazon CloudFront.

Assumptions

- The following examples provide a cost estimate for video streaming workload with 10 live streaming events per month, 60 mins in duration each, and driving 10,000 concurrent viewers. For each viewer, a playback token is generated once, just before the playout starts. It is assumed that during each event, 10 playback sessions are revoked manually, while 20 of them are detected and blocked as a result of automatic session revocation mechanism.

- Signing key is rotated automatically each day.
- To calculate the number of CloudFront Function invocations (example):

  A. Number of viewing sessions 10,000 viewers * 10 events = 100,000 sessions

  B. Average viewing duration per session (seconds) 60 minutes * 60 = 3600 seconds

  C. Segment length (seconds) 2 seconds and manifest request after 3 consecutive segment requests (3600/2 + 3600/(2*3) *100000 = 240M invocations

Base module

Cost to validate request tokens and key rotation

AWS service	Dimension/month	Cost [USD]
CloudFront Functions	240 million invocations	$24.00
Secrets Manager	3 secrets API call for 1 in 10 token generation operations Storage per secret per month	$1.25 $0.40 * number of keys
Step Functions	Number of transitions during key rotation workflow One rotation per day ~ 30/mo	< $0.01
Lambda	Lambda related costs during key rotation process One rotation per day ~ 30/mo	< $0.01
Total monthly cost:		**~$25.65 / month**

Note

This solution uses secrets key caching implemented in token generation method for Secrets Manager to reduce API calls and cost

Session revocation

(Part of base module but it's optional to use it)

Cost to block compromised playback sessions

AWS service	Dimension/month	Cost [USD]
AWS WAF	Web ACL + Rule Group + Rules (*it is assumed no WebACL was not used before for video delivery and AWS WAF is associated with CloudFront solely for session revocation purpose)	$6.14
AWS WAF	Requests – 240 million (*it is assumed no web ACL was not used before for video delivery and AWS WAF is associated with CloudFront solely for session revocation purpose)	$144.00
Total monthly cost:		**~$150.14 / month**

API Module

(Part of core module but it's optional to use it)

Cost to generate 100,000 of playback tokens per month

AWS service	Dimension/month	Cost [USD]
API Gateway	100,000 API calls	$0.10
DynamoDB	50,000 Read Request Units Assume 1 request = .5 RRU	$0.01

AWS service	Dimension/month	Cost [USD]
CloudFront (fronting API Gateway) (Data Transfer + Request charges)	100,000 HTTP requests with ~1kB response	$0.11
Lambda@Edge	100,000 function invocations	$0.60
Lambda	100,000 function invocations	$0.20
Total monthly cost:		**~$0.48 / month**

Note

Cost for API calls to Secrets Manager already included in Base module

Auto session-revocation

(In addition to session revocation costs)

Cost to run auto revocation pipeline

AWS service	Dimension/month	Cost [USD]
Step Functions	Number of transitions during session scanning and updating workflow	$0.14
Athena	CloudFront Access Logs Data Scanned* - 1545GB * *(single 1hr playback session produces ~270KB log data)*	$7.54
AWS WAF	Additional rules inserted into Rule Group from Session Revocation	$0.27

AWS service	Dimension/month	Cost [USD]
Total monthly cost:		**$7.95 / month**

Metrics monitoring

Cost of running the solution's dashboard

AWS service	Dimension/month	Cost [USD]
CloudWatch - Dashboard	Fixed cost for CloudWatch Dashboard	$5.00
CloudWatch Logs Insights – Data Scanned	Token verification results widget built on CloudWatch Logs insights – 72GB data scanned	$0.36
Total monthly cost:		**~$5.36 / month**

If you choose to deploy the demo website (which we recommend deactivating after you launch the solution in a production environment), the solution automatically deploys Amazon S3 bucket for storing the static website assets in your account, and use the same CloudFront distribution created in front of API Gateway endpoint. You are responsible for the incurred variable charges from these services.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared responsibility model reduces your operational burden because AWS operates, manages, and controls the components including the host operating system, the virtualization layer, and the physical security of the facilities in which the services operate. For more information about AWS security, visit AWS Cloud Security.

IAM roles

IAM roles allow customers to assign granular access policies and permissions to services and users on the AWS Cloud. This solution creates IAM roles associated with resources that needs to perform specific actions outlined in previous sections. Permissions defined in the policies created in the solution align with the principle of least privilege access, granting just those permissions that a specific component needs to fulfil its tasks fully. As one of the elements of the architecture is the solution's library that can be run outside of AWS environment, we recommend using a specific role with the right set of permission to perform the actions against AWS services implemented in the library. You can find a reference to that role in the output section of the deployed CloudFormation stack under the **RoleArn** key.

Amazon CloudFront

This solution deploys a demo website hosted in an Amazon S3 bucket. To help reduce latency and improve security, this solution includes an Amazon CloudFront distribution with an origin access identity, which is a CloudFront user that provides public access to the solution's website S3 bucket contents. For more information, refer to Restricting Access to Amazon S3 Content by Using an Origin Access Identity in the *Amazon CloudFront Developer Guide*. The same CloudFront distribution also interacts with API Gateway endpoint created for managing the access tokens. To maintain good security posture, API Gateway is configured to require AWS IAM authorization for invocation. Therefore, Lambda@Edge function with appropriate IAM permissions is used to sign CloudFront sourced requests. To further improve security of the demo website exposed to the user, a response header policy is attached in the configuration with a set of security headers returned to the viewer, **Strict-Transport-Security**, **X-XSS-Protection**, **X-Content-Type-Options**, **Referrer-Policy**, **X-Frame-Options**, **Content-Security-Policy**.

Solution's code library

The Secure Media Delivery at the Edge on AWS solution comes with a NodeJS library that was developed to make it easier to integrate solution into your Playback API workflow when adding token generation in it. Functions implemented in this library are built around API calls, made directly against AWS resources – API Gateway endpoint for token-related actions and DynamoDB table to submit the playback sessions that you want to be blocked. To make the API calls for AWS services work, IAM identity with right permission must be assumed when library's code is run. When run in an AWS environment, you can simply utilize the roles assumed in the context in which library code is initiated – for example, execution role of Lambda functions defined in API module:

`[Stack Name]_Generate Token` and `[Stack Name]_SaveManualSession`. If you choose to run your `Playback` API utilizing library provided functions outside an AWS environment, you must still assume the role using a standard approach of credential file, named profiles, or assuming role temporary through AWS Security Token Service; refer to Setting Credentials in Node.js. When you interact with the solution classes', you have an option to refer to a specific profile or Amazon Resource Name (ARN) reference of the role that must be assumed when underlying API calls are made. In this use case, we recommend providing ARN identifier of the dedicated role created specifically for the solution's library, as it has precisely defined permissions that allow only appropriate actions against specific resources created in the stack. You can find a reference to that role in the output section of deployed CloudFormation stack under **RoleArn** key.

Signing key protection

AWS Secrets Manager is a managed service to securely store and share the secrets based on the IAM defined permissions. A token-based access control mechanism, like the one used in this solution, works on the basis of generating and validating cryptographically created signature, which require a secret key to perform both tasks. AWS Secrets Manager stores the keys used in this solution. The keys are made available only to CloudFront Function and solution library methods, however they are distributed in a different way. CloudFront Functions receive updates with details about the new keys in a push model, when key rotation workflow generates and edits the keys known to CloudFront Function. Solution's library methods obtain the keys in the pull model by reaching out to AWS Secrets Manager to retrieve the keys when needed. Standard IAM based access model applies where the credentials or role used in the library must translate to appropriate IAM permissions that grant access to the secrets in the deployed stack. This solution has been designed in a way that you do not have to manually define and set keys, instead automation included in the base module addresses that. We recommend leveraging the key rotation mechanism available in the solution to update the signing keys regularly to avoid using the same key for a prolonged period of time.

API Gateway

HTTP API Gateway is configured to require AWS IAM authorization before the target Lambda functions responsible for token activities are invoked. Therefore, anonymous viewers without appropriate IAM permissions will not be able to make API calls. To successfully integrate other services with the API Gateway endpoint created in the solution, explicit IAM permissions must be granted to the corresponding user or role, allowing to invoke deployed API resource.

CloudFront prerequisites

The Secure Media Delivery at the Edge on AWS solution has been designed to enhance the security of an existing video streaming workload that already utilizes CloudFront for content distribution to the viewers. As you prepare to integrate the solution into your architecture, the following requirements and preceding steps must be reviewed before activating the solution for your workloads.

At the video origin level:

The only requirement that must be met in terms of configuration of your video at origin which produces media segments and corresponding manifest files is to ensure that all media objects that need to be protected by secure tokens, are referenced in the video manifest with a relative rather than absolute path. This solution operates on the basis of inserting a token at the beginning of the playback URL path and the assertion that client's player will repeat the token when subsequent elements that need to be processed are relative to the same root of the path, which is an inserted viewer token. Most of the video originating services compose the media manifest in that way. Once you confirm this also applies for your workload, proceed with the next steps to integrate the solution. If all or some of the objects in the manifest use absolute paths, which would override the token part when player makes a request for it, you must change your video origin configuration to replace absolute paths with relative paths. If this is not possible for all the objects, you can also exclude these specific objects from token validation step with the right adjustments in the CloudFront distribution configuration.

At the CloudFront distribution level:

In the existing CloudFront distributions used for the delivery of original video content, you must verify the following before you integrate this solution.

- Do the path patterns defined in cache behaviors match request URLs after you start inserting the access tokens? Comparing the URL path for the same object before and after secure tokens are added, you will find that a token appears as abstracted top-level directory in the URL path referencing video objects.
  - Before adding token: https://d1234.cloudfront.net/video_path_top_dir/video_path_subdir/object
  - After adding the token: https://d1234.cloudfront.net/**token**/video_path_top_dir/video_path_subdir/object

When verified against CloudFront distribution configuration, make sure that after token insertion appropriate cache behaviors match modified URL path including unique token values. Cache behavior requires the use of a wildcard at the beginning of the URL path pattern to match the arbitrary token value included in each request. In a common configuration in which you separate cache behavior for manifest files and video segments would like this:

	Precedence	Path pattern
○	0	*.m3u8
○	1	*.ts

Cache behavior - wildcard

That configuration requires no changes in the path pattern definition as preceding wildcard covers additional path component introduced by token presence. However, if you have more explicit path pattern definitions (for example, when you run multiple streaming channels), your cache behaviors configuration may resemble this:

	Precedence	Path pattern
○	0	/out/v1/df069e9593304746afc28eb0117f2dbd/*.m3u8
○	1	/out/v1/df069e9593304746afc28eb0117f2dbd/*.ts

Cache behavior – explicit path pattern

In that scenario, requests originated from the viewers with token included will not match with any of the listed cache behaviors as it starts with a fixed path pattern that does not accommodate part of an arbitrary token value at the beginning of the path. To change that, referenced path patterns can be modified as follows:

	Precedence	Path pattern
○	0	*/out/v1/df069e9593304746afc28eb0117f2dbd/*.m3u8
○	1	*/out/v1/df069e9593304746afc28eb0117f2dbd/*.ts
○	2	Default (*)

Cache behavior – non-fixed path pattern

- Because the token validation logic is operated by CloudFront Functions code generated as one of the solution resources, before that function can be associated with viewer request triggers for the appropriate cache behaviors, you must make sure there is no Lambda@Edge function associated with either viewer request or viewer response triggers for the same set of cache

behaviors. It is not possible to combine Lambda@Edge and CloudFront Functions associations for the same cache behavior.

- If you plan to leverage the session revocation feature offered by the solution, an AWS WAF web ACL must be created first and associated with the CloudFront distribution that will be integrated with the solution. If you run multiple CloudFront distributions that require token-based protection and session revocation capability, we recommend you run multiple stacks of the solution and associate created WAF rule group with separate AWS WAF web ACL dedicated for each distribution. This ensures efficient use of WAF Capacity Units (WCU) as each rule group will only store the sessions that are in use for a given distribution.
- Integrating your video streaming CloudFront distribution with auto session revocation requires additional steps to make the detail about all the incoming request accessible for Athena to distill suspicious sessions. Activating CloudFront access logs is the first step which delivers log files to the S3 bucket of your choice. For Athena to be able to run queries against these log files it needs additional metadata that will instruct how to parse and interpret log files, and also how to map column names to specific fields. You must define that information by creating database and table definitions in AWS Glue Data catalog. Refer to Querying Amazon CloudFront logs for a basic use case and additional information.

Supported AWS Regions

This solution can be launched in any AWS Region of your choice in which regional resources will be created:

- Secrets in Secrets Manager
- Step Functions workflows
- Lambda functions
- Dynamo DB tables

Note that because automatic session revocation module is deployed as a separate CloudFormation stack, you can deploy it in a different Region as the main stack. This can be beneficial in terms of reducing the costs arising from cross region Athena queries. Refer to Querying across regions for additional Amazon S3 data transfer.

No matter in what target Region you deploy the main stack, which includes the core components of the solution, a dependency with US East (N. Virginia) (us-east-1) Region will exist as some of the

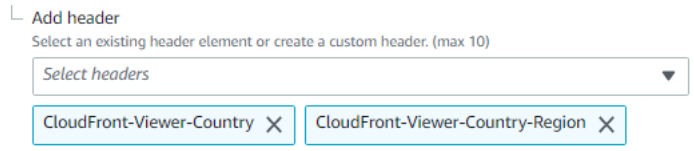
components must be defined in that region. This is because those components can be associated with the CloudFront distribution which is a global service. These components are WAF rule group and Lambda@Edge function for signing requests towards API Gateway. Both of these resources are created through custom resource logic synthesized in the main CloudFormation stack deployed in the target region.

Supported formats

This solution works with the video streaming workloads that use the most common adaptive bitrate streaming formats for distributing their video assets: HTTP Live Streaming (HLS), Dynamic Adaptive Streaming over HTTP (DASH), and Common Media Application Format (CMAF) which breaks down continuous video stream into discrete video and audio segments, suitable for Content Delivery Network (CDN) caching. Customers can use any media originating service which publishes video content in one or more of the mentioned formats, for example, Elemental MediaPackage, Elemental MediaStore, Amazon S3, or an external video packaging service.

Revise origin request policies

This solution provides you the ability to select a specific cache behavior in CloudFront configuration where token validation needs to be applied. Token validation logic takes viewer specific attributes, exposed at runtime through event object, to validate if the viewer using this token is in fact the viewer this token is for. This form of viewer uniqueness is achieved by including the token viewer specific attributes, some of them relating to the viewer's location, like country or region. Viewer location information is available only through CloudFront generated headers that token validation code has to pull that information from. However, that category of headers will only appear in the event object when these specific headers are specified either in cache or origin request policies. To prevent negative impact on cache hit ratio, rather than using cache policies we recommend that you add the two specific geolocation headers in each origin request policy that is associated with token protected cache behaviors. In the origin request policy definition, include both `CloudFront-Viewer-Country` and `CloudFront-Viewer-Country-Region` in the headers section.



Origin request policy - geolocation headers

Auto session revocation

Before you begin to use and deploy auto session revocation module in your architecture, review the following considerations to ensure the entire process is accurate and efficient.

- **Access log retentions** – Video streaming use cases at a large scale, inevitable will drive large amount of traffic. In turn, these will produce large volume of log entries which feed the auto session revocation analysis. Therefore, to optimize for costs we recommend looking into S3 lifecycle configuration options to expire and remove dated log files that do not provide actionable insights any more. This will not only save the costs of S3 storage but also amount of data that needs to be scanned by Amazon Athena which is one of the main cost components of this service.

- Another technique to reduce the cost associated with the size of data scanned by Amazon Athena is to employ log partitioning method to segregate the log files being produced in the supporting folder hierarchy that organize the files per year, month, day, and hour. This is because when you run Athena query, only the recent log entries represent the present usage of various playback sessions, usually up to tens of minutes at most. By limiting lookback period from the current time, you only dedicate the available resources to inspect current traffic and to block ongoing playback sessions rather the ones which were only active before. With partitioned log files, SQL query can take advantage of that by limiting the scope of the target files to a specific time period and vastly minimize the amount of data that is scanned. We recommend reviewing one of the possible solutions for partitioning CloudFront Logs as described in Analyze your Amazon CloudFront access logs at scale. Auto session revocation module supports boot partitioned and non-partitioned log structure.

- One of the key dimensions of determining which sessions have been compromised, is the request rate corresponding to a given session relative to the median value. The assumption behind it is that when you issue an individual playback session you should observe approximately the same levels of request rates oscillating around median for each viewer. If playback session and corresponding access token is reused by multiple viewers, this would stand out as abnormal request rate, and the suspicion score increases as more viewers would share the same token and session ID. This assumption holds true when the video assets delivered through the target CloudFront distribution theoretically prompts the same request rate across legitimate viewers. If the request rate differs between video assets or renditions for the same content, the resulting median value will not be representative reference point and this can lead to identifying some

legitimate session ID as compromised incorrectly. To avoid that effect, it is important to verify your assets have the same characteristic in terms of segment length and that you do not see significant differences in request rate from various clients. If you have video assets of varying characteristics, but you want to leverage auto session revocation mechanism, you can either turn off tracking request rates as one of the score components, or spread your video assets delivery across multiple CloudFront distributions according to their characteristic, and run separate auto session revocation pipelines for each. Note that tracking request rates in log processing pipeline aggregates multiple byte range requests against the same object. This means that the video clients requesting full segments will yield the same request rate result as the clients downloading the same segments partially with multiple range request.

Large viewership spikes

The Secure Media Delivery at the Edge on AWS solution has been designed to support workloads of various scale including streaming services that draw very high viewership levels by using highly scalable components in processing the access tokens and processing log entries. One of the scaling factors is also ability to generate the tokens at the rate of viewers requesting playback URLs, and with the ability to scale underpinning resources to produce and serve access tokens back. In the design of this solution, access token is generated only once for each playback session and repeated by the same viewer for subsequent requests which vastly reduces the load at the playback API stage provided that new playback sessions spread over time. However, it is still possible that during specific periods of time, the load on playback API resulting from new playback requests and resulting number of parallel processes producing the tokens can exceed the available limits or underlying resource capacity. A typical example would be highly popular live streaming event, starting at a precise moment in time leading to excessive number of requests from new viewers starting the stream almost simultaneously. If your workload experiences this type of event pattern or simply serves very high number of new viewers in a steady state operation, make sure the underlying compute assets are able to scale appropriately to serve expected number of new viewers. In the reference architecture, the API module is capable of running multiple concurrent processes responsible for generating the token and supplying playback URL with the token in response. This is possible by the use of the standard scaling model of Lambda. You should monitor the concurrency metric in the region where the solution was deployed and request an increase if you are approaching this limit, or when you anticipate an upcoming event can drive exceedingly high viewership level. For the sudden spikes of playback API requests, you must also account for the rate at which Lambda concurrency can raise, which is 500 per minute. If this proves to be

insufficient for the type of events you serve, consider using Lambda provisioned concurrency to improve the ability to absorb rapid increase of new viewers.

The other aspect of scalability in the process of generating the tokens are the limits on API calls for AWS Secrets Manager. Note that in default implementation, a solution's library method reaches out to Secrets Manager to obtain the signing keys required to issue a token. In the region where secrets are stored, API rate quota exists which limits the number of **GetSecretValue** calls made from solution's library method and equals to 5000 transactions per second. This does not indicate that the maximum token generation rate is at the same level. In the solution's library implementation of managing the secrets, a local memoization technique is used to cache the secrets retrieved in the context of running process. You need to specify for how long a secret should be cached in memory so that it can be reused by the same threads initiated in the scope of the same process. For this reason, to enhance reusability of the once retrieved key, we recommend producing the tokens by using long running process to serve the requests and asynchronous threads that could share the same object holding the keys. This will effectively reduce the number of API calls towards Secrets Manager. For large-scale events that need to run a large number of parallel processes, to the point when Secrets Manager API calls limit can become a concern, it is possible to define custom key retrieval function that would introduce another layer of caching. For instance, you can create a dedicated function which would retrieve original key from AWS Secrets Manager but also store it in a shared space between the processes, for instance using in-memory data store like Amazon ElasticCache.

Alternative approaches to carry the token

The default and suggested method for including the token persistently throughout the viewer playback time is to use URL path. By inserting the token at the beginning of the playback URL which makes a reference point for the player for subsequent requests, it makes the solution more universally supported as it uses the most fundamental mechanism of HTTP delivery, and minimizes implementation efforts as this mechanism is transparent from the video origin standpoint. However, if this approach is not viable for your specific use case, you can choose a different way of attaching the token with the requests which does not have to rely on the URL path. Cookies, custom headers, query string parameters are potential other carriers for the token and using the solution's library it is possible to generate a plain token that would normally appear in URL path in default mode. However, this involves customization of CloudFront Functions code to look up and parse the token and session ID in a place other than the URL path, by modifying the source code appropriately. The rest of the token validation logic would remain unchanged regardless of how the token is attached with the request. You must ensure token stickiness so that with a different

approach to carry the token, this token is repeated in the future requests made by the same client for the protected objects.

Quotas

Service quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account.

Quotas for AWS services in this solution

Make sure you have sufficient quota for each of the services implemented in this solution. For more information, see AWS service quotas.

Use the following links to go to the page for that service. To view the service quotas for all AWS services in the documentation without switching pages, view the information in the Service endpoints and quotas page in the PDF instead.

As you evaluate the deployment of the Secure Media Delivery at the Edge on AWS solution, one limit that should be accounted for is WCU capacity of the web ACL used with CloudFront, and when session revocation module is to be launched. The default WCU limit for AWS WAF web ACL equals to 1500 WCU which is a shared capacity for the customer defined and managed WAF Rules, as well as the rule group created in the main module of the solution. For any rule group created, its WCU limit must be declared at the time of its creation and the value you specify at this stage is what gets consumed from web ACL general WCU limit. Rule group WCU limit cannot be modified after creation, therefore you must plan in advance how much WCU you want to allocate for session revocation rule group (recall that a rule to block a single session is worth 2 WCU). The limit you plan to apply for the rule group when launching the solution cannot exceed WCU headroom left in the web ACL that the rule group will be attached to. If there is not enough headroom left for the size of the rule group you plan to create when launching the solution, request WCU limit increase for that target web ACL to increase available WCU headroom.

Deploy the solution

This solution uses AWS CloudFormation templates and stacks to automate its deployment. The CloudFormation template specifies the AWS resources included in this solution and their properties. The CloudFormation stack provisions the resources that are described in the template.

Prerequisites

This solution is designed to work with Amazon CloudFront distributions used for video streaming. If you do not have one configured already, complete the applicable task before you launch this solution. For testing purposes, you can create video delivery pipeline including CloudFront distribution using the Live Streaming on AWS solution.

Refer to CloudFront prerequisites, which outlines which CloudFront configuration settings should be revised prior to launching the solution.

Inspect the body of video manifests published by the video origin service in use. If your video origin service references other objects using absolute URL paths, configure it to switch to relative paths instead.

Deployment process overview

Before you launch the solution, review the cost, architecture, security, and other considerations discussed earlier in this guide.

Time to deploy: Approximately 5-10 minutes

Step 1. Launch the stack

Step 2. Define video assets and token policies

Step 3. Prepare your CloudFront distributions

Step 4. Test the solution

> **Important**
>
> This solution includes an option to send anonymized operational metrics to AWS. We use this data to better understand how customers use this solution and related services and

products. AWS owns the data gathered though this survey. Data collection is subject to the AWS Privacy Notice.
To opt out of this feature, download the template, modify the AWS CloudFormation mapping section, and then use the AWS CloudFormation console to upload your updated template and deploy the solution. For more information, see the Anonymized data collection section of this guide.

AWS CloudFormation template

You can download the CloudFormation template for this solution before deploying it.

View template

secure-media-delivery-at-the-edge-on-aws.template - Use this template to launch the solution and all associated components. The default configuration deploys the core and supporting services found in the AWS services in this solution section, but you can customize the template to meet your specific needs. Use the CDK deployment model if you want to deploy the auto session revocation module.

Note

AWS CloudFormation resources are created from AWS CDK constructs.

Note

If you have previously deployed this solution, see Update the solution for update instructions.

Step 1: Launch the stack

Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 5-10 minutes

1. Sign in to the AWS Management Console and select the button to launch the `SECURESTREAM.template` AWS CloudFormation template.

2. The template launches in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar.
3. On the **Create stack** page, verify that the correct template URL is in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack. For information about naming character limitations, see IAM and AWS STS quotas, name requirements, and character limits in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

   Session Revocation

Parameter	Default	Description
Retention	`30`	Expressed in minutes. Retention time for the sessions submitted for the revocation. After retention time elapses session is no longer considere d for blocking and will be removed from WAF Rule Group next time it is updated.
Web ACL capacity units (WCU)	`100`	WCU limit allocated to the AWS WAF Rule Group created to store the rules to block revoked sessions. Note this value is immutable and can't be changed after

Parameter	Default	Description
		Rule Group is created. Single session ID included in the Rule Group utilizes 2 WCU from the configured limit.

Key Rotation Frequency

Parameter	Default	Description
Week of the month	N/A	Specify the week number in each month that key rotation will be scheduled for. This parameter can be set to a value from a range 1 to 4.
Day of the week	N/A	After selecting a week in a month, provide a specific day in that week when key rotation should occur. Value from 1 to 7, where 1 means Monday and 7 means Sunday.
Hours	N/A	An hour when key rotation workflow will be triggered.
Minutes	N/A	A minute in the selected hour when key rotation workflow will be triggered.

DASH Stream

If you do not have a video asset available in DASH format, you can leave the inputs empty in which case example values will be set. You can augment these inputs after the stack is deployed.

Parameter	Default	Description
Hostname for asset delivery	`https://d1234.cloudfront.net`	Domain name served by CloudFront distribution hosting video following protocol prefix (**http://** or **https://**). If no input is provided an example, default value will be set in the target DynamoDB table.
URL path for asset delivery	`/video/2/index.mpd`	Full URL path of the video asset. This parameter must start with '/' and point to an object used by the player to initiate a playback, like master manifest (mpd file). If no input is provided an example, default value will be set in the target DynamoDB table.
TTL for token	`+30m`	**Mandatory.** Time period determining for how long newly issued token will be valid. If not specified , example values will be populated.

HLS Stream

If you do not have a video asset available in HLS format, you can leave the inputs empty in which case example values will be set. You can augment these inputs after the stack is deployed.

Parameter	Default	Description
Hostname for asset delivery	https://d1234.cloudfront.net	Domain name served by CloudFront distribution hosting video following protocol prefix (**http://** or **https://**). If no input is provided an example, default value will be set in the target DynamoDB table.
URL path for asset delivery	/video/1/index.m3u8	Full URL path of the video asset. This parameter must start with '/' and point to an object used by the player to initiate a playback, like master manifest (m3u8 file). If no input is provided an example, default value will be set in the target DynamoDB table.
TTL for token	+30m	**Mandatory.** Time period determining for how long newly issued token will be valid. If not specified , example values will be populated.

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review and create** page, review and confirm the settings. Select the box acknowledging that the template will create IAM resources.
9. Choose **Submit** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a CREATE_COMPLETE status in approximately 5 to 10 minutes.

Step 2. Define video assets and token policies

After the solution's CloudFormation stack is deployed, you can define your video assets details (if you haven't provided them as CloudFormation stack input) by following the next steps:

1. After the solution has deployed, navigate to the DynamoDB console and **Explore items** page under **Tables.**
2. Select the table which name starts with following string: **[Stack Name]-ApiDemoTable.**
3. Under **Items returned** section you can add and modify the list of items which detail original video asset's hostname with URL path and a token policy for that content. In the demo web page deployed in the solution, two video assets are requested by the use of their ID corresponding to the entries in this table. To successfully start a video playback on the website, make sure that table entry with the **ID** that equals to 1, references HLS stream and the other one with ID of 2, to DASH stream.
4. To edit the details of the HLS stream, select an item with ID that equals to 1 and select **Action,** then choose **Edit Item.**
5. Make sure that the **endpoint_hostname** fields and **url_path** are filled in correctly as instructed in the previous **HLS Stream** parameter description.
6. Expand **token_policy** property and modify the values of predefined properties which determine the parameters of the output token that will be created for this specific video asset.
7. Choose **Save changes** to submit the changes.
8. For DASH stream, repeat steps from 4 to 7.
9. Test if API Gateway returns a valid playback URL with a secure token. Record the distribution hostname fronting API Gateway. You can find it in the launched stack output tab with a key value starting with **ApiEndpointsDistributionDomainName**. Using a utility tool like curl, make an HTTP request as follows:

For HLS:

```
curl [ApiEndpointDistributionDomainName]/tokengenerate?id=1
```

For DASH:

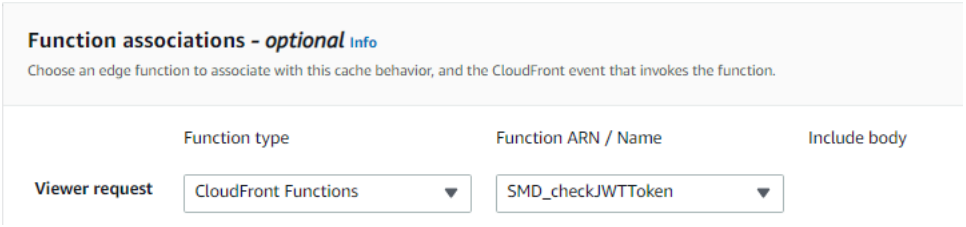
```
curl [ApiEndpointDistributionDomainName]/tokengenerate?id=2
```

In response, you should see the playback URL comprised of video asset hostname and URL path, with the secure token added at the beginning of the original URL path.

Step 3. Prepare your CloudFront distributions

After deploying the CloudFormation stack and configuring DynamoDB table with inputs needed to generate the token, follow the additional steps below to integrate the Secure Media Delivery at the Edge on AWS solution with the existing CloudFront distributions used to deliver video streams.

1. Validate if the defined cache behaviors related to the objects which are supposed to be token protected have their path pattern set correctly as described in the Design considerations section.
2. When using viewer's geolocation as one of the token attributes, make sure the same cache behaviors have origin request policies attached, which include CloudFront-Viewer-Country and CloudFront-Viewer-Country-Region headers.
3. For each cache behavior subject to token protection, associate a function created when solution's stack was launched. From CloudFront's console, open distribution settings and navigate to a specific cache behavior configuration. In **Function associations** section, from **Viewer Request** select **CloudFront Function** event and form **Function ARN / Name**, then select the **[Stack Name]_checkJWTToken** function.



Function associations

4. Choose **Save Changes** and repeat the above steps for each distribution and cache behavior where token validation mechanism must be in place.
5. If you intend to use manual session revocation, add a WAF rule group created for storing session IDs (identified to be blocked) to the web ACL associated with the CloudFront distribution used for streaming video content. On the web ACL page, select **Global (CloudFront)**, then select the web ACL associated with your CloudFront distribution and select the **Rules** tab then choose **Add Rules** > **Add my own rules and rule groups.**

6. In the **rule type** setting, select **Rule group** and enter a name to the rule you are defining. Under **Rule Group**, select the **[Stack_Name]_BlockSessions** rule group from the dropdown list .
7. Choose **Add rule** and adjust the priority of this rule group within the web ACL, then choose **Save**.

Step 4. Test the solution

After all the components and configurations that integrate your video delivery workload with the solution are in place, you can use the demo website to test the end-to-end workflow.

1. From the **Outputs** tab of the solution's CloudFormation stack, refer to the output starting with **ApiEndpointsDistributionDomainName** and select the link which will take you to the demo website.
2. You will see two buttons, one for each format – HLS and DASH, each corresponding to the video assets with ID attributes 1 and 2 respectively, configured previously in video assets DynamoDB table.
3. Choose either **HLS** or **DASH** to initiate an API call from the web client requesting playback URL and secure token.
4. If the playback URL and token were retrieved successfully the video playback will start. You can switch between the streams at any time.
5. Upon successful API request that returned a playback URL with the token, a demo website displays the details about the content of the token, showing the claims included in its body. These can be helpful in performing troubleshooting activities.
6. To test manual session revocation, select **Revoke current session** to see how current playback session will get blocked when the session ID corresponding to the used token will start being blocked by AWS WAF.
7. If you need to generate a new token and refresh current playback session, select the **Refresh token** option on the website.

CDK deployment

In addition to the CloudFormation template, you can also use the CDK project this solution was built on to integrate it into your environment. Review the following considerations to make an informed decision whether to use the CDK deployment model over the default CloudFormation template.

- With CDK, you can selectively choose which modules and elements needs to be deployed for each new stack you create. While the base module is always deployed, API module, demo website, and auto session revocation are optional and each of the listed components can be turned on or off. However, the CloudFormation template has all the components (except auto session revocation module) integrated within the template, and cannot be turned off.
- Auto session revocation module is only available when deployed through CDK as an optional module.
- CDK deployment model makes it easier to manage the configuration inputs when running multiple stacks.

If you decide that the CDK deployment model is more suitable for your workloads, use the following procedure to launch this solution in your environment.

Prerequisites

- AWS Command Line Interface
- Node.js 12.x or later
- AWS CDK 2.24.1

Deployment procedure

If you are deploying the solution for the first time using the CDK toolkit, run **cdk bootstrap** to setup the required CDK resources in your AWS account. For more details on using the CDK bootstrapping process, refer to Bootstrapping.

> **Note**
>
> If you're planning on using multiple regions, the bootstrapping process must be done for each AWS region. In case you have gone through the process of installing CDK bootstrap tooling in a given account and region, you can ignore this step.

1. Clone the solution source code from the repository.

```
git clone https://github.com/aws-solutions/secure-media-delivery-at-the-edge.git
```

2. Install the dependencies of the project to make it ready to use. To do so, run the below commands in the folder where you cloned the source code to.

   On Linux

```
cd source
```

```
./install_dependencies.sh
```

   On Windows

```
cd source
```

```
./install_dependencies.ps1
```

3. Run the built-in wizard which will prompt you with questions about the modules to deploy and the number of parameters which determine the operations of the functional components chosen to be deployed.

```
npm run wizard
```

   For more details about the meaning and usage of parameters prompted in the wizard, refer to the Design considerations and Session revocation sections.

   The wizard will then generate a configuration in the `solution.context.json` file that is located in the same CDK project directory. This config file stores all the inputs provided through

the wizard and can be modified any time you want to choose certain parameters and redeploy the stack.

4. Deploy the solution in your account.

```
npx cdk deploy --all
```

5. After CDK deployment finishes, follow the steps from 2 to 4 as described in the Deployment process overview section for CloudFormation.

> **Note**
>
> When running from your local environment, the determination of the target account and region will be based on the AWS profile in use as specified in the config and credentials file. If you want to deploy the stack in multiple accounts and regions, you must define the appropriate profiles associated with the account and right default region that you need. For more information refer to Specifying credentials and region.

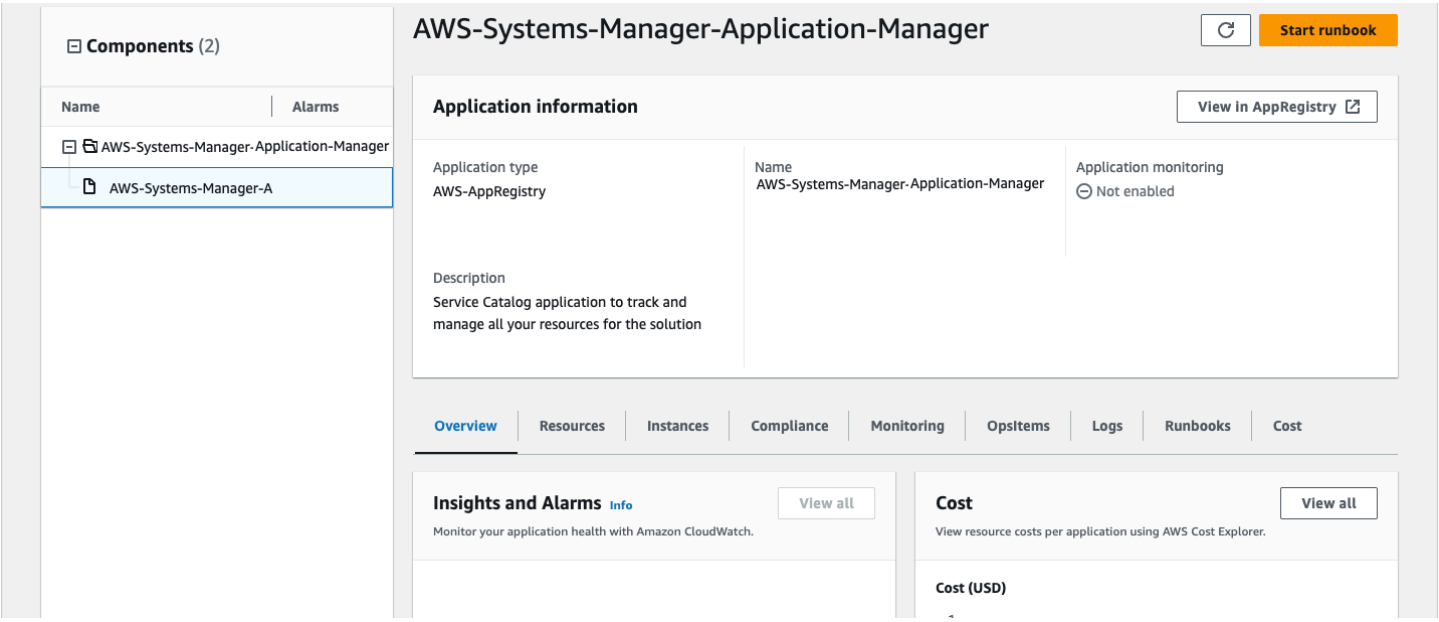
Monitor the solution with Service Catalog AppRegistry

This solution includes a Service Catalog AppRegistry resource to register the CloudFormation template and underlying resources as an application in both Service Catalog AppRegistry and AWS Systems Manager Application Manager.

AWS Systems Manager Application Manager gives you an application-level view into this solution and its resources so that you can:

- Monitor its resources, costs for the deployed resources across stacks and AWS accounts, and logs associated with this solution from a central location.
- View operations data for the resources of this solution (such as deployment status, CloudWatch alarms, resource configurations, and operational issues) in the context of an application.

  The following figure depicts an example of the application view for the solution stack in Application Manager.



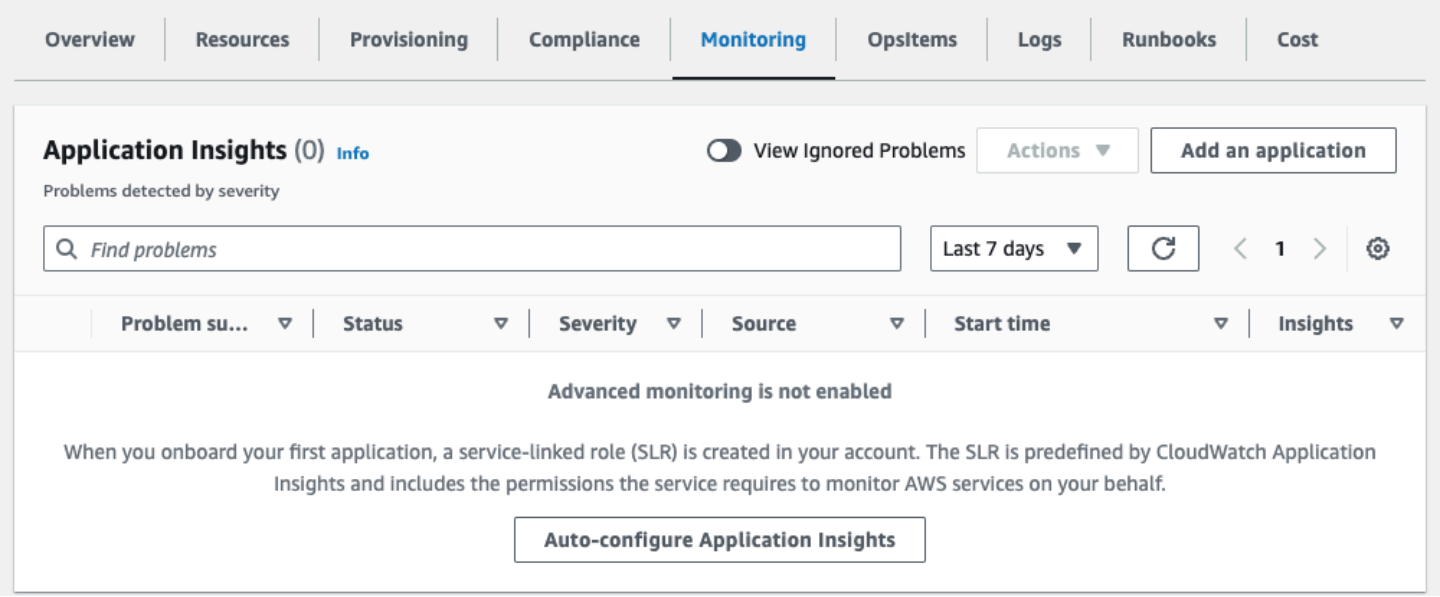
Solution stack in Application Manager

Activate CloudWatch Application Insights

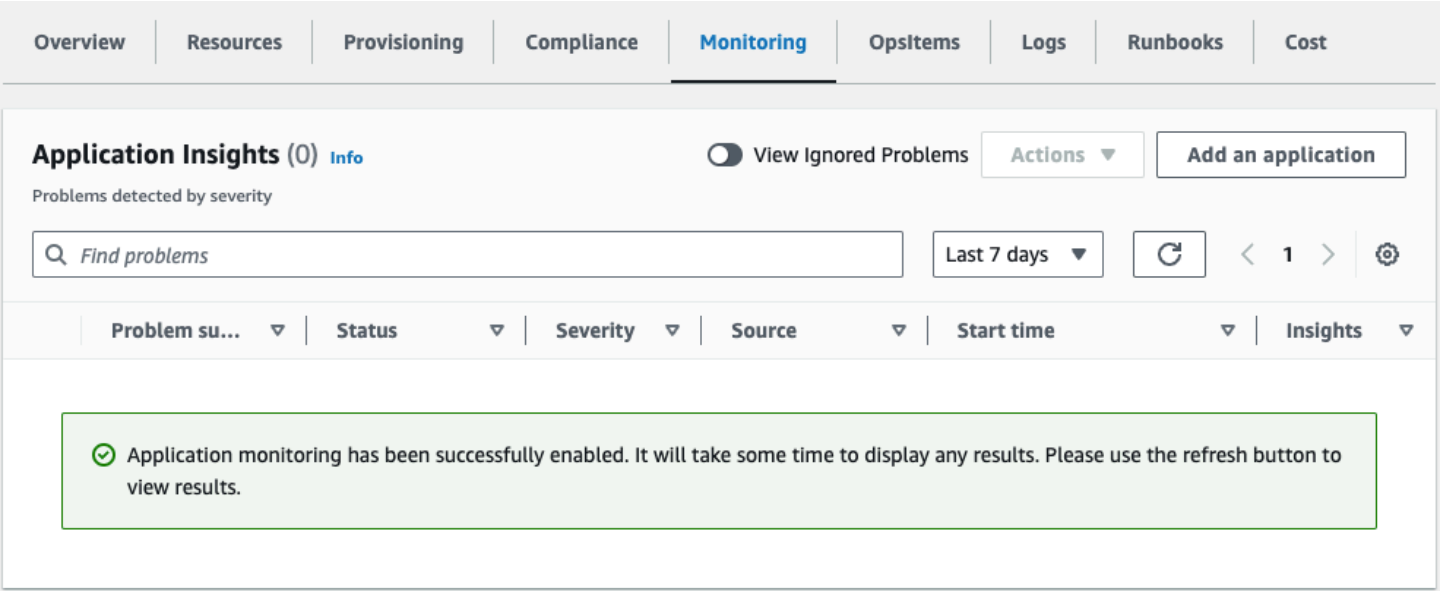
1. Sign in to the Systems Manager console.

2. In the navigation pane, choose **Application Manager**.
3. In **Applications**, search for the application name for this solution and select it.

   The application name will have App Registry in the **Application Source** column, and will have a combination of the solution name, Region, account ID, or stack name.
4. In the **Components** tree, choose the application stack you want to activate.
5. In the **Monitoring** tab, in **Application Insights**, select **Auto-configure Application Insights**.



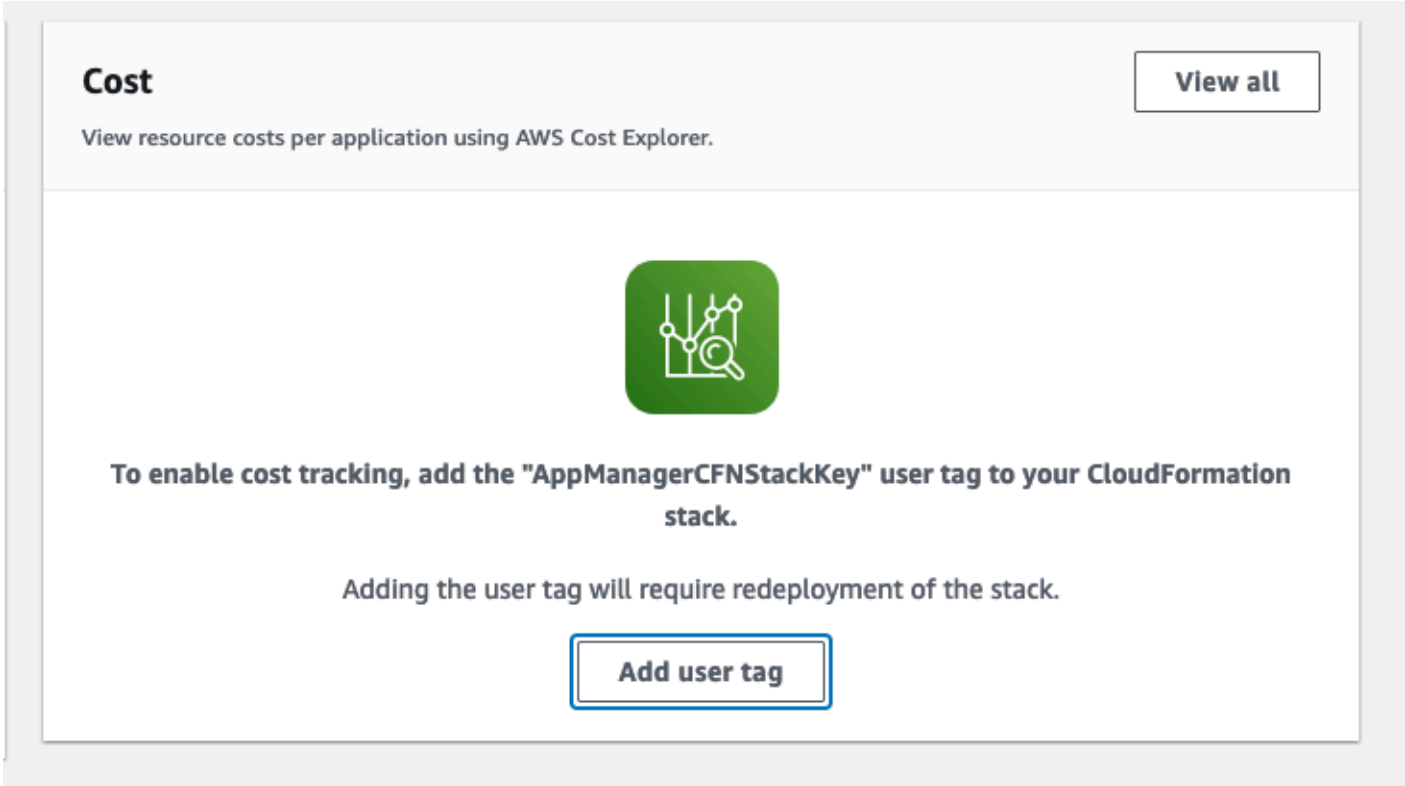
Monitoring for your applications is now activated and the following status box appears:



Confirm cost tags associated with the solution

After you activate cost allocation tags associated with the solution, you must confirm the cost allocation tags to see the costs for this solution. To confirm cost allocation tags:

1. Sign in to the Systems Manager console.
2. In the navigation pane, choose **Application Manager**.
3. In **Applications**, choose the application name for this solution and select it.
4. In the **Overview** tab, in **Cost**, select **Add user tag**.



5. On the **Add user tag** page, enter `confirm`, then select **Add user tag**.

The activation process can take up to 24 hours to complete and the tag data to appear.

Activate cost allocation tags associated with the solution

After you confirm the cost tags associated with this solution, you must activate the cost allocation tags to see the costs for this solution. The cost allocation tags can only be activated from the management account for the organization.

To activate cost allocation tags:

1. Sign in to the AWS Billing and Cost Management and Cost Management console.
2. In the navigation pane, select **Cost Allocation Tags**.
3. On the **Cost allocation tags** page, filter for the `AppManagerCFNStackKey` tag, then select the tag from the results shown.
4. Choose **Activate**.

AWS Cost Explorer

You can see the overview of the costs associated with the application and application components within the Application Manager console through integration with AWS Cost Explorer. Cost Explorer helps you manage costs by providing a view of your AWS resource costs and usage over time.

1. Sign in to the AWS Cost Management console.
2. In the navigation menu, select **Cost Explorer** to view the solution's costs and usage over time.

Update the solution

If you have previously deployed the solution, follow this procedure to update the solution's CloudFormation stack to get the latest version of the solution's framework.

1. Sign in to the CloudFormation console, select your existing Secure Media Delivery at the Edge on AWS CloudFormation stack, and select **Update**.
2. Select **Replace current template**.
3. Under **Specify template**:
   a. Select **Amazon S3 URL**.
   b. Copy the link of the latest template.
   c. Paste the link in the **Amazon S3 URL** box.
   d. Verify that the correct template URL shows in the **Amazon S3 URL** text box, and choose **Next**. Choose **Next** again.
4. Under **Parameters**, review the parameters for the template and modify them as necessary. For details about the parameters, see Step 1. Launch the stack.
5. Choose **Next**.
6. On the **Configure stack options** page, choose **Next**.
7. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create IAM resources.
8. Choose **View change set** and verify the changes.
9. Choose **Update stack** to deploy the stack.

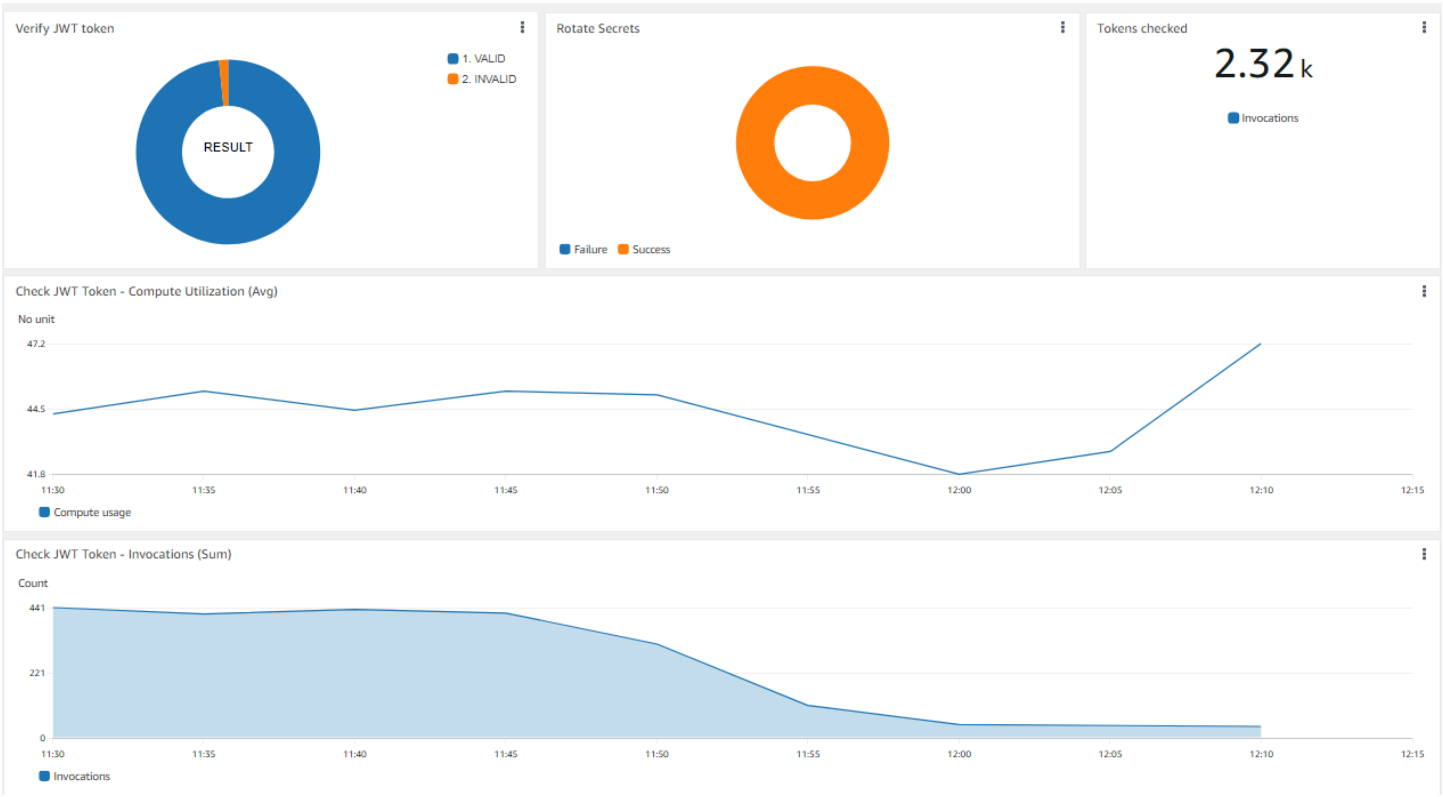
You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a UPDATE_COMPLETE status in approximately 5-10 minutes

Troubleshooting

If these instructions don't address your issue, see the Contact AWS Support section for instructions on opening an AWS Support case for this solution.

Monitoring dashboard

As you begin to use the Secure Media Delivery at the Edge on AWS solution, keep track of the operational metrics relating to various workflows comprising the solution. A good place to start to get an overview of the metrics relevant for this solution is CloudWatch Dashboard which is created automatically when you launch the solution. You can find it in CloudWatch console in the Dashboards view, present under the **Custom Dashboards** list. For each deployed stack there is a corresponding dashboard named: **[Stack Name]-Secure-Media-Delivery.**



Example monitoring dashboard

The metrics on the dashboard indicate if specific components and processes comprising the solution operate as expected, and if there are any anomalies that must be investigated. The metrics contained in the dashboard widgets indicate:

- Number of viewer tokens that CloudFront Function validation logic evaluated as valid and invalid.
- Number of key rotation procedure implemented in a given time frame and how many of the attempts ended successfully
- Total number of validated tokens
- Average compute utilization over time of the token validation function
- Number of token validation function invocations over time
- Number of tokens generated by the Lambda function from the API Module

Failed token validation

Issue: Legitimate viewer request is rejected by CloudFront and returns a 401 response due to failed token validation.

Resolution:

1. Retrieve the request ID of the failed request. CloudFront Request ID value can be either found in the response header **x-amz-cf-id** when the request is made or in the CloudFront access logs under **x-edge-request-id** field.
2. Open the CloudWatch console in the same account where the solution stack is implemented, and select `us-east-1` from the Region selector in the upper-right corner.
3. Navigate to **Logs,** select **Log Groups** and find a log group for the CloudFront Function created by the solution. The name of the log group will be similar to `/aws/cloudfront/function/[Stack Name]_checkJWTToken`. Select the log group identified.
4. Narrow down the time window for the log streams. Select **Search all** and in the filter panel limit the time boundaries to when the failed request was made.
5. In **Filter events,** enter the request ID recorded in step 1.

In the result output, you will see the log entries emitted from CloudFront Function that explains at which stage of the token verification it failed.

Internal signature verification failed error

Issue: When the token was generated, if the viewer attributes do not match with the attributes CloudFront function found in the event object, it results in a failed token error: **Internal signature verification failed**. When you generate the token and decide to use one or more of the following

parameters: **session Id**, **headers**, **query string**, **viewer country**, **viewer region**, **viewer IP address** this will produce an internal signature build from all the selected inputs, inserted in the token payload as **intsig** claim. Based on the other claim values, CloudFront Function reproduce that input string using the selected attributes by reaching them from the event object available at runtime. If any of these attributes are missing or changed their value form when the token was generated, it will cause a mismatch of signatures and result in this error.

Resolution:

1. From the CloudFront Function Log group identify the entries associated with failed request (as specified in the description of the previous issue), and find an entry which includes **Indirect attributes input string** which displays an input string recreated by CloudFront Function from the event object. This string will contain concatenated list of attributes with colon : as a delimiter. Take a note of it, in particular session ID which should be first element in that concatenated string.

   > **Note**
   >
   > If you do not use session ID, we recommend that you activate it when you start using and testing the solution to facilitate troubleshooting.

2. On the **Log Groups** page, change the region to the one where the solution stack was implemented with API module, using region selector.
3. Find the log groups associated with the Lambda function that generates the token. Select the log group name starts with /aws/lambda/[Stack Name]_GenerateToken.
4. Narrow down the time window for the log streams. Select **Search all,** and in the filter panel limit the time boundaries to when the token for the viewer was made.
5. In **Filter events**, enter the same session ID that was found in step 1 in the CloudFront Functions log stream.
6. In the results output, find a line which includes **Input for internal signature:** which was an input string used for generating internal signature.
7. Compare the values of both input strings and identify the attributes that do not match, or if any attribute is missing for CloudFront Function entry.
8. Consider common causes for discrepancies in the input string values:
   - Mismatch in viewer IP address – viewer IP address may change between the request for a token and request for video stream object as explained in the Access token management

guide section. In this case, consider excluding the IP address when creating the token, or include it conditionally only when it is probable that the viewer's IP address won't change (for example, connected TVs).

- Missing or different value of the viewer's country or region – for CloudFront Function, to retrieve geolocation information about the request, Origin request policy associated with cache behavior which includes token validation, must include appropriate geolocation headers.Refer to the Access token management guide section for more details.
- Missing or different value of the viewer's header – make sure that the headers included in token creation won't change during playback. For instance, when using **referer** header in the policy it is possible that the browser construct referer header differently when making request to the playback API, and when making a request to the CloudFront distribution, depending on Referrer Policy.

Auto session revocation

Auto session revocation module, when activated, regularly inspects CloudFront access logs with an objective to detect session IDs that exhibit suspicious traffic patterns in terms of request rate and composition of viewer attributes. The Session revocation guide section provides extensive overview of what factors are accounted for when determining the suspicion score and all the inputs and settings you have control of to impact the suspicion score formula and detection threshold. If playback session gets blocked unexpectedly, complete the following steps to work backwards and identify what factored in that decision.

1. Open DynamoDB table used as a data store for the session submitted for revocation. Table name can be found in the stack's CloudFormation output tab associate to the key which starts with **[Stack name]-SessionToRevoke**. Locate that name in DynamoDB console in correct region on the **Explore items** page.
2. Use Filters fields to look up the blocked session ID.

Filters

3. In the **Items returned** section an entry corresponding to this session will be displayed with a type attribute set to **AUTO**. Sum of **ip_penalty**, **ip_rate**, **referrer_penalty**, **ua_penalty** should equal to the final score attribute value which was above the threshold set.
4. You can further inspect underlying SQL query which produced this entry. Take a note of the timestamp in **last_updated** attribute which informs when the entry was submitted. You should expect the respective query to run shortly before that point in time.
5. Navigate to the Amazon Athena console and then to **Query editor** page.
6. Open **Recent queries** tab and look for the records run around the time indicated by the timestamp in DynamoDB table.
7. Click on the **Execution ID** attribute which will redirect you to the query editor populating the query that was run at the time. You can run and deconstruct this query to further analyse what drove specific components that added to the final score.

Contact Support

If you have AWS Developer Support, AWS Business Support, or AWS Enterprise Support, you can use the Support Center to get expert assistance with this solution. The following sections provide instructions.

Create case

1. Sign in to Support Center.
2. Choose **Create case**.

How can we help?

1. Choose **Technical**.
2. For **Service**, select **Solutions**.
3. For **Category**, select **Other Solutions**.
4. For **Severity**, select the option that best matches your use case.
5. When you enter the **Service**, **Category**, and **Severity**, the interface populates links to common troubleshooting questions. If you can't resolve your question with these links, choose **Next step: Additional information**.

Additional information

1. For **Subject**, enter text summarizing your question or issue.
2. For **Description**, describe the issue in detail.
3. Choose **Attach files**.
4. Attach the information that Support needs to process the request.

Help us resolve your case faster

1. Enter the requested information.
2. Choose **Next step: Solve now or contact us**.

Solve now or contact us

1. Review the **Solve now** solutions.
2. If you can't resolve your issue with these solutions, choose **Contact us**, enter the requested information, and choose **Submit**.

Uninstall the solution

You can uninstall the Secure Media Delivery at the Edge on AWS solution from the AWS Management Console or by using the AWS Command Line Interface.

Before you perform the steps to uninstall the Secure Media Delivery at the Edge solution, ensure you disassociate all the components created by the solution with the resources that are operated externally. In particular, review the configuration in:

- AWS WAF web ACL which includes the rule group for session revocation. Remove the corresponding rule from the web ACL list.
- CloudFront distributions' cache behaviors protected by secure tokens provided by this solution. Remove the association between all the viewer request triggers and function used to validate the token. You can see all the distributions and cache behaviors which use the function by navigating in CloudFront console to **Functions** page, selecting token validation function and switching to **Publish** tab which details all associations for this function.

Using the AWS Management Console

1. Sign in to the CloudFormation console.
2. On the **Stacks** page, select this solution's installation stack.
3. Choose **Delete**.

Using AWS Command Line Interface

Determine whether the AWS Command Line Interface (AWS CLI) is available in your environment. For installation instructions, refer to What Is the AWS Command Line Interface? in the *AWS CLI User Guide*. After confirming that the AWS CLI is available, run the following command.

```
aws cloudformation delete-stack --stack-name [Stack Name]
```

Using CDK toolkit

If you deployed the solution through CDK, depending on if you activated the Auto Session Revocation module, one or two stacks will be deployed in the region, named:

- **[Stack name]** – deploys components from Base and API module
- **[Stack name]AutoSessionRevocation –** includes components used in the Auto Session Revocation workflow

To remove both of the stacks run the following command from the project folder:

```
npx cdk destroy --all
```

Developer guide

This section provides the source code for the solution, a NodeJS library reference, an access tokens management guide, a session revocation guide, and playback API integration.

Source code

Visit our GitHub repository to download the source files for this solution and to share your customizations with others. The Secure Media Delivery templates are generated using the AWS CDK. Refer to the README.md file for additional information.

NodeJS library reference

The key step in the process of integrating the solution with your architecture involves incorporating token management methods with the services that you expose to the client's application directly. When you launch the API module an example API endpoint that acts as such service is created but in your actual implementation you may be tasked with adding token management operations into your workflow comprised of preexisting services. This solution comes with a NodeJS library, which facilitates this procedure by abstracting the token management operations that need to take place with code constructs that you can import in your own code. This section defines how the solution's NodeJS library has been structured, what classes and methods are made available for use and provide detailed reference of usage specific constructs included in the library.

On a high level

The library is provided as a single module that you can import with a single import command. From the module we expose all functionalities you interact with through classes. These classes are:

- Secret - to manage the keys used for signing and validating the tokens
- Token - to manage the token generation process
- Session - to manage viewer sessions when generating token and revoking the session

Secret

Object Properties

keys: object an object holding primary and secondary key pairs for each instance. Structure of this object must conform with the following format:

```
{
    'primary': {
      'uuid': primary_key_uuid: string,
      'value': primary_key_value: string
    },
    'secondary': {
      'uuid': secondary_key_key: string
      'value': secondary_key_value: string
    }
}
```

It is mandatory that any function used for retrieving signing keys return keys object in this exact form.

ttl: `[int]` → time to live expressed in seconds, detailing for how long keys should be considered as valid after they were retrieved. If the TTL expired, a new call to retrieve the keys will be invoked.

retrieveMode: `[string] = ['native|'custom'] (default 'native')` → determines what function is used to retrieve the keys. native refers to the internal method which interacts with Secrets Manager to retrieve the secrets storing primary and secondary keys.

retrieveFunction: `[function] (default null)` → a reference to the function provided by the user, that is called when key needs to be retrieved. It is only used when **retrieveMode** is set to `custom`. The custom function must return object of the same structure as specified in **keys** property.

retrieveFunctionArgs: `[array] (default [])` → stores positional arguments passed as input parameters when **retrieveFunction** is called.

stackName: `[string] (default: null)` → stores the name of the deployed stack. In native mode of retrieval, it is required to derive Secrets Manager secrets name, which is: `{stack_name}_PrimarySecret` and `{stack_name}_SecondarySecret`. stackName value binds the key retrieval process with the specific stack's secrets.

Constructor

Secret(stackName: string, ttl: int, retrieveMode: string = 'native', retrieveFunction: string = null, retrieveFunctionArgs: array = []) → constructor function maps the provided input to the object attributes

Instance Methods

getKeyValue(key_alias: string) *returns: string #* returns the value of the specified key alias, either `primary` or `secondary`

getKeyUUID(*key_alias: string*) *returns: string #* returns the UUID of the specified key alias, either primary or secondary

initSMClient({region: string, role: string, profile: string}) *returns: Boolean* → using provided inputs a new Secrets Manager client is created using the adequate credentials. Execution role will be derived as follows:

- If no role or profile was provided as an input, AWS SDK will determine which role to assume by following the standard process - it will look into environment variables, service execution role as described in Setting Credentials in Node.js.
- If profile attribute was provided – use the credentials associated with it.
- If no profile attribute was provided except iam_role- assume iam_role through AWS Security Token Service (AWS STS).

If the Region was not specified in the input object, AWS SDK provided method is used to verify what is the default region for the active profile. If none is specified, we default to `us-east-1`. This function is used when **retrieveMode** is set to `native`. Returns `true` if the client set up is successful, or `false` if it fails.

Session

Class Properties

revocationTable: `[string] (default: '')` → name of DynamoDB table created in the solution to store list of the sessions submitted for revocation.

Class Methods

Session.initialize(revocationTable ,{region: string, role: string, profile: string}) *returns: boolean* → sets revocationTable property referencing DynamoDB table for storing compromised sessions. Second parameter is an object with the parameters used when a new DynamoDB client is created

using the adequate credentials. Execution role will be derived in the same way as **initSMClient** in secret class.

Object Properties

id: string → mandatory attribute that must be set when creating the object. It is case sensitive. It is a session identifier issued for a playback session and tied to a specific access token.

suspicion_score: integer → considered when revocation list is compiled. As revocation list can hold only limited number of revoked sessions, the ones with highest score will be added within that limit.

Constructor

Session(id: string = null, autogenerate: boolean = false, suspicion_score: int = 0) → constructor function maps the provided inputs (ID, suspicion_score) to the object attributes. If autogenerate = true, ID will be casted to integer and ID of that length will be autogenerated. If ID is not provided, session ID of 12 characters length will be autogenerated.

Instance Methods

revoke(ttl: number = 86400, reason: string = 'COMPROMISED') *returns boolean* → pushes session to the **revocationTable**. It inserts an entry with the following properties and returns **True** after the session is successfully added to DynamoDB.

- session_id = session's identifier which equals to ID property of the object
- last_updated = current timestamp (second level precision)
- TTL = time period in seconds after which session entry in DynamoDB will be considered expired and removed
- type = 'MANUAL'
- reason = as per function input

To use that method, DynamoDB client must be initialized first through the **Session.initialize** class method.

Token

The main objective of token class is to generate unique tokens for the playback session. There are two main inputs of expected structure that determine the resulting token: **viewerAttributes** and

tokenPolicy. **viewerAttributes** details the viewer specific inputs, which are the basis for calculating internal signature unique to that viewer. **tokenPolicy** defines a set of parameters which directly inform how the resulting token will be calculated by describing what attributes should be included in the token, what are the path patterns, expiry time and other relevant fields. Below are the examples that illustrate the structure of both objects that need to be supplied as an input:

Viewer Attributes - Example

```
{
"co": "GB",
"reg": "ENG",
"ip": "10.0.2.1",
"headers": {
"user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; ...)",
"referer": "https://example.com/video1",
"x-auth": "uhfsdguyfde438t27gsda",
},
"qs": {
"m": "123456",
"filter": "mobile"
},
"sessionId": 'k1a12app2z'
}
```

Parameter	Default
co	viewer's country as a two-letter code according to ISO-3166-1 alpha 2 standard
reg	region code that corresponds to viewer's region. The value should match with the region codes of first-level subdivision of the ISO 3166-2 category
ip	viewer's source IP address (either IPv4 or IPv6)
headers	set of request headers and their values that viewer must include in all the requests including the token

Parameter	Default
qs	set of query strings parameters and their values that viewer must include in all the requests including the token
sessionId	Explicit session ID value that would be associated to the token and signed. If you prefer **sessionId** to be auto-generated, omit this field

Token Policy - Example

```
{
"co": false,
"co_fallback": true,
"reg": false,
"reg_fallback": true,
"exc": [
"/tm/"
],
"exp": "+1h",
"headers": [
"user-agent",
"referer"
],
"querystrings": [
"m"
],
"ip": false,
"nbf": "1645000000",
"paths": [
"/v1/master/b3dae6b5505e7de150071442385b3f3817cfa640/",
"/v1/segment/b3dae6b5505e7de150071442385b3f3817cfa640/",
"/v1/manifest/b3dae6b5505e7de150071442385b3f3817cfa640/",
"/out/v1/df069e9593304746afc28eb0117f2dbd/"
],
"ssn": true,
"session_auto_generate": 12
}
```

Parameter	Default
co	a flag which determines whether viewer's country attribute should be included in the token
co_fallback	a flag instructing token validation function how to handle the request when country (**co**) is included in the token however it is not possible to verify viewer's country when cloudfront-viewer-country header is missing. When set to true, request will be allowed in this scenario. When set to false request is blocked when viewer's country can't be determined.
reg	a flag which determines whether viewer's country region attribute should be included in the token
reg_fallback	a flag instructing token validation function how to handle the request when country (**reg**) is included in the token however it is not possible to verify viewer's country region when cloudfront-viewer-country-region header is missing. When set to true, request will be allowed in this scenario. When set to false request is blocked when viewer's country can't be determined.
exc	List of the URL paths prefixes that should be excluded from the viewer attributes verificat ion step. For any request which URL path starts with one of the URL paths listed in this property, only token integrity check, **nbf** timestamp and **exp** timer will be verified. This field allows to specify the URL path prefixes

Parameter	Default
	that would be subject to soft verification where it is known or likely that that not all the conditions can be met but access to this content should be still allowed (for instance advertisement segments requests)
exp	**Mandatory attribute.** Value determining how expiry time is set in the token. It can either be set as a relative time reference to the current time expressed in hours (for example, `+1h`) or minutes (for example, `+30m`). Alternati vely, an absolute timestamp with second level precision can be set as well.
headers	List of viewer request headers subject to viewer attributes verification
querystrings	List of viewer request querystring parameters subject to viewer attributes verification
ip	a flag informing if source IP address should be included in the token
nbf	Timestamp indicating the time before which the access token must not be accepted for processing

Parameter	Default
paths	**Mandatory attribute** List of URL path prefixes the token can be used for. Unless viewer request matches with URL path included in exc property, URL must match with one of the paths in this property. The items on this list don't specify the full URL path but just prefixes which spans from the beginning up to an arbitrary position in the URL path. each item in this list must be a substring of the actual viewer request URL to evaluate as a match.
ssn	A flag indicating whether session ID should be one of the inputs for token signature
session_auto_generate	If **sessionId** attribute is not present in viewerAttributes object, a number set in this property will set the length of the session ID to be automatically generated

Object Properties

defaultTokenPolicy: [object] → stores the default token policy object used in case a specific token policy is not provided as in input when calling **generate** method.

secret: [Secret] → reference to the instance of the Secret class, that provides the signing keys for the token generation process.

Constructor

Token(secret: Secret, defaultTokenPolicy: object = null) → constructor function maps the provided inputs (id, suspicion_score) to the object attributes

Instance Methods

generate(viewer_attributes: object, playback_url: string = null, token_policy: object = self.defaultTokenPolicy, key_alias: string = 'primary') *returns string* → generate a token and

returns either as a string which represents secure access token. If playback_url is provided, the function returns modified URL with resulting string inserted. In that function Session class is used internally to generate **sessionId** in case it wasn't provided explicitly in viewer_attributes object as an input.

*How the session ID is derived if token policy **ssn** flag is set to true?* Order of precedence is as follows:

- explicit **sessionId** value from viewer_attributes input
- auto generated using session_auto_generate value from token policy: new Session(viewer_policy.session_auto_generate, true)
- auto generated with the default session length of 12 characters

Determining querystring parameter value used when generating a token.

When a querystring parameter is listed in querystrings property in token policy, the value of which is later used when calculating a signature related to viewer attributes, **generate** method looks up its value in a following order:

- playback_url input parameter when **generate** method is called
- viewer attributes querystring object

Example of token generation code

Below code snippet serves as an example to show the sequence of steps that needs to be reflected in the code to produce a token at the level of playback API endpoint. It does not include the logic required to obtain the information about the video assets or how to compose viewer attributes and token policy objects, which would be specific to your use case and integration with content management system. In the API module, Lambda functions are deployed and can be considered as a more complete reference example of how to implement complementary logic and combine it with the solution's library.

```
//Import library
const awsSMD = require("./aws-secure-media-delivery");

//create secret instance, parameters specify stack name, secret caching

//ttl, 'native' informs to use AWS SecretsManager client for key retrieval
let secret = new awsSMD.Secret('StackName',10,'native');
```

```
//initiate AWS Secrets Manager client associated with secret instance
//role refers to the solution created role, region specifies in which region
//signing keys are stored in Secrets Manager
secret.initSMClient({role:'arn:aws:iam::1234:role/stack_role', region: 'eu-west-2'})
//initiate instance of Token class and associate it with secret object
let token = new awsSMD.Token(secret);

async function vendToken(viewer_attributes, playback_url, token_policy){

//generate the token
let outputToken = await token.generate(viewer_attributes, playback_url, token_policy);
return outputToken;
}
```

Example of session revocation code

```
//Import library
const awsSMD = require("./aws-secure-media-delivery");

//initialize Session class attributes, first argument points to DynamoDB //table which
 stores session to be revoked
awsSMD.Session.initialize('StackName-SessionToRevoke1234',
 {role:'arn:aws:iam::1234:role/stack_role', region: 'eu-west-2'});

async function blockSession(sessionId){
  //initiate instance of Session class with explicit sessionId provided
  let revokeSession = new awsSMD.Session(sessionId);
  let result = await revokeSession.revoke(86400);
  return result;
}
```

Access tokens management guide

Varying token attributes

When the access token is created upon viewer request, its final structure is defined by the claims encapsulated in the token. This allows you to vary the parameters used in the token depending on the requested content or viewer type. This is possible because the structure of the token is not fixed and each token can carry a varying set of attributes. Token validation logic implemented in CloudFront Functions has been designed for that as from the set of claims included in JWT token

this validation logic at the edge will derive how to correctly process its payload. This quality can be helpful as you can set different tiers of token restrictiveness reflected in used set of attributes for each tier, for example:

- For viewers originating from countries with a wide geographical range, include region attribute (**reg**) on top of country (**co**) to reduce the risk of unauthorized reuse of the token
- When a playback session is initiated from the connected TV type of client, you can issue the token based on the viewer's IP address (**ip**) as it is unlikely it would change throughout the session
- If the player is capable of generating and repeating a unique session identifier as a custom header, include that header and its value in the viewer attribute and token policy objects
- For the most premium video assets add more headers in the token policy definition (user-agent, referer etc.) to make it more restrictive

The list of parameters that can be inserted in the token policy with description can be found in the NodeJS library reference.

By setting token parameters in a dynamic manner, you can better align the scope of the token for various viewer groups and assets type. In the API module of this solution, and you can test that ability by specifying separate token policies for each video asset inserted in the **[Stack Name]-ApiDemoTable**.

Choosing session duration time

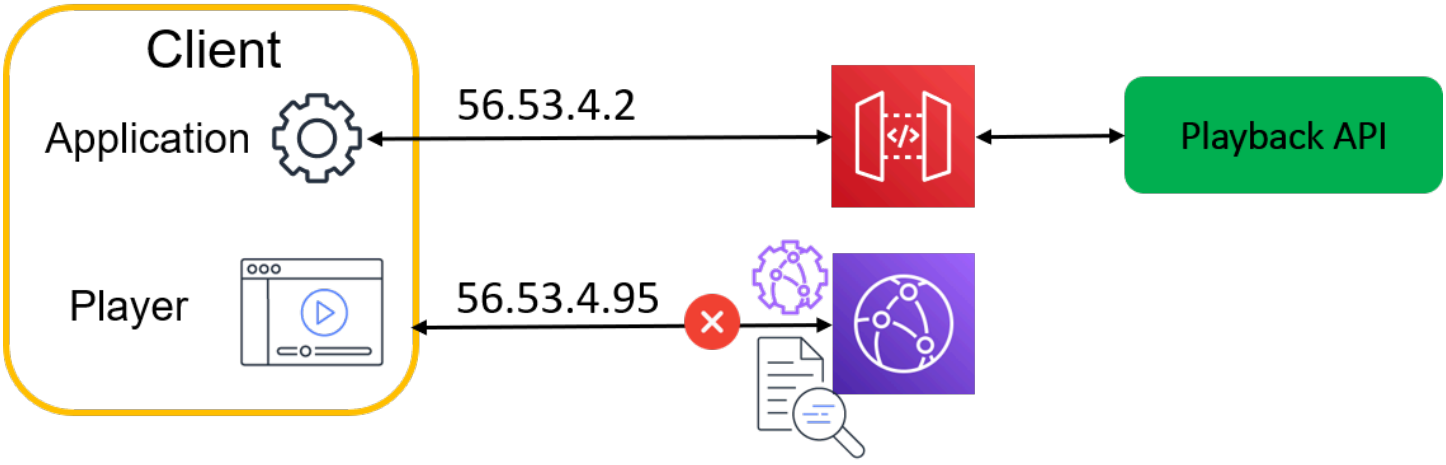
This solution works on the principle that entire playback session is protected by the use of a single token generated only once by the playback API when the playback URL with a token is passed to the player. The issued token is then repeated by the player and therefore it eliminates the need to recompute and refresh the token for the duration of the stream watched. However, when a viewer watches the stream, because the same token is used from the beginning until the end of the playback, it is important to set a correct expiry time to match the expected content duration when known, or a maximum period of time the viewer is expected to watch the same stream continuously (without switching to other channels or video assets). In the case of a stream scheduled with a specific start and end time, you can reflect these time boundaries in **nbf** and **exp** attributes in the token policy using precise timestamps values. For the continuous video streaming channels, when a viewer can start and stop watching the stream at any time, we recommend setting a reasonably high relative time reference in **exp** attribute which reflects typical audience behaviors for how long an uninterrupted, single streaming session can last. For instance, +4h would

issue a token that allows a 4 hours watch time. When a viewer goes past the expiry timestamp after 4 hours, further attempts to use the same token will fail and the client's application must refresh the stream by requesting a new token from the playback API.

Using viewer's source IP in the token

A viewer's source IP can be a very effective mean to reduce the scope of the token and tie it specifically to a single user, eliminating the risk of re-sharing the same playback URL link with the token. While using viewer IP as an input to produce the token gives very strong session uniqueness characteristics of the token, it is also prone to well-known problems that can interrupt a legitimate playback session. This is because a viewer's IP is not always persistent during the entire time of watching a video content. For example:

- When a mobile client is switching networks, the source IP changes
- Some clients are behind gateways or NATs which translate internal client's address into one of the many potential public addresses available in the pool. While a public IP assigned to the client for any new established connection stays the same for the same TCP session established, the problem arises when the public IP used when communicating with CloudFront distribution differs from the IP address that was used when contacting the playback API. That creates a mismatch between the IP used for generating the token and the IP used when interacting with CloudFront, as shown in the following diagram.



Source IP in the token

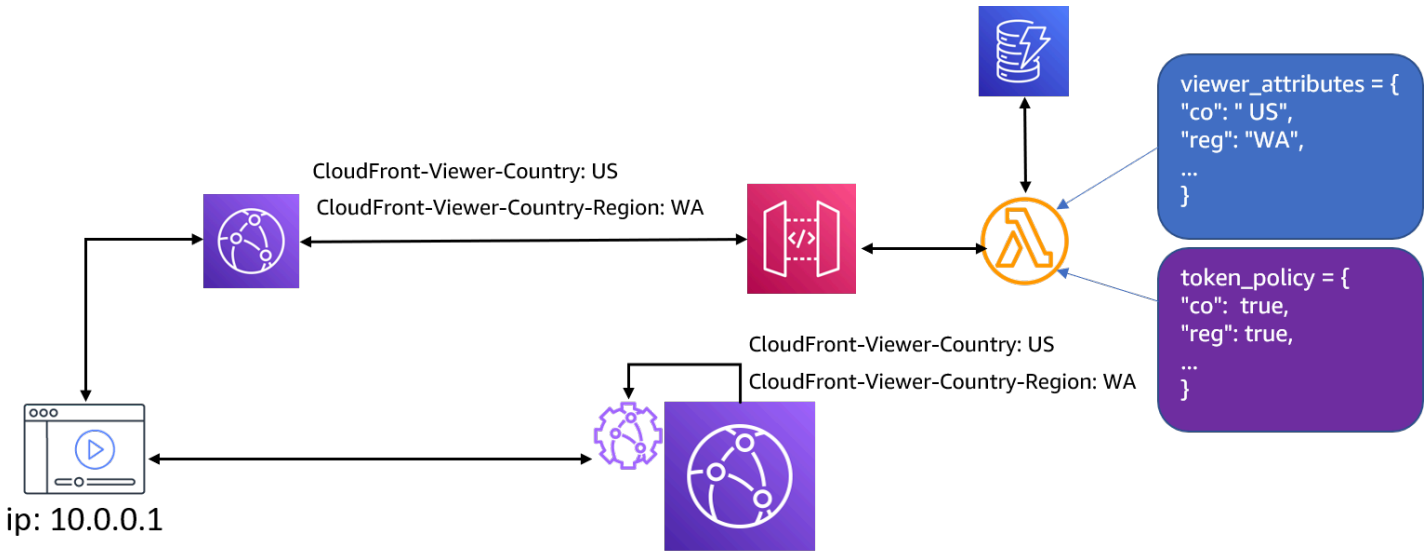
To avoid situations in which token validation fails due to a viewer's IP transitions, use viewer IP with caution when issuing the tokens. We recommend that you use viewer IP only when you have a high

level of confidence that the public IP used by the viewer is stable over time, for instance you can set **ip** claim to true (informing the token generate method to use viewer's IP) when you are able to confirm that client's device is a connected TV and if you assess that probability of IP change is low. CloudFront can facilitate making this type of decisions with CloudFront generated headers which specify viewer's device category.

Using geo restriction attributes

One of the categories of available token attributes is related to the geolocation of the viewer, which is verified by the token validator function at the edge. At the token generation stage, you can specify country and region attributes in viewer context object and set the corresponding flags **co** and **reg** to true in the token policy to factor in both when creating the token. Only the viewers whose source IP address is mapped to the same country and region by CloudFront would be permitted provided other matching conditions are also met. The viewer's country and region are determined by CloudFront and that information is exposed to CloudFront Function through CloudFront generated headers.

Bear in mind that CloudFront is able to map viewer's IP address to the geographic location details through an integration with a third-party database as explained in the documentation. If you are using your own geolocation database with a playback API to determine viewer's location, it is possible that for certain IP addresses, viewer's location details are not the same between these the two database sources. This misalignment can in turn result in blocking a legitimate viewer request if geolocation attributes are part of the token. To remove this effect, we recommend that you also retrieve viewer's geolocation mapping from CloudFront geolocation headers in the playback API request flow. By doing so, you use the same geolocation databases at the stage of producing the token and validating it. To obtain the geolocation information sourced from CloudFront in your playback API, you must set up a CloudFront distribution in front of your API endpoint and make the clients send the requests towards your API through CloudFront distribution, rather than sending request directly to your endpoint. In the configuration of CloudFront distribution facing API endpoint, create and attach the origin request policy to the corresponding cache behavior that includes CloudFront-Viewer-Country and CloudFront-Viewer-Country-Region as the headers to be forwarded to origin. The requests forwarded through CloudFront will contain viewer's geolocation details in these two headers that an API can parse and insert into the viewer attributes object. This design has been implemented in the solution's API module where API requests are made through CloudFront distribution which supplies upstream API endpoint with geo headers that can be used when creating the token.



Geo restriction attributes

In rare cases, CloudFront might not be able to determine viewer's location, therefore corresponding geolocation headers will not be present. If the geo restriction attributes **co** or **reg** were added to the token, there are respective fallback claims – **co_fallback** and **reg_fallback** flags in the token policy that will also be signalled in the resulting token payload. When set to true, these flags instruct the token validator function to skip the verification step of the viewers' attributes if viewer location cannot be concluded from the geolocation headers. In that case, the token validation process ignores viewer attributes which includes headers, querystring parameters, session ID, source IP and geolocation attributes, as all of them are verified in a single step. However, JWT token integrity is still validated as well as other attributes as detailed in the token, that is URL path match, expiry time, and nbf timestamp.

Defining paths list

One of the key and mandatory input parameters that must be set in the token policy is a list including path prefixes that the single output token will be covered during the playback session. Each path item in that list can be considered as having implicit, trailing wildcard. That means that if following path item is added in the path's property:

- `/video/1/`

  You can use the token for all the assets sharing that path structure, therefore if all the objects required for the playback are available in this directory or its subdirectories, this single path would be enough to restrict the token to that asset and provide uninterrupted playback. All the following requests which come with the token including that path will match this pattern:

- `/video/1/index.m3u8`
- `/video/1/uhd/index1.m3u8`
- `/video/1/index1_02.ts`

There are some cases however when URL path for the consecutive video objects vary to a high degree where it is impossible to formulate a single path prefix that would scope down the token to a single video asset while covering all the varying URL paths requested during playback. Example of this type of video workload could be AWS Elemental MediaTailor backed workflow, where Elemental MediaTailor becomes a proxy for manifest requests and abstracts directory structure relative to the original path of the manifest. At the same time, URL paths for the original video segment objects remains unchanged and additional ad segments objects are inserted into manifest with their own path patterns. As a result, in a single playback session with MediaTailor in place, you can identify URL paths similar to the following:

- `/v1/master/`**`1234abcd`**`/`**`Video-HLS`**`/index.m3u8` > master manifest
- `/v1/manifest/1234abcd/Video-HLS /xwyz-987654321/2.m3u8` > child manifest
- `/v1/segment/1234abcd/Video-HLS/xwyz-987654321/4/11223355` > ad tracking endpoint
- `/out/v1/df123456789/index_2_23456.ts?m=1646863308`
- `/tm/1234abcd/bebgiqk6y5etzpgwfswp6jqccyu7c7ve/asset_720_4_4_00007.ts`

Assuming all these objects are delivered through CloudFront distribution, to construct a single token that would match all these URL paths patterns but not be over permissive in terms of giving access to more than one video asset it was produced for, craft **path** parameter in the token policy object in a following way:

```
{
  ...

  "paths": [
    "/v1/master/1234abcd/Video-HLS/",
    "/v1/manifest/1234abcd/Video-HLS/",
    "/v1/segment/1234abcd/Video-HLS/",
    "/out/v1/df123456789/",
    "/tm/1234abcd"
  ],
   ...
```

```
}
```

Session revocation guide

For improving video streams protection, this solution relies on restricting the usage of the token by scoping down viewer specific attributes and video assets path by the relevant token claims. In an ideal situation, the generated token would work with a single video asset and more importantly grant access to a single individual the token was created for. To accomplish this type of strong uniqueness in token to viewer mapping requires selecting a right set of viewer attributes when creating the token, which in aggregation create a unique attributes combination not easily replicable. The weaker the level of viewer attributes uniqueness used in the token, the easier is to reuse the token by other viewers which would give them unauthorized access. Therefore, while it is recommended to include a number of attributes that increase uniqueness of resulting their sum it can also come at the price of false positives as explained in the Using viewer's source IP in the token section as an example. To better manage that tradeoff, this solution also provides an option to revoke playback sessions that were identified as compromised ones – meaning, shared with other viewers through unauthorized channels. If you decide to complement token-based protection with session revocation, think about what type of logic you can employ to discover and block suspicious traffic pattern.

SessionId and access token relation

When preparing the inputs required for issuing an access token, you can also attach a **sessionId** with that token, which makes a unique identifier of the playback session. The **sessionId** has to be present alongside the access token when viewer makes a request. While the access token is a JWT token that has all the necessary claims in the payload, subject to the verification by the CloudFront Function logic, **sessionId** value is not encapsulated in the payload itself but appended to the JWT token string at the beginning of the URL path. This approach optimizes the utilization of WAF rule group capacity available, as the session blocking rules created for compromised sessions are defined with specific URL path matching conditions consuming the least amount of WCU capacity (2 WCU units per each session blocking rule). JWT access token only holds a claim – **ssn**, assuming true or false value indicative if **sessionId** was attached with the token. If the claim is set to true any attempt to remove or modify the **sessionId** value in the URL path will result in an error upon token validation. This is because the **sessionId** associated with the token is used as one of the inputs when calculating viewer attributes specific signature.

Manual session revocation

The solution's library comes with a dedicated class that can be used to block any session ID that has been previously created with the use of **revoke** method as shown in the **Example of Session Revocation code**. You can submit block instructions against individual sessions from any environment provided that the role used to make API call towards DynamoDB has the right set of permissions to insert new entries into the table. When initiating **Session** class for the purpose of revoking individual session you also need to initialize it with the correct table name corresponding to the created stack. You can find that name in stacks' CloudFormation output tab. How you derive the session IDs that must be blocked is subject to your own processes and due diligence. Third party A/B watermarking solution can be a viable solution to provide this type of intel. Through forensic analysis of the redistributed stream using non-authorized channels you are able to identify the original source of that stream. If that stream is token protected with session ID included, you can create a mapping of the A/B watermark identifier with session ID at the time of generating playback URL and use that one-to-one correlation to respond with the blocking action against the source of the illegally redistributed stream. Manual session revocation also allows you to respond to link-sharing occurrences through publicly accessible forums and social media. Lastly, because the session ID is retrievable from the URL path, you can build your own detection mechanism responsible for tracking session level anomalies and pick on the patterns that you recognize as a signal for malicious activity.

Auto session revocation module

CDK deployment model offers additional module that can be deployed as part of the solution, which gives a reference for how to automate the process of detecting suspicious sessions based on the set of selectable criteria and threshold adequate for your workload. Solution overview describes the end-to-end workflow of this module which is built around running Athena SQL queries against CloudFront access logs to collect session level metrics used to calculate suspicion score. You must provide multiple inputs before you start using auto session revocation as shown in the table.

Value	Description
Frequency of running auto session revocation module	In the range of minutes, express how often auto session revocation workflow will be initiated.

Value	Description
Athena Database name	Database name Athena SQL query will run against when initiated.
Athena Table name	Table name Athena SQL query will run against when initiated.
Request IP column	Name of the column in table schema which stores viewer's IP.
UA column name	Name of the column in table schema which stores viewer's user-agent header value.
Referer column name	Name of the column in table schema which stores viewer's referrer header value.
URI column name	Name of the column in table schema which stores request URL path.
Status column name	Name of the column in table schema which indicates HTTP status code returned in the response returned to the request.
Response bytes column name	Name of the column in table schema which specifies response size in bytes sent to the viewer.
Data column name	nName of the column in table schema with representing the date when request was made.
Time column name	Name of the column in table schema representing the time when request was made.

Value	Description
Lookback period	Expressed in minutes, used to derive a time window for the log entries that will be considered for analysis. It spans from the current time back to the specific number of minutes as specified in this parameter.
IP penalty	Assumes true or false value. It informs whether the presence of multiple source IPs which used the same session ID should be considered as a suspicious factor.
IP rate	Assumes true or false value. It informs whether the request rate calculated for the session ID should be included in calculating suspicious factor.
Referer penalty	Assumes true or false value. It informs whether the presence of multiple referer headers which used the same session ID should be considered as a suspicious factor.
Multiple User-Agent penalty	Assumes true or false value. It informs whether the presence of multiple referer headers which used the same session ID should be considered as a suspicious factor.
Minimum sessions number	The minimal number of active playback sessions in the inspected time frame that must be met in order to produce any output.
Minimum session duration threshold	Expressed in minutes. A minimum period of time session must be active during analyzed time frame to be included for further analysis.

Value	Description
Score threshold	Suspicion score threshold. Any session that yields higher result will be included in the output and pushed to revocation table.
Partitioned access logs	True or false. Specify if your access logs are partitioned in S3 bucket per year, month, day and hour.

Each of these inputs is used in one of the stages auto session revocation workflows which works as follows:

Prerequisites

1. Turn on CloudFront access logs and start collecting them in the target S3 bucket.
2. Optionally, apply log processing pipeline to start partitioning log files to improve efficiency of running queries as described in Analyze your Amazon CloudFront access logs at scale. Note that the solution assumes predefined partitioning model, in which log files are segregated by year, month, day, hour. All these partition levels must be available. Importantly, respective column names must be set to following values: **year**, **month**, **day**, **hour**.
3. Create a database and the table which define table schema used when parsing and querying access logs. Take a note of **Athena Database name** and **Athena Table name.**
4. When following CDK deployment path, provide **Athena Database name** and **Athena Table name** from the previous step in the CDK configuration wizard.
5. If you implemented log partitioning mechanism in compliance with the mentioned requirements, set **Partitioned access logs** in the CDK configuration wizard to true.

After auto session revocation module is deployed with the **Frequency of running auto session revocation module** you defined in the wizard, a workflow running SQL query through Athena will be executed.

Narrowing down requests in scope

If you inspect SQL query processed by Athena, you will find multitude of filtering condition included. It is important step in the process to obtain high signal output by removing access log

entries which are irrelevant for outlier analysis and outside of the time boundaries that you want to investigate. For this reasons SQL query applies filtering criteria to scope down set of input log entries for further analysis based on:

1. Request timestamp – consider only the requests that were recorded within the **Lookback period** defining point in time (current time minus **Lookback period**) for which any requests that was registered before is not factored in the calculations
2. Returned response **Status** code – look only for the requests that were served with 200 or 206 responses
3. Ignoring the entries when **Response bytes** size was lower than 1KB.
4. Ruling out the sessions of very short duration within analyzed time frame (for instance newly started session), shorter than **Minimum session duration**
5. Validating if the number of active sessions is during inspected timeframe is greater than **Minimum sessions number**. If not, no results will be returned

Calculate session-level metrics

After limiting the scope of the requests represented as log entries as per predefined criteria, for each session a set of metrics will be calculated from the information contained the log entries:

- Request rate for each session relative to request rate measured at p50 against all the sessions. For this metric, request rate calculated for each session is a count of distinct source IP and request URL pairs for each session. This will keep the result consistent if client retrieves a single object by making multiple byte range requests. Session level request rate is normalized with p50 request value, output result of this metric should be approximately close to 1.0. The greater the value the more requests use the same session ID which suggests that the same session is shared by multiple viewers. This metric will be added to the final suspicion score if **IP Rate** parameter is set to true.
- Signal of multiple user-agents using the same session ID, based on user-agent header value found in **UA column.** This metric equals to two discrete values: 0 or 1, the latter one indicating presence of more than one user-agent values in the access logs. If **Multiple User-Agent penalty** is set to true, this value will be added to the suspicion score.
- Signal of multiple referer headers values using the same session ID, based on referer header value found in **Referer column name.** This metric equals to two discrete values: 0 or 1, the latter one indicating presence of more than one user-agent values in the access logs. If **Referer penalty** is set to true, this value will be added to the suspicion score.

- Signal of multiple source IPs using the same session ID, based on IP information included in **Request IP column.** This metric equals to two discrete values: 0 or 1, the latter one indicating presence of more than one viewer IP values in the access logs. If **IP penalty** is set to true, this value will be added to the suspicion score.

With all the metrics calculated for each of the of the session that was left in the scope of analysis, the final suspicion score is derived as a sum of all the components mentioned:

Suspicion Score = Request IP rate + IP penalty + Referrer penalty + UA penalty

The last three components can only have a discrete value of either `0` or 1 each. Request IP rate is linear value that can vary from `0` and is not bounded by upper value. A typical score for a session should stay in close proximity of `1.0` as if session is used by a single viewer all the penalty factors should equal to `0` and Request IP rate should be close to `1.0` as some variance from median (p50 value) is possible. At times, score value can be elevated even when used by a single user. That can occur for a new playback session when player fills the buffer and would request multiple segments in a short period of time, or when viewer switches the network which would set the IP penalty factor to 1.

Having suspicion scores in place for each session, the final output list includes only the ones exceeding **Score threshold.** The list is supplied to the Lambda function in the Step Functions workflow, which sorts and prioritizes the sessions that will be eventually placed in the WAF rule group responsible for blocking compromised sessions. Refer to the Architecture details section for more details.

Note that the default and suggested value of **Score threshold** is set to `2.2` which accounts for the momentary increase in the score that stems from the situations mentioned before. We recommend that you keep track of the resulting suspicion scores calculated during the tests for a representative viewership and content, and validate whether majority of the sessions yield the score within the threshold you have set.

Normalized IP request rate

One of the primary components that can impact the suspicion score for all the sessions is **Request IP rate** as this component value is not bounded and is relative to the median value calculated for all the session in the analyzed score. Because the calculation of this metric strongly depends on the normalization factor, it is important that this factor is stable and cannot be easily distorted by a few sessions that have been compromised, even to a large extent. To achieve the desired stability and prevent fluctuation of that factor in the presence of one or few sessions which have been

compromised on a significant scale, 50th percentile (p50) measure is used instead of the average as a normalization factor.

Playback API integration

The Secure Media Delivery at the Edge on AWS solution provides a reference architecture which encompasses the entire process of managing secure access tokens in the video streaming workload. The primary and universal components of the solution are defined in the base module which governs the functionality of token validation and signing key management. This part of the solution remains unchanged as you implement your solution across your workloads and it is not expected that this part of the workflow, responsible for token validation, would require any customization to watch varying video streaming workloads. On the other hand, the API module is offered as a reference implementation of the initial stage of the end-to-end workflow which entails Playback API service – an endpoint that is responsible for vending the secure tokens to authorized viewers and returning playback URLs to the viewers. This module also comes with a demo website that allows you to test and validate the usage of the solution but it is not meant to use in production environments. As this solution provides incremental security layer to existing video streaming workloads, depending on your current implementation of a Playback API endpoint you can decide the best approach to customize and integrate the elements available in the solution to introduce the token generation function into your existing workflow.

Before you implement the most preferable integration option, make sure that before using the solution in production you turn off and remove the artifacts that the demo website is built on.

Deactivating demo website

By redeploying CDK stack

If you deployed the solution through CDK, you can simply deactivate the demo website and its related components by modifying the configuration file with the parameters that determine what solution components are deployed.

1. Navigate to the CDK project folder you used to configure and launch the solution in your account.
2. Edit `solution.context.json` document and within **api** property change the **demo** value to `false` and save:

```
{
 "main": {
```

```
  ...
 },
 "api": {
   "language": "nodejs",
   "demo": false
 }
 ...
}
```

3. Redeploy CDK stack.

```
npx cdk deploy [stack_name]
```

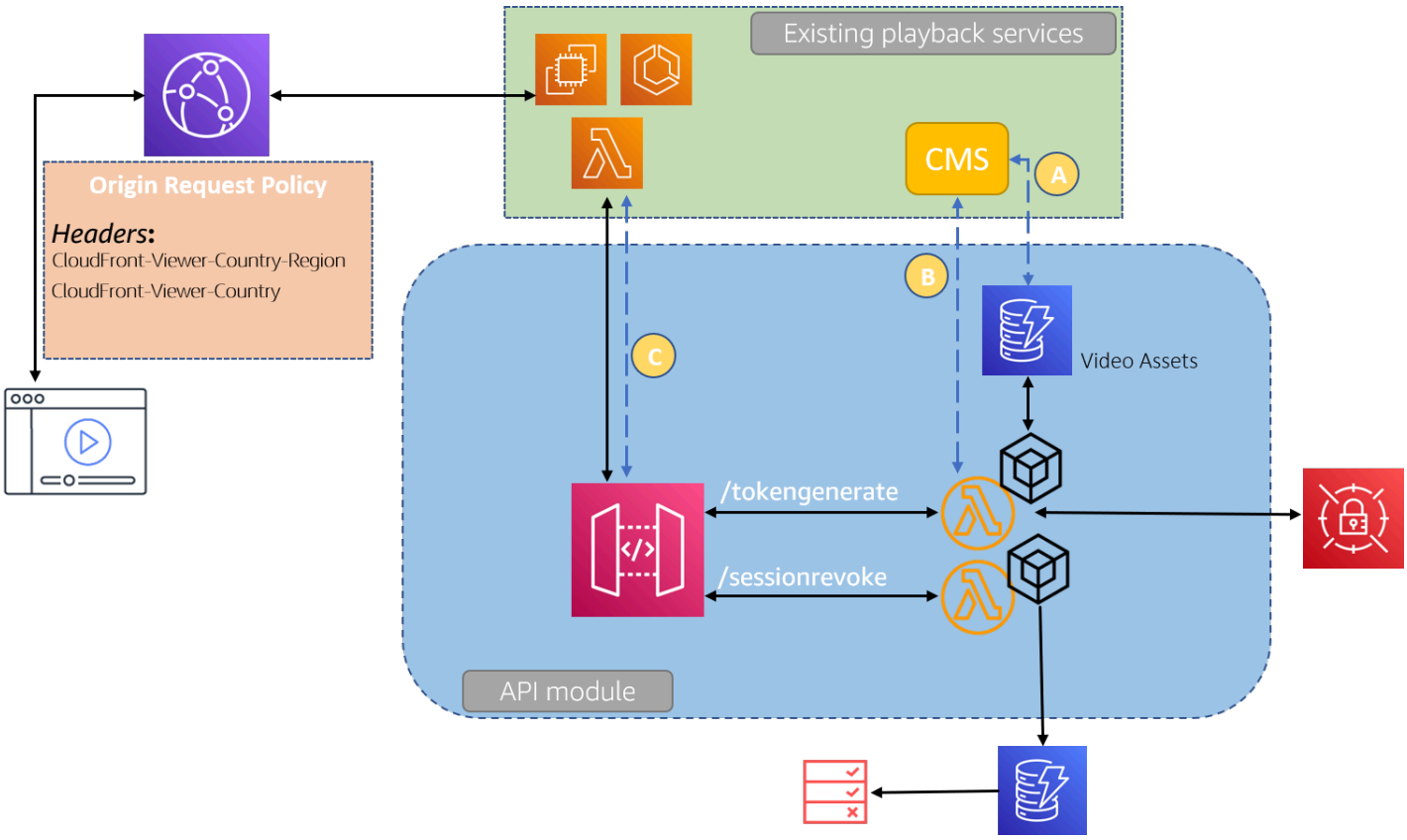
By deactivating CloudFront distribution

If you deployed this solution using the CloudFormation template, the demo website will be automatically published with the assets stored in a dedicated S3 bucket delivered through CloudFront. You can shut down the website by simply deactivating the CloudFront distribution which is the only entry point for the website as the S3 bucket is private.

1. In AWS Management Console, navigate to the CloudFront page.
2. From the list of all the CloudFront distributions associated with your account, select the one that was created with the solution stack with the description that matches with **[Stack Name]-Demo website Secure Media Delivery,** and choose **Disable**.
3. After few minutes, the **status** of the distribution changes from `Enabled` to `Disabled` making it effectively unreachable.

Reuse and modify solution's API Gateway workflow

After deactivating the demo website, you can retain the API Gateway endpoints and their integrations with Lambda functions and have your own services, which interact directly with the client applications, to treat that API Gateway endpoint as an internal token service. The security of interfacing API Gateway privately relies on IAM authorization as IAM authorizer is turned on by default in API Gateway configuration fronting Lambda functions that generate the tokens, and revoke the sessions. Another aspect is integrating your CMS with the Lambda function for token generation as you must provide the inputs that define token policy for a given video asset, unless the token policy is static and same token policy template can be used. The diagram below depicts the integration model:



API Gateway workflow: Reuse and modify

To integrate your existing playback services with the API module provided in the solution, review the following steps:

1. Configure your playback services to include additional step of retrieving the token before serving playback URL back to the viewer and revoking compromised sessions. This is done through REST API calls for the API Gateway towards `/tokengenerate` and `/sessionrevoke` paths respectively. Make sure that your services communicating with API Gateway have the adequate IAM permission to invoke mentioned resource paths for the API created by the solution. You can find API ID in the CloudFormation output tab under the key which starts with: **ApiEndpointsApiEndpoint**. The API ID is a first fragment of the API hostname visible for that output in the form of `[API_id].execute-api.[region].amazonaws.com`. Example policy statement which can be used when calling API:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

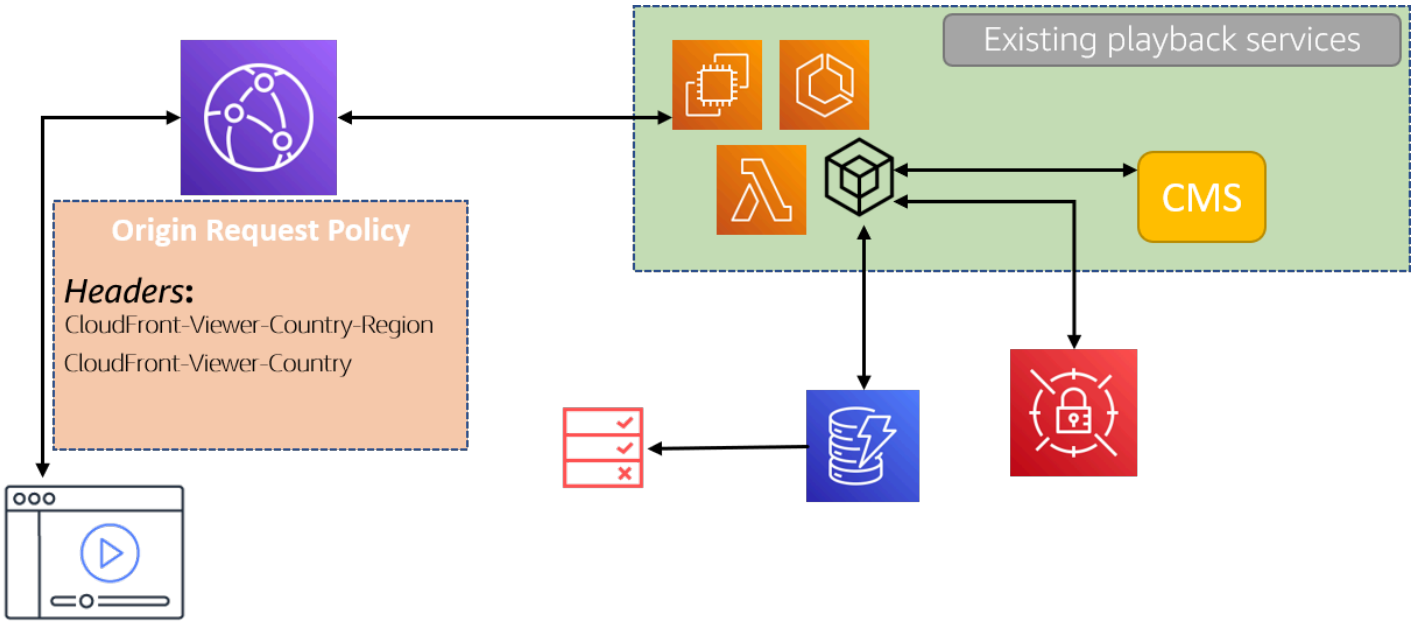
```
        {
            "Effect": "Allow",
            "Action": [
                "execute-api:Invoke"
            ],
            "Resource": [
                "arn:aws:execute-api:us-east-1:111122223333:api-id/*/
GET/pets/a1b2"
            ]
        }
    ]
}
```

2. If you want to use the geolocation (country or region) when issuing the token and want to obtain viewer's geo attributes from CloudFront (as recommended in the Access Token Management Guide section), use CloudFront distribution in front of your playback services with **CloudFront-Viewer-Country** and **CloudFront-Viewer-Region** headers included in origin request policy. Next, pass these headers when calling the token API endpoints from API module. Similarly, all the header values and query string parameters that originally come from the viewer's request and you want to incorporate in the scope of the token, must be forwarded to API Gateway in the same form as they were received by your playback services, so that Lambda function can access them accordingly.
3. Expose the metadata managed with your services to the Lambda function attached to API Gateway. At minimum, when Lambda function starts token generation process, token policy must be fetched from the metadata source as one of the inputs. Video asset hostname and URL path are also required to issue full playback URL, however if your metadata source only provide the token policy but no video asset's hostname and URL path, only the token will be returned as Lambda output. You can explore the following metadata sharing options with in order to supply token policy and video asset's location (as depicted on the previous diagram):
    a. **Scenario A** - Create an integration between your CMS and DynamoDB table created under API Module, storing token policies and video assets metadata. In this approach, CMS must push the new table entries in the right format which Lambda function is able to parse. See existing items in the DynamoDB table to determine the expected structure for the new items. .
    b. **Scenario B** - Customize Lambda function used for token generation to retrieve the required information from your CMS directly. This approach involves code changes in the Lambda function to be able to fetch necessary information, and subsequently parse it to formulate viewer context and token policy objects required to generate the token.

c. **Scenario C** - Provide the inputs as additional parameters when making API call. You can modify Lambda function to parse the incoming request and look for the token policy and video asset metadata in the query string parameters of that request. Apply necessary code changes into the Lambda function to include this processing logic and specify which query string attributes would be inspected accordingly.

Integrate solution's library into existing playback services

Another option to add the token generation step into your workflow is to include the solutions' library into your existing playback services. The library can be used in NodeJS runtimes and is available in the solution's source code repository. There are no specific requirements or restrictions as to where you should be running your playback API services. Solution's library contains a set of constructs and methods that interact directly with the specific components created in the base module of the solution. These components provide necessary data for the library to work (like token signing keys) and also accepts the inputs originated from the library calls (like information about session ID to be revoked). Conceptually, this is illustrated in the following diagram.



Integrate solution library into existing playback service

Integrating the library into generic environment

When integrating the solution's library in your own generic compute environment in which you manage the application stack, complete the following steps to prepare your environment:

1. Download the solutions' library from the solution's source code repository available under the `/source/resources/sdk/node/v1/` path.
2. Copy library's files `aws-secure-media-delivery.js` and `package.json` into the `aws-secure-media-delivery` folder in your Node.js project folder
3. Include the library in the `package.json` as a project's dependency:

   ```
   "dependencies": {
       ...
       "aws-secure-media-delivery": "file:./aws-secure-media-delivery/"
   },
   ```

   And, install the dependencies running `npm install` command.

   In case the project has been already initialized and you want to add library to and existing project, run `npm install –save ./aws-secure-media-delivery`
4. Import the library in your code:

   ```
   awsSDM = require('aws-secure-media-delivery');
   ```

Use the library's methods and constructs as instructed in NodeJS library reference to generate the token and revoke the sessions.

Integrating the library into Lambda functions

If your existing playback API services are implemented in the AWS environment with the use of Lambda services and you plan to integrate the solution's library into an existing or new Lambda function, you can simplify the process of installing solution's library into the NodeJS runtime by leveraging Lambda Layer created by the API module, which already includes the library.

1. In your Lambda function configuration, under **Layers** select **Add a layer.**
2. From **Choose a layer,** select **Custom layers**.
3. From the **Custom layers** list, select the one which starts with **APIGenerateTokeyLayer**, largest version number from the **Version** selector, and choose **Add**.
4. After adding the layer, you can import the solution's library directly in your Lambda function code with:

```
awsSDM = require('aws-secure-media-delivery');
```

Configuring library with the stack

After you have successfully added the solution's library into your environment and imported it into your code, you must provide references to the solution stack that you launched in your AWS account. To allow the library methods to communicate and interact with the solution components deployed from the solution stack, look up the references of these component and provide them when utilizing library methods in your environment. Below are the references you can find in the output tab of the launched solution's stack in CloudFormation configuration:

- **Stack Name** – used in Secret class to derive the secrets names in Secrets Manager, where signing keys are stored. Stack name must be provided as an input parameter for the Secret's constructor.
- **SecretsPrimarySecret** and **SecretsSecondarySecret** – if you decide to use custom function for retrieving the signing keys from Secrets Manager the values corresponding to these outputs are secrets' identifiers you can reference when making API calls to Secrets Manager.
- **SessionRevoke** – a DynamoDB table name which is an entry point for submitting session IDs identified as suspicious one that should be processed for blocking. The name of that table has to be provided as an input parameter for the **initialize** method of Session class.

Importantly, because solutions' library interacts directly with AWS components which comprise the solution architecture, IAM permission model applies. When underlying AWS SDK methods are called from the library, there need to be AWS credentials set for the service clients to be able to initiate necessary API calls. Therefore, when running any process that runs the solution's library code, you must make sure that the right AWS credentials are provided. Depending on the integration model, you have multiple options to ensure the right set of permissions are in use.

When using Lambda function

The common way for granting the right set of permission for a Lambda function which make API calls to other AWS services is to adjust **Execution role permissions** accordingly. If you are unsure and looking for minimal set of permissions for your Lambda function to work with the other elements of the stack, you can copy and include in execution role configuration the same policy as the one attached to the dedicated library's role. You can find the role's identifier in CloudFormation output under **RoleARN** key. Navigate to that role definition in IAM settings console and in the **Roles** page, select the role and you will find the policy under the **Permission Policy** tab.

When using solution's library in any environment

If you run your playback API services in any generic environment, to be able to use the solution's library you need to AWS Credentials available in this environment that AWS SDK underpinning solution's library can assume. For more information about how to manage AWS credentials and profiles in your environment, refer to Setting Credentials in Node.js.

Aside from the standard methods of supplying AWS credentials into NodeJS runtime as outlined in the documentation, the solution's library also offers the ability to overwrite the permissions assumed in a standard way. When calling the library's method that initiate service clients for AWS Secrets Manager and DynamoDB table, you can supply an input object in which you can reference specific profile, role, and region used by the underlying client created specifically for library's operations. The object has following structure:

```
{
  region: [Region for the target service endpoint],
  profile: [Name of the profile, defined in local environment to be used],
  role: [ARN of the role to be assumed through AWS STS]
}
```

When both profile and role properties are provided, profile takes precedence. Note, that in order to assume the role with the ARN specified, the default set of AWS credentials retrievable from the execution environment needs map to the right set of permissions allowing to perform **sts:AssumeRole** action against the referenced role. We recommend that when you decide to use the role, that you reference the role created when solution is deployed. You can find the created role ARN identifier in CloudFormation output tab of the deployed stack under RoleARN key.

There are two methods in the solution's library in which you can provide this object to override the role and target region assumed by AWS SDK through the standard procedure:

initSMClient({region: string, role: string, profile: string}) – in Secrets class as an instance method. It initiates Secrets Manager client to retrieve the signing keys.

Session.initialize(revocationTable ,{region: string, role: string, profile: string}) – in Session class, a class method used to create DynamoDB client which sends the details of the sessions to be revoked.

Reference

This section includes information about an optional feature for collecting unique metrics for this solution, pointers to related resources, and a list of builders who contributed to this solution.

Anonymized data collection

This solution includes an option to send anonymized operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When invoked, the following information is collected and sent to AWS:

- **Solution ID** - The AWS solution identifier
- **Unique ID (UUID)** - Randomly generated, unique identifier for each Secure Media Delivery at the Edge on AWS deployment
- **Timestamp** - Data-collection timestamp

AWS owns the data gathered though this survey. Data collection is subject to the AWS Privacy Notice. To opt out of this feature, complete the following steps before launching the AWS CloudFormation template.

1. Download the AWS CloudFormation template to your local hard drive.
2. Open the AWS CloudFormation template with a text editor.
3. Modify the AWS CloudFormation template. Find the Lambda resources definitions with Environment subsection including METRICS parameter and change it from `true`:

```
Type: AWS::Lambda::Function
...
Environment:
    ...
    METRICS: "true"
```

to `false`:

```
Type: AWS::Lambda::Function
...
Environment:
```

```
    ...
    METRICS: "false"
```

4. Sign in to the AWS CloudFormation console.
5. Select **Create stack**.
6. On the **Create stack** page, **Specify template** section, select **Upload a template file**.
7. Under **Upload a template file**, select **Choose file** and select the edited template from your local drive.
8. Choose **Next** and follow the steps in Launch the stack in the Automated Deployment section of this guide.

If the solution was deployed through CDK, complete the following steps.

1. Navigate to the CDK project folder where the solution was deployed from
2. Open `solution.context.json` file which was created by the solution wizard
3. Edit the file and in the main section of the file, change the **metrics** value from `true`:

```
{
  "main": {
    "stack_name": "SecureMediaDeliveryStack",
    "wcu": "200",
    "retention": "15",
    ...
    "metrics": true
  },
...
}
```

to `false`:

```
{
  "main": {
    "stack_name": "SecureMediaDeliveryStack",
    "wcu": "200",
    "retention": "15",
    ...
    "metrics": false
  },
...
```

```
}
```

Finally, deploy the new stack:

```
npx cdk deploy --all
```

Related resources

The Live Streaming on AWS is a solution designed to create a fully functional video streaming deployment based on Amazon CloudFront for end user delivery. This solution complements the Secure Media Delivery at the Edge on AWS solution, as it provides necessary components that the solution can be integrated with.

Contributors

- Kamil Bogacz
- Corneliu Croitoru
- John Councilman
- Eddie Goynes
- San Dim Ciin
- David Chung
- Raul Marquez

Revisions

Date	Change
August 2022	Initial release
September 2022	Release v1.0.1: Bug fixes. For more informati on, refer to the CHANGELOG.md file in the GitHub repository.
October 2022	Release v1.1.0: For more information, refer to the CHANGELOG.md file in the GitHub repository.
February 2023	Release v1.1.1: Bug fixes. For more informati on, refer to the CHANGELOG.md file in the GitHub repository.
April 2023	Release v1.1.2: Mitigated impact caused by new default settings for S3 Object Ownership (ACLs disabled) for all new S3 buckets. For more information, refer to the CHANGELOG .md file in the GitHub repository.
May 2023	Release v1.1.3: Bug fix to address issues with demo player. For more information, refer to the CHANGELOG.md file in the GitHub repository.
June 2023	Release v1.1.4: Updated aws-lib-cdk package, which upgrades Node.js to 16 for a custom resource. For more information, refer to the CHANGELOG.md file in the GitHub repository.
June 2023	Release v1.2.0: Added integration with Service Catalog AppRegistry and AWS Systems Manager Application Manager. Upgraded to Node.js to 18 and JS SDK to v3. Updated

Date	Change
	parameter names for consistency. For more information, refer to the CHANGELOG.md file in the GitHub repository.
August 2023	Documentation update: Built out additional examples in the Cost section tables.
September 2023	Release v1.2.1: Fixed an issue where customers were not able to deploy the solution with cdk deploy command. For more information, refer to the CHANGELOG.md file in the GitHub repository.
November 2023	Release v1.2.2: Updated package versions to resolve security vulnerabilities. For more information, refer to the CHANGELOG.md file in the GitHub repository.
November 2023	Documentation update: Added Confirm cost tags associated with the solution to the Monitoring the solution with AWS Service Catalog AppRegistry section.
June 2024	Release v1.2.3: Updated CDK to version 2.143.0 and updated other libraries. For more information, refer to the CHANGELOG.md file in the GitHub repository.
August 2024	Release v.1.2.4: Updated package versions to resolve security vulnerabilities. For more information, refer to the CHANGELOG.md file in the GitHub repository.
September 2024	Release v.1.2.5: Updated package versions to resolve security vulnerabilities. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Date	Change
November 2024	Release v.1.2.6: Updated package versions to resolve security vulnerabilities. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents AWS current product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. AWS responsibilities and liabilities to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.