Media Services Application Mapper

AWS Implementation Guide

Tom Gilman

Jim Thario

December 2018

Last updated: January 2020 (see revisions)

Contents

Overview ... 3

Cost ... 3

Architecture Overview ... 4

Solution Components ... 4

Solution Microservices ... 5

Core API Microservice ... 5

Connection Discovery Microservice ... 5

Inventory Discovery Microservice ... 5

Alarm State Microservice ... 5

Event Handler Microservice ... 6

Amazon DynamoDB ... 6

Read and Write Capacity ... 6

DynamoDB Auto Scaling ... 6

Browser Application ... 6

Channel Tiles ... 7

Design Considerations ... 7

Customization ... 7

AWS CloudFormation Templates ... 8

Automated Deployment ... 8

What We'll Cover ... 8

Step 1. Launch the DynamoDB Template ... 9

Step 2. Launch the Core Template ... 10

Step 3. Launch the Events Template ... 12

Step 4. Launch the Browser App Template ... 13

Step 5. Configure the Browser App ... 14

Security ... 15

Additional Resources ... 15

Source Code 16

Document Revisions 16

About This Guide

This implementation guide discusses architectural considerations and configuration steps for deploying the Media Services Application Mapper on the Amazon Web Services (AWS) Cloud. It includes links to a AWS CloudFormation template that launches, configures, and runs the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, administrators, and DevOps professionals who have practical experience with media workflows and architecting on the AWS Cloud.

Overview

Amazon Web Services (AWS) offers several media services that make it fast and easy to prepare, process, and deliver broadcast and over-the-top video from the AWS Cloud. These pay-as-you-go services replace costly, hardware-based video infrastructure to deliver great viewing experiences on multiple screens. With AWS, you can innovate, test, and deploy video services without spending a lot of time or money to procure and integrate technology.

However, it can be a challenge to understand the workflow relationships between different media services and identify the root cause of a problem when multiple media services are sending messages to Amazon CloudWatch, a monitoring and management service.

To help customers more easily visualize media service relationships and the real-time status of linear video services, AWS offers the Media Services Application Mapper. This solution enables customers to display the logical connections between media services, visualize error messages and counts, and produce a list of confidence-ranked root causes for problematic workflows. The solution currently monitors AWS Elemental MediaLive, AWS Elemental MediaPackage, Amazon CloudFront, and Amazon Simple Storage Service.

Cost

You are responsible for the cost of the AWS services used while running this solution. The total cost of this solution depends on your resource inventory and activity. As of the date of publication, the cost for running this solution with default settings in the US East (N. Virginia) Region is approximately **$1.50 per day.** This does not include variable charges for

Amazon DynamoDB Auto Scaling for larger inventories. Prices are subject to change. For full details, see the pricing webpage for each AWS service you will be using in this solution.

Architecture Overview

Deploying this solution builds the following environment in the AWS Cloud.

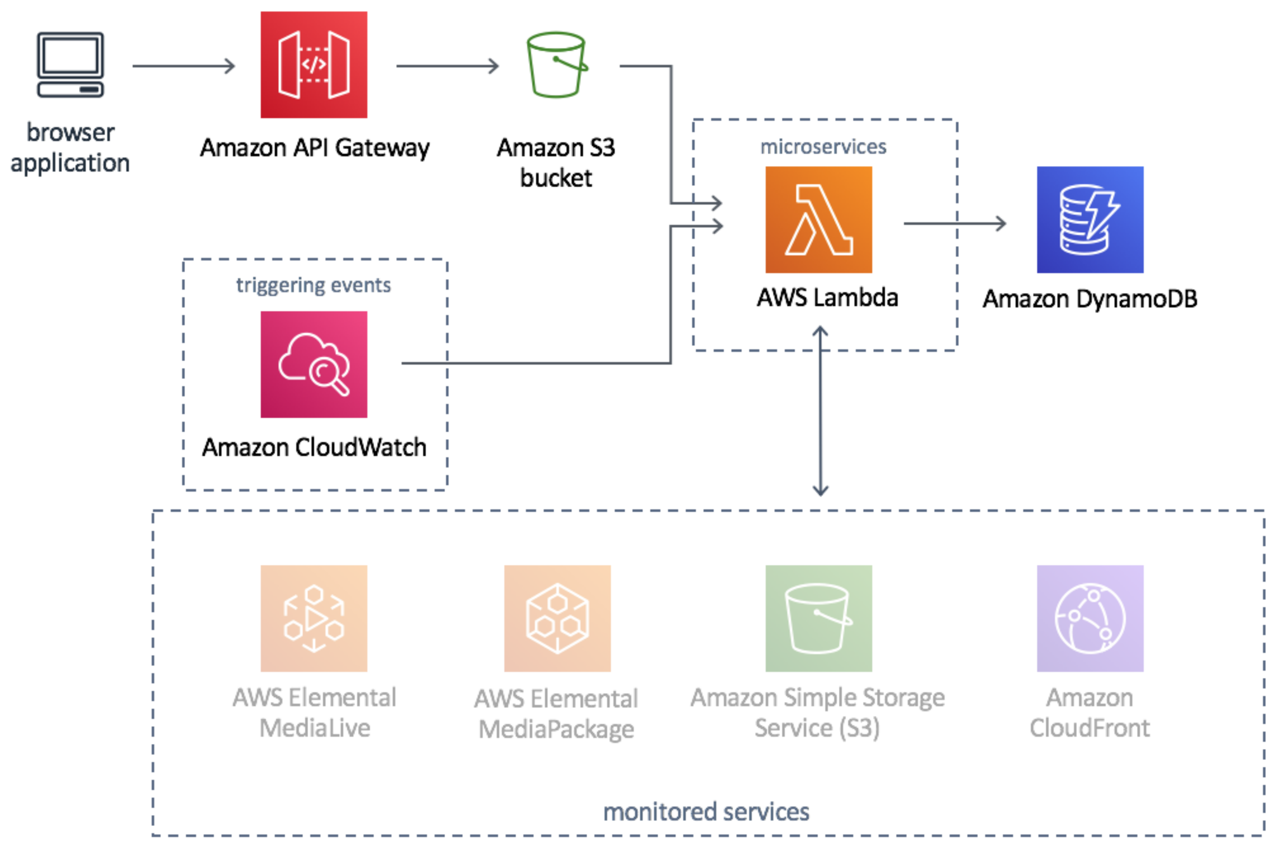


Figure 1: Media Services Application Mapper architecture on AWS

This solution includes an AWS CloudFormation template that creates a browser application and deploys it to an Amazon Simple Storage Service (Amazon S3) bucket. The browser application visualizes media service relationships and the real-time status of linear video services. The solution also deploys an Amazon API Gateway to host the solution's RESTful APIs, Amazon CloudWatch to monitor your media services for changes and errors, a set of microservices (AWS Lambda functions) that manage your inventory, connections, alarms, and events, and Amazon DynamoDB tables that store all solution-related data.

Solution Components

Solution Microservices

The Media Services Application Mapper microservices are a set of AWS Lambda functions that provide the business logic and data access layer for all solution operations. Existing applications can interact with solution data securely through the RESTful APIs to perform activities such as preloading channel tile definitions or adding custom content to the cache.

Each Lambda function assumes an AWS Identity and Access Management (IAM) role with least-privilege access to perform its designated functions. The following sections outline each microservice.

Core API Microservice

The `core API` Lambda function handles all browser requests and displays information from the solution's Amazon DynamoDB tables in the browser application. The `core API` Lambda function also handles all administrative services including layout management and general settings.

Connection Discovery Microservice

An Amazon CloudWatch event invokes the `connection discovery` Lambda function every two minutes. The function looks for connections between your media services, and stores any connections it could determine in the `content` DynamoDB table. When a user accesses the browser application, the `core API` Lambda function retrieves the connection information from the `content` table and displays it in the browser. Resources that do not have connections are grouped by service and displayed in the browser. For example, Amazon Simple Storage Service (Amazon S3) buckets that do not have connections will be displayed together in the browser.

Inventory Discovery Microservice

An Amazon CloudWatch event invokes the `inventory discovery` Lambda function every two minutes. The function looks for your media services resources, and stores information about the resources in the `content` DynamoDB table. When a user accesses the browser application, the `core API` Lambda function retrieves the resource information from the `content` table and displays the resources and their connections in the browser.

Alarm State Microservice

An Amazon CloudWatch event invokes the `alarm state` Lambda function every minute. The function looks for resources that have generated an alarm and adds the information to the `alarms` DynamoDB table. When a user accesses the browser application, the `core API` Lambda function retrieves the alarm information from the `alarms` table and displays it in the browser.

Event Handler Microservice

Amazon CloudWatch Rules receives an event when there is a change in your media services or their status and routes the event to the `event handler` Lambda function. The function adds the change to the `events` DynamoDB table. When a user accesses the browser application, the `core API` Lambda function retrieves the updated information from the `events` table.

Amazon DynamoDB

The Media Services Application Mapper provisions six Amazon DynamoDB tables during initial deployment: `alarms`, `channels`, `content`, `events`, `layout`, and `settings`.

The `alarms` table stores alarms states. The `channels` table stores AWS Elemental MediaLive channel definitions. The `content` table stores cached content. The `events` table stores information on changes in your media services or their status. The `layout` table stores diagram layout information for the browser application. The `settings` table stores general settings information for the browser application.

Read and Write Capacity

By default, the solution sets the Amazon DynamoDB throughput capacity to 10 read capacity units and 10 write capacity units. For customers with more than 50 OTT channels, we recommend increasing the DynamoDB throughput capacity. For more information, see Throughput Capacity for Reads and Writes.

DynamoDB Auto Scaling

The solution supports DynamoDB Auto Scaling. With auto scaling, you define a range (upper and lower limits) for read and write capacity units. If you choose to use auto scaling with this solution, we recommend setting minimum provisioned capacity to 5 units and maximum provisioned capacity to 250 units with a target utilization of 25%. We recommend setting your target utilization percentage low enough to ensure that scaling occurs before your usage reaches a critical level. For more information, see Managing Throughput Capacity Automatically with DynamoDB Auto Scaling.

Browser Application

This solution features a browser application that visualizes the relationship between your media services and the status of your resources and pipelines. The application visualizes your resources and connections as a workflow diagram.

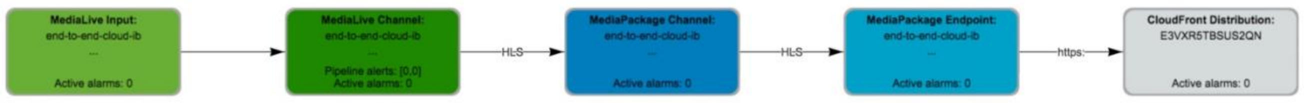
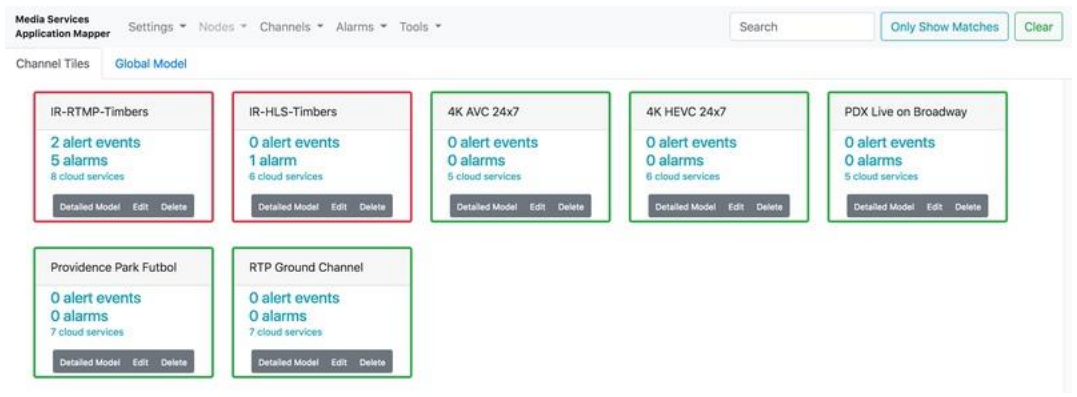


Figure 2: Example workflow diagram

The application includes two default layout configurations: vertical (top to bottom) and horizontal (left to right). You can also select and move individual items or groups of items to new locations to create your own custom layout. When you move items, the layout is saved in DynamoDB so all users will see the same diagrams at the same endpoint.

Channel Tiles

The application also features a tile view. Tiles aggregate resources into a single item that represents a single streaming video channel. You can create single tiles through the browser application or multiple tiles at one time with the solution's RESTful API.



IR-RTMP-Timbers
2 alert events
5 alarms
8 cloud services

IR-HLS-Timbers
0 alert events
1 alarm
6 cloud services

4K AVC 24x7
0 alert events
0 alarms
5 cloud services

4K HEVC 24x7
0 alert events
0 alarms
6 cloud services

PDX Live on Broadway
0 alert events
0 alarms
5 cloud services

Providence Park Futbol
0 alert events
0 alarms
7 cloud services

RTP Ground Channel
0 alert events
0 alarms
7 cloud services

Figure 3: Example channel tiles

Each tile displays the aggregated media service configuration information for all resources included in the file. You can also select the tile to see the tile's resources on the diagram.

Design Considerations

Customization

The Media Services Application Mapper is designed to be customized or extended. For example, you can add custom node types with the browser application, or by caching them into a database through a cloud-side task. You can also extend connection discovery and visualization overlay functionality to on-premise equipment.

AWS CloudFormation Templates

This solution uses AWS CloudFormation to automate the deployment of the Media Services Application Mapper on the AWS Cloud. It includes the following AWS CloudFormation templates, which you can download before deployment:

View template **msam-dynamodb-release.json:** Use this template to launch the solution's Amazon DynamoDB tables that store all solution-related data. You can also customize the template based on your specific needs.

View template **msam-core-release.json:** Use this template to launch all solution microservices. The default configuration deploys AWS Lambda functions that provide the business logic and data access layer for all solution operations. You can also customize the template based on your specific needs.

View template **msam-events-release.json:** Use this template to launch the solution's event handler microservices. You can also customize the template based on your specific needs.

View template **msam-browser-app-release.json:** Use this template to launch the solution's browser application for visualizing your media service resources and connections. You can also customize the template based on your specific needs.

Automated Deployment

Before you launch the automated deployment, please review the architecture, configuration, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the Media Services Application Mapper into your account.

Time to deploy: Approximately 40 minutes

What We'll Cover

The procedure for deploying this architecture on AWS consists of the following steps. For detailed instructions, follow the links for each step.

Step 1. Launch the DynamoDB Template

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters: **Stack Name**.

Step 2. Launch the Core Template

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters: **Stack Name**, **Alarms Table Name**, **Channels Table Name**, **Content Table Name**, **Events Table Name**, **Layout Table Name**, **Settings Table Name**.
- Review the other template parameter, and adjust if necessary.

Step 3. Launch the Events Template

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters: **Stack Name**, **Events Table Name**, **Events Table Region**.
- Review the other template parameter, and adjust if necessary.

Step 4. Launch the Browser App Template

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters: **Stack Name**.

Step 5. Configure the Browser App

- Enter values for the **Endpoint URL** and **API Key**.

Step 1. Launch the DynamoDB Template

This automated AWS CloudFormation template deploys the solution's Amazon DynamoDB tables.

Note: You are responsible for the cost of the AWS services used while running this solution. See the Cost section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Sign in to the AWS Management Console and click the button to the right to launch the `msam-dynamodb-release` AWS CloudFormation template.

   Launch Template

   You can also download the template as a starting point for your own implementation.
2. The template is launched 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 **Specify Details** page, assign a name to your solution stack.
4. Choose **Next.**
5. On the **Options** page, choose **Next**.

6. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
7. Choose **Create** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should see a status of CREATE_COMPLETE in roughly 10 minutes.
8. After the stack deploys, navigate to the stack **Outputs** tab and note the values of the `SettingsTable`, `ChannelsTable`, `LayoutTable`, `ContentTable`, `EventsTable`, and `AlarmsTable` keys. You will use these values as input parameters for the other templates.

Note: After the DynamoDB stack is created, you can launch the `Core`, `Events`, and `Browser App` templates at the same time. You do not need to wait for each stack launch to complete before you deploy the next template.

Step 2. Launch the Core Template

This automated AWS CloudFormation template deploys the Media Services Application Mapper microservices on the AWS Cloud.

Note: You are responsible for the cost of the AWS services used while running this solution. See the Cost section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Sign in to the AWS Management Console and click the button to the right to launch the `msam-core-release` AWS CloudFormation template.

   Launch Template

   You can also download the template as a starting point for your own implementation.
2. The template is launched 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 **Specify Details** page, assign a name to your solution stack.
4. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
Alarms Table Name	*<Requires input>*	The name of the table that stores alarm states. The name is generated by the DynamoDB template you deployed in Step 1.

Parameter	Default	Description
		Note: You can find the name of the alarms table in the DynamoDB stack **Outputs** tab. The table name is value of the **AlarmsTable** key.
Cache Item TTL	`7200`	The maximum time (in seconds) a cached item is retained, if never updated
Channels Table Name	*<Requires input>*	The name of the table that stores channel definitions. The name is generated by the DynamoDB template you deployed in Step 1. **Note:** You can find the name of the channels table in the DynamoDB stack **Outputs** tab. The table name is value of the **ChannelsTable** key.
Content Table Name	*<Requires input>*	The name of the table that stores cached content. The name is generated by the DynamoDB template you deployed in Step 1. **Note:** You can find the name of the content table in the DynamoDB stack **Outputs** tab. The table name is value of the **ContentTable** key.
Events Table Name	*<Requires input>*	The name of the table that stores events. The name is generated by the DynamoDB template you deployed in Step 1. **Note:** You can find the name of the events table in the DynamoDB stack **Outputs** tab. The table name is value of the **EventsTable** key. This table name must match the table name you enter in the **Events Table Name** events template parameter.
Layout Table Name	*<Requires input>*	The name of the table that stores diagram layout information. The name is generated by the DynamoDB template you deployed in Step 1. **Note:** You can find the name of the layout table in the DynamoDB stack **Outputs** tab. The table name is value of the **LayoutTable** key.
Settings Table Name	*<Requires input>*	The name of the table that stores configuration settings. The name is generated by the DynamoDB template you deployed in Step 1. **Note:** You can find the name of the settings table in the DynamoDB stack **Outputs** tab. The table name is value of the **SettingsTable** key.

5. Choose **Next.**
6. On the **Options** page, choose **Next**.
7. On the **Review** page, review and confirm the settings. Be sure to check the boxes acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
8. Select **Create Change Set**.
9. Choose **Execute** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should see a status of CREATE_COMPLETE in approximately 10 minutes.

Step 3. Launch the Events Template

This automated AWS CloudFormation template deploys the solution's Amazon DynamoDB tables.

Note: You are responsible for the cost of the AWS services used while running this solution. See the Cost section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Log in to the AWS Management Console and click the button to the right to launch the `msam-events-release` AWS CloudFormation template.

   Launch Template

   You can also download the template as a starting point for your own implementation.
2. The template is launched 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 **Specify Details** page, assign a name to your solution stack.
4. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
Events Table Name	*<Requires input>*	The name of the table that stores events. The name is generated by the DynamoDB template you deployed in Step 1. **Note:** You can find the name of the events table in the DynamoDB stack **Outputs** tab. The table name is value of the **EventsTable** key. This table name must match the table name you entered in the **Events Table Name** core template parameter.

Parameter	Default	Description
Events Table Region	*<Requires input>*	The AWS Region where the events table is located
Item TTL	`604800`	The maximum time (in seconds) a record in the events table is retained

5. Choose **Next.**
6. On the **Options** page, choose **Next**.
7. On the **Review** page, review and confirm the settings. Be sure to check the boxes acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
8. Select **Create Change Set**.
9. Choose **Execute** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should see a status of CREATE_COMPLETE in roughly 10 minutes.

Step 4. Launch the Browser App Template

This automated AWS CloudFormation template deploys the solution's Amazon DynamoDB tables.

Note: You are responsible for the cost of the AWS services used while running this solution. See the Cost section for more details. For full details, see the pricing webpage for each AWS service you will be using in this solution.

1. Log in to the AWS Management Console and click the button to the right to launch the `msam-browser-app-release` AWS CloudFormation template.

   Launch Template

   You can also download the template as a starting point for your own implementation.
2. The template is launched 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 **Specify Details** page, assign a name to your solution stack.
4. Choose **Next.**
5. On the **Options** page, choose **Next**.
6. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.

7. Choose **Create** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should see a status of CREATE_COMPLETE in roughly 10 minutes.

Step 5. Configure the Browser App

1. Navigate to the AWS CloudFormation console and check the box next to the core solution stack name.
2. In the stack **Outputs** tab, note the value of the **EndpointURL** key. You will need it later.
3. Navigate to the Amazon API Gateway console.
4. In the navigation pane, select **Usage Plans**.
5. Select **MSAM Usage Plan**.
6. In the **API Keys** tab, select the API key.
7. Select **Show**.
8. Note the API key. You will need it later.
9. Navigate to the AWS CloudFormation console and check the box next to the browser app stack name.
10. In the stack **Outputs** tab, select the value of the **MSAMBrowserURL** key.
11. In the **API Endpoint Connection** window, enter the **Endpoint URL** and the **API Key** you noted earlier.
12. Turn on **Do Not Remember** if you do not want to store the endpoint and the API key in your browser cookies after the browser is closed. If you do not turn on **Do Not Remember**, the endpoint and API key will be stored in your browser cookies and reused the next time you attempt to connect.

    Important: If a stored endpoint and API key are not used for seven consecutive days, they will be deleted automatically. When you use a stored endpoint and key, the expiration timer is reset.

13. Select **Connect**.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model can reduce your operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. For more information about security on AWS, visit the AWS Security Center.

Additional Resources

AWS services

- AWS CloudFormation
- AWS Lambda
- Amazon API Gateway
- Amazon DynamoDB

Source Code

You can visit our GitHub repository to download the templates and scripts for this solution, and to share your customizations with others.

Document Revisions

Date	Change
December 2018	Initial release
June 2019	Clarified information about the solution table names and where to find them
January 2020	Fixed Amazon CloudWatch link for EMX (MediaConnect) flows, added inventory and visualization support for AWS Elemental MediaLive multiplex, and made front-end performance improvements (repeated REST calls, expensive loop optimization)