



User Guide

Elemental Inference



Elemental Inference: User Guide

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What Is Elemental Inference?	1
Accessing Elemental Inference	1
Pricing	2
Setting up AWS Elemental Inference	3
Signing up for AWS	3
Sign up for an AWS account	3
Download tools	3
Quotas	5
Reference	6
Media requirements	6
Creating a workflow	7
Create the feed	7
Creating using the console	7
Creating using the CLI	9
Configuring each feature	10
Format the source media	22
Format requirements	22
Example	24
Deliver the source media	24
Streaming requirements	25
Example	26
Query the metadata	26
Metadata for smart crop	27
Metadata for smart subtitles	32
Querying contextual metadata	33
Playground	40
How the playground works	40
Permissions	41
Service role	41
User permissions	41
Creating a playground job	42
Viewing job results	42
Pricing	43
Troubleshooting	43

Working with existing feeds	45
Editing, archiving, and deleting feeds using the console	45
Revising using the CLI	46
Feed policies	47
Prerequisites	47
Policy requirements	47
Managing a policy using the console	48
Managing policies using the CLI	49
Attaching a policy (PutFeedPolicy)	49
Retrieving a policy (GetFeedPolicy)	49
Deleting a policy (DeleteFeedPolicy)	50
Example policy for MediaTailor access	50
Security considerations	51
Monitoring activity	52
Lifecycle of a feed	52
Lifecycle and status of a feed	52
Status of a resource	53
Status of an output	53
Lifecycle of metadata	54
Monitoring using the console	54
General details panel and feed association panel	54
Monitoring tab	55
Feed outputs tab	55
Monitoring using the CLI	56
Monitoring with metrics	56
Components of a metric	57
Pricing	57
Service availability and API metrics	57
Elemental Inference performance metrics	58
Monitoring events	60
Clip Metadata Generated event	61
CloudTrail logging	62
Elemental Inference information in CloudTrail	62
Understanding Elemental Inference log file entries	63
Security	66
Data protection	66

Identity and Access Management	67
Audience	68
Authenticating with identities	68
Managing access using policies	69
Compliance validation	71
Resilience	71
Infrastructure security	72
Document history	73

What Is Elemental Inference?

AWS Elemental Inference is a real-time service that lets you easily apply machine learning models to video, audio, and image content for automated analysis, classification, and insights generation. Features include smart crop, event clipping, smart subtitles, and contextual metadata. Smart subtitles uses automatic speech recognition (ASR) to generate TTML subtitles from audio in your source media. Contextual metadata uses AI to classify video content at the shot level and at the scene level. These classifications enable contextual ad targeting.

To preview smart cropping on a recorded video without setting up a live feed, use the Elemental Inference playground. For more information, see [Playground](#).

Topics

- [Accessing Elemental Inference](#)

Accessing Elemental Inference

You can access Elemental Inference using any of the following methods:

- **AWS Management Console** – The procedures throughout this guide explain how to use the AWS Management Console to perform tasks for AWS Elemental Inference.
- **AWS SDKs** – If you're using a programming language that AWS provides an SDK for, you can use an SDK to access AWS Elemental Inference. SDKs simplify authentication, integrate easily with your development environment, and provide easy access to Elemental Inference commands. For more information, see [Tools for Amazon Web Services](#).
- **AWS Elemental Inference API** – If you're using a programming language that an SDK isn't available for, see the [AWS Elemental Inference API Reference](#) for information about API actions and about how to make API requests.
- **AWS Command Line Interface** – For more information, see the [AWS Command Line Interface User Guide](#).
- **AWS Tools for Windows PowerShell** – For more information, see the [AWS Tools for PowerShell User Guide](#).

Pricing for AWS Elemental Inference

As with other AWS products, there are no contracts or minimum commitments for using Elemental Inference. You're charged only for AWS resources that your account uses. Pricing is pay-as-you-go and consists of the following:

- A per-minute charge for each feature (output) that you include in a feed, when the feed is processing source media.
- Savings accrue as you add more features.

For detailed pricing information, see [AWS Elemental Inference pricing](#).

If you use the Elemental Inference playground to preview smart cropping, each job incurs AWS Elemental MediaConvert charges in addition to Elemental Inference charges. You're charged only when a job completes successfully. For MediaConvert pricing, see [AWS Elemental MediaConvert pricing](#). For more information about the playground, see [Playground](#).

Setting up AWS Elemental Inference

This section provides procedures to set up your organization to use AWS Elemental Inference. It also provides information about determining the IAM permissions that users and other AWS identities require. These permissions let you impose restricted controls on users and other AWS identities, in conformance with the security policies and procedures of your organization.

Topics

- [Signing up for AWS](#)
- [Download tools](#)

Signing up for AWS

Sign up for an AWS account

To get started with AWS, you need an AWS account. For information about creating an AWS account, see [Getting started with an AWS account](#) in the *AWS Account Management Reference Guide*.

Download tools

The AWS Management Console includes a console for Elemental Inference, but if you want to access the services programmatically, see the following:

- The API guides document the operations that the services support and provide links to the related SDK and CLI documentation:
 - [AWS Elemental Inference API Reference](#)
- To call an API without having to handle low-level details like assembling raw HTTP requests, you can use an AWS SDK. The AWS SDKs provide functions and data types that encapsulate the functionality of AWS services. To download an AWS SDK and access installation instructions, see the applicable page:
 - [Go](#)
 - [JavaScript](#)
 - [.NET](#)

- [Node.js](#)
- [Python](#)
- [Ruby](#)

For a complete list of AWS SDKs, see [Tools for Amazon Web Services](#).

- You can use the AWS Command Line Interface (AWS CLI) to control multiple AWS services from the command line. You can also automate your commands using scripts. For more information, see [AWS Command Line Interface](#).
- AWS Tools for Windows PowerShell supports these AWS services. For more information, see [AWS Tools for PowerShell Cmdlet Reference](#).

Quotas for AWS Elemental Inference

Your AWS account has default quotas for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, and other quotas cannot be increased.

To view the quotas for Elemental Inference, open the [Service Quotas console](#). In the navigation pane, choose **AWS services** and select **Elemental Inference**.

To request a quota increase, see [Requesting a Quota Increase](#) in the *Service Quotas User Guide*. If the quota is not yet available in Service Quotas, use the [limit increase form](#).

Reference

Media requirements

Topic	Characteristic	Requirements
Container	Live or VOD?	Live only
Video	Codec	H.264 or H.265
	Framerate	30 frames per second
	Aspect ratio	Any
	Resolution	1280x720
Audio	Codec	AAC

Creating an Elemental Inference workflow

You must create a feed and enable at least one AI feature in that feed. After you have created the feed, you must associate one resource, which represents the source media that Elemental Inference will work on.

When you are ready, you must deliver the source media to Elemental Inference. Elemental Inference will produce metadata for each of the features that you set up. You must obtain that metadata and use it to produce the desired media, for example, to produce a video file of an event.

Topics

- [Step A: Create the feed in Elemental Inference](#)
- [Step B: Format the source media](#)
- [Step C: Deliver the source media](#)
- [Step D: Query the metadata](#)

Step A: Create the feed in Elemental Inference

You must create a feed that contains the features (outputs) that you want to use. After you've created the feed, you must associate a source media (resource) with it.

You can create an Elemental Inference feed using the Elemental Inference console or the AWS CLI.

Topics

- [Creating using the console](#)
- [Creating using the CLI](#)
- [Configuring each feature](#)

Creating using the console

This section describes how to use the Elemental Inference console to create an Elemental Inference feed.

Integration with AWS Elemental MediaLive

We recommend that you use Elemental Inference integrated with MediaLive. This combined workflow delivers live video processing alongside Elemental Inference. For more information, see the [AWS Elemental MediaLive User Guide](#).

Create the feed

1. Open the [Elemental Inference console](#).
2. In the navigation pane, choose **Feeds**. On the **Feeds** page, choose **Create feed**.
3. On the **Create feed** page, complete the following sections:
 - **Feed settings** – Enter a name for the feed. You might want to choose a name that helps you identify the source media that you plan to use with this feed. For example, **feed-soccer**.
 - **AI features** – Select at least one feature to enable. Each feature that you select becomes an output in the feed. The available features are **Smart Cropping**, **Event Clipping**, **Smart Subtitling**, and **Contextual Metadata**. For the configuration details of each feature, see the sections that follow this procedure.
 - **Access role** – Choose how Elemental Inference obtains the IAM role that it assumes on your behalf to access the resources a feed needs. You can use an existing role, create a new role from a template, or use a custom role ARN. An access role is required when a smart cropping output uses template groups, so that Elemental Inference can read your template images from Amazon S3.
 - (Optional) **Resource policy** – Add a resource policy to grant another AWS account or service read access to the feed's metadata. For more information, see [Feed policies](#).
 - (Optional) **Tags** – Add tags to the feed.
4. Choose **Create feed**. The **Feeds** page appears, showing a list with one line for each feed. After a few moments, the status of the feed that you created becomes **Available**, which means that the feed isn't currently associated with a source media.

Associate the resource

1. On the feed details page, in **Feed association**, choose **Add association**. Enter a name for the source media (resource) that you intend for this feed. You might want to choose a name that helps you identify the feed that this source media belongs to. For example, **source-soccer**.

2. Choose **Save** to confirm the association. When a resource is associated, the **Feed** information on the page updates:

- The status of the feed changes to **Active**, which means that a resource is associated with the feed.
- The **Integration** field shows the data endpoint for the feed.
- The status of each output changes to **Enabled**.

For information about feed and output status, see [the section called “Lifecycle of a feed”](#).

3. Make a note of the data endpoint (in the **Integration** field). You need this value to deliver the source media to Elemental Inference.

Creating using the CLI

This section describes how to use the AWS CLI to create an Elemental Inference feed.

You must set up a fully-configured feed: resource - feed - output or outputs, where the MediaLive channel is the resource and each output represents one Elemental Inference feature.

1. Use `create-feed` to create a new feed.

Include one output for each feature you want to implement. Set the status to `ENABLED` in each output.

(Optional) Specify an **access role** (`accessRoleArn`), the ARN of an IAM role that Elemental Inference assumes on your behalf to access the resources a feed needs. An access role is required when a smart crop output uses graphic composition, so that Elemental Inference can read your template images from Amazon S3.

2. The response includes the following information that you should make a note of:

- `id`: The feed ID, which you will need for CLI commands on this feed.
- `arn`: The feed ARN. You can also obtain the ARN using `get-feed`.
- `dataEndpoints`: The ARN of the data endpoint for this feed. You will use this ARN when you send source media to Elemental Inference for processing, and when you retrieve the metadata that is the result of this processing.

3. After the feed is created, the status of the feed will eventually change to `AVAILABLE`, indicating that it is ready to have a resource (source media) associated with it.

4. Use `associate-feed` to associate the source media with the feed. The source media is the resource for the feed.

You now have a useable feed: `resource - feed - output`.

Configuring each feature

Following are details about how to configure each feature (output) that you include in a Elemental Inference feed.

Configuring contextual metadata

Contextual metadata uses AI to classify video content at the shot level and at the scene level. It produces IAB Content Taxonomy v3.1 category tags and GARM (Global Alliance for Responsible Media) brand safety ratings. It also produces descriptive labels for the content, such as domains, objects, actions, people, environments, brands, and on-screen text, and a mood profile. For more information about these fields, see [the section called "Using the metadata"](#). The classifications and labels enable contextual ad targeting when used with AWS Elemental MediaLive and AWS Elemental MediaTailor ad insertion workflows.

Console

On the **Create feed** page, in the **AI features** section, select the **Contextual Metadata** check box. Configure the following settings:

- **Summary generation** (optional) – Controls whether Elemental Inference generates a descriptive summary and the objects and actions labels. Supported values:
 - **ENABLED** – Generates a descriptive summary and the objects and actions labels (default).
 - **DISABLED** – Doesn't generate a descriptive summary or the objects and actions labels.

If you omit `summaryGeneration`, Elemental Inference treats it as **ENABLED**. Set it to **DISABLED** if you don't want the summary, objects, and actions fields in the `GetMetadata` response.

- **Extended analysis** (optional) – Controls whether Elemental Inference generates the people, environments, brands, and `onScreenTexts` labels. Supported values:
 - **ENABLED** – Generates the people, environments, brands, and `onScreenTexts` labels (default).
 - **DISABLED** – Doesn't generate the people, environments, brands, or `onScreenTexts` labels.

If you omit `extendedAnalysis`, Elemental Inference treats it as `ENABLED`. You can set `summaryGeneration` and `extendedAnalysis` independently. For example, you can disable summaries and keep the extended analysis labels.

CLI example

The following example shows how to include a contextual metadata output when creating a feed using the CLI:

```
aws elemental-inference create-feed \  
  --name "my-feed" \  
  --outputs ' [{  
    "name": "contextual-metadata",  
    "status": "ENABLED",  
    "outputConfig": {  
      "contextualMetadata": {  
        "summaryGeneration": "ENABLED",  
        "extendedAnalysis": "ENABLED"  
      }  
    }  
  } ]'
```

Integration with AWS Elemental MediaTailor

Contextual metadata is typically used with the AWS Elemental MediaLive to AWS Elemental MediaTailor ad insertion workflow. To enable MediaTailor to query the metadata, attach a resource-based policy to the feed and configure a MediaTailor Function. For more information, see [Feed policies](#) and [the section called “Integration with AWS Elemental MediaTailor”](#).

Configuring event clipping

In **Callback config**, you can enter a string that you want Elemental Inference to always include in the event clipping metadata for this output. This information is useful when you later work with Elemental Inference events in Amazon EventBridge. You will be able to filter events using this information, in order to find the events for one feed. The string might identify the sports event in the feed, for example.

Aligning clips to plays with Sports Data

To use Sports Data, you find the game in the sports data feed and attach it to an event clipping output. Elemental Inference then aligns the clips for that output to the plays in the game.

Note

You must have a feed on a running MediaLive channel that carries the game. You must search for the game and attach it from the same AWS account. You can attach one game to each event clipping output.

Supported sports and leagues

Sports Data supports the following sports and leagues.

Sport	Leagues
Basketball	NBA, WNBA, NCAA Men's Basketball, NCAA Women's Basketball
American football	NFL, NCAA Men's Football

For each sport, Elemental Inference aligns clips to the following events. The tag is the value that identifies the event in the game data.

Basketball events

Event	Tag
Dunk	dunk
Three-pointer	threepointer
Two-pointer	twopointer

American football events

Event	Tag
Touchdown	touchdown
Punt return touchdown	puntreturntouchdown
Kickoff return touchdown	kickoffreturntouchdown
Interception	interception
Fumble	fumble
Pick six	picksix
Strip sack	stripsack
Turnover on downs	turnoverondowns

Step 1: Find your game

Use the `SearchFixtures` operation to find the game that you want to align clips to. You provide a sport and a date range, and optionally filter by team. Each game in the response includes a `fixtureId` that you attach to your event clipping output in the next step.

The following example searches for basketball games on 2026-03-03 and 2026-03-04 that involve the Comets.

```
$ awscli --service "elemental-inference" --region <region> \
-X POST "https://elemental-inference.<region>.amazonaws.com/v1/fixtures" \
-H "Content-Type: application/json" \
-d '{
  "sport": "basketball",
  "startDate": "2026-03-03",
  "endDate": "2026-03-04",
  "filters": [{ "name": "COMPETITOR", "values": ["Comets"] }]
}'
```

The response returns the matching games. Note the `fixtureId` of the game that you want to clip.

```
{
  "fixtures": [
    {
      "fixtureId":
"NwrW5B17Rw9W25g1jPlobSGdQfB0KyD3227bcsRK0w3Bq7aG2xRr08qx2XEj16R1s0H2kwyjbojN4Rk7qHiF2w",
      "name": "Northport Comets vs. Southport Meteors",
      "fixtureGroup": "Regular Season",
      "scheduledStart": "2026-03-04T03:00:00Z",
      "status": "Scheduled",
      "competitors": [
        { "name": "Meteors", "isHome": true },
        { "name": "Comets", "isHome": false }
      ]
    }
  ]
}
```

CLI example

The following example makes the same request using the AWS CLI:

```
aws elemental-inference search-fixtures \
  --sport "basketball" \
  --start-date "2026-03-03" \
  --end-date "2026-03-04" \
  --filters '[{"name": "COMPETITOR", "values": ["Comets"]}']'
```

Request parameters

Parameter	Required	Description
sport	Yes	Either <code>basketball</code> or <code>american-football</code> .
startDate	Yes	The first day to search, in <code>YYYY-MM-DD</code> format (UTC).
endDate	No	The last day to search, in <code>YYYY-MM-DD</code> format (UTC). Defaults to <code>startDate</code> . The window must be 7 days or fewer, inclusive.
filters	No	Up to 10 filters, with up to 10 values each. See the section called “Filtering by team” .

Parameter	Required	Description
maxResults	No	The maximum number of games to return, from 1 to 100. Defaults to 100. If the response includes a <code>nextToken</code> value, pass it in a subsequent request to retrieve the remaining games.

Game status

The status of a game tells you whether it is safe to attach.

Scheduled

The game has not started yet. You can attach it now, and clips begin when play starts.

InProgress

The game is underway. Clips align to plays as they happen.

Completed

The game has finished. Attach it to clip an already-aired game.

Note

Dates are whole UTC days. An evening game in the United States can fall on the next UTC day. For example, a game that starts at 03:00 UTC on March 4 appears on the March 4 search date, not March 3. If you don't see your game, set `endDate` to a day later.

You can search for past games the same way. Set `startDate` and `endDate` to dates in the past, and attach a `Completed` game to clip a game that has already aired.

Filtering by team

When you filter by team, every word that you send must appear in the game's team names. Matching is case-insensitive and matches partial words, but does not do fuzzy matching. You can search by city name even though the response returns only the team name.

For the game *Northport Comets vs. Southport Meteors*, the values `comets`, `Comet`, and `Northport` all match, but `Cometz` does not.

- **Either team** – use one filter with two values, such as ["Comets", "Meteors"].
- **An exact matchup** – use two filters, with one team in each.

Words match independently, so Northport Meteors also matches the game in the previous example.

Step 2: Attach the game to your output

Attach the game to an event clipping output by including the `fixtureId` in the output's `dataSourceConfiguration`. You can do this when you create the feed, or on an existing feed by using `UpdateFeed`. The `fixtureId` is the only value you provide, and you can attach one game to each event clipping output.

The following example attaches a game to an event clipping output when creating a feed.

```
aws elemental-inference create-feed \  
  --name "my-feed" \  
  --outputs ' [{  
    "name": "clipping",  
    "status": "ENABLED",  
    "outputConfig": {  
      "clipping": {  
        "dataSourceConfiguration": {  
          "fixtureId": "NwrW5B17Rw9W25g1jPlObSGdQfB0KyD3227bcsRK0w3..."  
        }  
      }  
    }  
  } ]'
```

Checking that it works

Clips appear only after both of the following are true: the game has started, and your channel is sending media. You can confirm the results as follows:

- Call `GetFeed` to confirm that the `fixtureId` stored on the output is the game that you picked.
- As clips are produced, they appear in the metadata that Elemental Inference emits for the output. This is the primary confirmation that Sports Data is working.

If you don't see clips yet, check the following.

Feed isn't receiving media

The feed isn't associated with a running channel yet, so no media is arriving. Start the channel, and confirm that it is sending to this feed.

Game hasn't started

A Scheduled game produces no clips until play starts. This is expected, not a fault.

Wrong game attached

Call `GetFeed` and compare the `fixtureId` against your search result.

Configuring smart crop

Smart crop has no required configuration. Optionally, you can add **graphic composition** to a smart crop output to detect known graphics, such as scoreboards and advertisements, in your source media. You provide one or more reference images, called **templates**, and Elemental Inference reports, for each analyzed frame, whether each graphic is present and where it appears as a bounding box. Elemental Inference returns graphic composition results as part of the smart crop metadata. For more information, see [the section called “Metadata for graphic composition”](#).

You can configure one to four template groups of reference images, where each group represents a single graphic to detect. Configuring at least one group enables graphic composition in the output. Each group has the following settings:

- **Group name** (required) – A name for the graphic. The name can be 1–128 characters, must start and end with an alphanumeric character, and can contain letters, numbers, hyphens (-), and underscores (_). Elemental Inference returns this same name in the metadata so that you can identify which graphic was detected.
- **Template image URIs** (required) – Up to two Amazon S3 URIs of reference images for the graphic. Provide more than one image when the same graphic can appear in more than one variation.

Note

Store your reference images in an Amazon S3 bucket in the same account that creates the feed, and that Elemental Inference can read using the access role (`accessRoleArn`) associated with the feed.

Template image recommendations

For the best detection performance, follow these recommendations when you prepare your template images:

- Provide each template as a PNG image with an alpha channel. Set the alpha value to 0 for any regions that Elemental Inference should ignore when matching. Elemental Inference uses only the non-transparent regions for detection. Make transparent the parts of the graphic that change from frame to frame. Leave the parts that stay the same fully opaque. For example, in a scoreboard template, set the alpha to 0 over the score and clock digits and over any animated or moving elements. Leave the parts that don't change, such as the scoreboard outline and the team labels, fully opaque.
- Graphic detection runs at a resolution of 2560 x 1440. For the best results, size each template to match how large the graphic appears in a 2560 x 1440 frame.

CLI example

The following example shows how to include a smart crop output that detects two graphics, a scoreboard and ads, when creating a feed using the CLI:

```
aws elemental-inference create-feed \  
  --name "my-feed" \  
  --access-role-arn "arn:aws:iam::111122223333:role/my-ei-access-role" \  
  --outputs '[{  
    "name": "crop",  
    "status": "ENABLED",  
    "outputConfig": {  
      "cropping": {  
        "templateGroups": [  
          {  
            "name": "scoreboard",  
            "templateUri": [  
              "s3://amzn-s3-demo-bucket/scoreboard-v1.png",  
              "s3://amzn-s3-demo-bucket/scoreboard-v2.png"  
            ]  
          },  
          {  
            "name": "ads",  
            "templateUri": [  
              "s3://amzn-s3-demo-bucket/ads.png"  
            ]  
          }  
        ]  
      }  
    }  
  ]
```

```
}  
  ]  
    }  
      }  
        }  
      } ]'
```

Configuring smart subtitles

Smart subtitles uses automatic speech recognition (ASR) to generate TTML subtitles from the audio in your source media. Configure the following settings for the smart subtitles output:

- **Language** (required) – The language of the audio in the source media. Elemental Inference uses this setting to optimize transcription accuracy. Supported values:
 - deu – German
 - eng – English
 - eng-au – English (Australia)
 - eng-gb – English (Great Britain)
 - eng-us – English (United States)
 - fra – French
 - ita – Italian
 - por – Portuguese
 - spa – Spanish
- **Aspect ratio** (optional) – The width and height of the output video, specified as integer values. Elemental Inference uses the aspect ratio to determine subtitle layout and line lengths.
- **Custom dictionary** (optional) – The ID of a custom dictionary to improve transcription accuracy for domain-specific terminology. For information about creating and managing dictionaries, see [the section called “Managing dictionaries”](#).
- **Profanity filter** (optional) – Controls how profanity is handled in the generated subtitles. Supported values:
 - DISABLED – No filtering (default). All words appear as transcribed.
 - CENSOR – Replace profanity with asterisks.
 - DROP – Remove profanity from the transcript entirely.

The profanity filter is available only for English, Spanish, and Italian.

CLI example

The following example shows how to include a smart subtitles output when creating a feed using the CLI:

```
aws elemental-inference create-feed \  
  --name "my-feed" \  
  --outputs ' [{  
    "name": "subtitles",  
    "status": "ENABLED",  
    "outputConfig": {  
      "subtitling": {  
        "language": "eng",  
        "aspectRatio": {"width": 16, "height": 9},  
        "profanityFilter": "DISABLED"  
      }  
    }  
  } ]'
```

Managing dictionaries

A dictionary contains custom words and phrases that the ASR engine might not recognize, such as brand names, technical terms, or proper nouns. You can reference a dictionary when configuring a smart subtitles output to improve transcription accuracy for domain-specific terminology.

Use the Elemental Inference dictionary API operations to manage dictionaries:

- **CreateDictionary** – Create a new dictionary. Specify a name, language, and optionally provide entries.
- **GetDictionary** – Retrieve details about a dictionary.
- **UpdateDictionary** – Update the name, language, or entries of a dictionary.
- **ExportDictionaryEntries** – Export the entries from a dictionary.
- **ListDictionaries** – List all dictionaries in your account.
- **DeleteDictionary** – Deletes a dictionary. You cannot delete a dictionary that is referenced by a feed. To delete a dictionary, first update or delete any feeds that reference it.

Dictionary validation rules

When creating or updating a dictionary, the following validation rules apply:

- **Name**

- 1–128 characters
- Must start and end with an alphanumeric character
- Allowed characters: letters, digits, hyphen (-), underscore (_)

- **Language**

One of: deu, eng, fra, ita, por, spa

Regional language variants (such as eng-au, eng-gb, eng-us) are not supported for dictionaries. Use the base language code (for example, eng) instead.

- **Entries** (JSON payload)

- Must be a valid JSON array. A top-level object or scalar is rejected.
- Maximum 40 KB serialized payload size.
- Each entry must include a content field that is not blank.
- Each entry may optionally include a sounds_like field. If provided, it must be an array of non-blank strings.
- Each sounds_like hint must contain only characters from the dictionary language's primary script. Currently, all supported languages use Latin script (Latin alphabet, accented letters, and script-neutral punctuation are accepted; non-Latin scripts are rejected).

Managing dictionaries using the console

You can create and manage dictionaries using the Elemental Inference console.

Creating a dictionary

1. In the navigation pane, choose **Dictionaries** to open the **Dictionaries** list.
2. Choose **Create dictionary**.
3. On the **Create dictionary** page, under **Dictionary settings**, enter a **Name** and choose a **Language**.
4. Under **Entries**, add entries in one of the following ways:
 - Choose **Add entry** to add an entry.
 - Paste a JSON array of entries.
 - Choose **Choose file** to upload a JSON file.

For example, the following entry defines the term Kubernetes and how it sounds:

```
[{"content": "Kubernetes", "sounds_like": ["koo-ber-net-eez"]}]
```

Entries are limited to 40,000 characters.

5. Optionally, add tags. Then choose **Create dictionary**.

Viewing dictionary details

To view a dictionary, choose it from the **Dictionaries** list. The **Dictionary details** page shows the following information:

- **General details** shows the **Dictionary ID** and the **Language**.
- The **Entries** tab lists the entries, with a **Content** column and a **Sounds like** column.
- The **References** tab lists the feeds that reference the dictionary, in a **Feed ID** column. You can't delete a dictionary that a feed references.
- To delete a dictionary, choose **Delete** on the **Dictionaries** list or the dictionary details page, and then confirm.

Editing dictionary entries

To edit the entries in a dictionary, on the **Entries** tab, choose **Edit** to open the **Edit entries** page. When you save, Elemental Inference replaces all of the dictionary's entries.

Step B: Format the source media

Make sure that the source media meets the requirements.

Format requirements

You must format the source media according to the CMAF Ingest (Interface-1) version 1.2 specification.

The following table identifies specific requirements for Elemental Inference.

Characteristic	Requirement
Media fragments	Fragmented CMAF Ingest containerized media fragments
MovieFragmentBox	One per segment
Initialization segment: naming	Include an initialization segment with each stream, as follows: • For video: Streams(default-video.cmfv)/InitializationSegment • For audio: Streams(default-audio.cmfa)/InitializationSegment
Media segments: naming	Media segments after the initialization segment must following this naming pattern: Streams default-< <i>type</i> >.< <i>ext</i> >/Segment (< <i>sequence-number</i> >) Where: <type> is video or audio <ext> is cmfv or cmfa <sequence-number> must increase monotonically, although it doesn't have to be contiguous. Each sequence number must match the sequence number in the MovieFragmentHeader box. For example: Streams default-video.cmfv/Segment(< <i>sequence-number</i> >)
End of Stream indicator	The last media segment in the session must be: • A media segment with the lmsg brand included in the compatible brands under the SegmentTypeBox.

Characteristic	Requirement
	If you can't signal the end of stream in this way, there is a workaround. See the section called “Deliver the source media” .

Example

The following code shows how to use FFMPG to format the media to follow these requirements. The commands demux, segment, and containerize the video and audio. Note the `-init_seg_name` and `-media_seg_name` lines

```
$ mkdir 'Streams(default-video.cmfv)'
$ ffmpeg -i input.mp4 \
-map 0:v:0 -c:v libx264 \
  -profile:v main -pix_fmt yuv420p \
  -g 30 -keyint_min 30 -sc_threshold 0 \
  -force_key_frames 'expr:gte(t,n_forced*1)' \
  -f dash -seg_duration 1 -use_timeline 0 \
  -use_template 1 -remove_at_exit 0 \
  -init_seg_name 'Streams(default-video.cmfv)/InitializationSegment' \
  -media_seg_name 'Streams(default-video.cmfv)/Segment($Number%09d$)' \
  'video.mpd'

$ mkdir 'Streams(default-audio.cmfa) '$ ffmpeg -i input.mp4 \
-map 0:a:0 -c:a aac -ar 48000 -ac 2 \
-f dash -seg_duration 1 -use_timeline 0 \
-use_template 1 -remove_at_exit 0 \
-init_seg_name 'Streams(default-audio.cmfa)/InitializationSegment' \
-media_seg_name 'Streams(default-audio.cmfa)/Segment($Number%09d$)' \
'audio.mpd'
```

Step C: Deliver the source media

To deliver the source media, you must use an AWS API or SDK. You can't deliver the media using the Elemental Inference console.

1. Obtain the data endpoint for the feed:

- Using an AWS API or SDK: Use the GetFeed operation of Elemental Inference. For information about the operation and parameters, see https://docs.aws.amazon.com/elemental-inference/latest/APIReference/API_GetFeed.
 - Using the Elemental Inference console: Choose Feed in the left navigation bar, then select the feed. In the details page, the data endpoint is in the **Integration** field.
2. Use the Elemental Inference PutMedia operation to deliver the source media to that data endpoint. Make sure to stay within the quotas for the Request rate for PutMedia and the Request rate for PutMedia (in a burst). To view the current quotas, open the [Service Quotas console](#). In the navigation pane, choose **AWS services** and select **AWS Elemental Inference**.

Streaming requirements

You must deliver the source media according to the CMAF Ingest (Interface-1) version 1.2 specification.

The following table identifies specific requirements for Elemental Inference.

Characteristic	Requirement
Media segment duration	0-1.2 seconds
Ingestion order	Elemental Inference will ingest all media segments (audio and video) for a given sequence number before proceeding to the next sequence number
End of Stream indicator (workaround)	Typically, the last media segment includes 1msg. However, if you can't signal the end of stream in this way (for example, you are using FFMG), then flush the Elemental Inference internal buffer as follows: • Send up to 10 seconds of slate.
Manifest	CMAF Ingest doesn't support manifests.

Example

The following code sample shows how to use CURL to use the PUT operation to send the content to the data endpoint of a feed. (The example assumes that you have exported your AWS credentials as environment variables, which is standard practice when using the AWS REST API.)

```
# Initialization
$ awscli --region <region> --service elemental-inference -X PUT \
  'https://<data-endpoint>/v1/feed/<feed-id>/input/0/media/Streams(default-audio.cmfa)/
InitializationSegment' \
  --data-binary -d '@Streams(default-audio.cmfa)/InitializationSegment'

$ awscli --region <region> --service elemental-inference -X PUT \
  'https://<data-endpoint>/v1/feed/<feed-id>/input/0/media/Streams(default-video.cmfv)/
InitializationSegment' \
  --data-binary -d '@Streams(default-video.cmfv)/InitializationSegment'

# Media
$ awscli --region <region> --service elemental-inference -X PUT \
  'https://<data-endpoint>/v1/feed/<feed-id>/input/0/media/Streams(default-audio.cmfa)/
Segment(<sequence>)' \
  --data-binary -d '@Streams(default-audio.cmfa)/Segment(<sequence>)'

$ awscli --region <region> --service elemental-inference -X PUT \
  'https://<data-endpoint>/v1/feed/<feed-id>/input/0/media/Streams(default-video.cmfv)/
Segment(<sequence>)' \
  --data-binary -d '@Streams(default-video.cmfv)/Segment(<sequence>)'
```

Step D: Query the metadata

Use the Elemental Inference GetEndpoint operation to obtain the metadata that Elemental Inference generates.

For example, the following CURL code shows how to use the POST command to query for the metadata for the output named testOutput. The query is for the first second of metadata. This one second span is identified by the start PTS of 0 and the end PTS of 1001.

```
# Query the first second of metadata
$ awscli --service "elemental-inference" --region <region> \
  -X POST 'https://<data-endpoint>/v1/feed/<feed-id>/input/0/metadata' \
  -H "Content-Type: application/json" \
```

```
-d '{"outputName": "testOutput", "timeSpecification": { "ptsBased":
{ "startPts":0, "endPts": 1001, "timescale": 1000 } }, "parameters": {"smartCropping":
{"frameRate": { "numerator": 24, "denominator": 1}}}}'
```

For information about the metadata returned for each feature, see the following topics.

Topics

- [Metadata for smart crop](#)
- [Metadata for smart subtitles](#)
- [Querying contextual metadata](#)

Metadata for smart crop

The following CURL code shows the query command plus the results when the output testOutput is a smart crop output.

```
# Query the first second of metadata
$ awscli --service "elemental-inference" --region <region> \
-X POST 'https://<data-endpoint>/v1/feed/<feed-id>/input/0/metadata' \
-H "Content-Type: application/json" \
-d '{"outputName": "testOutput", "timeSpecification": { "ptsBased": { "startPts":0,
"endPts": 1001, "timescale": 1000 } }, "parameters": {"smartCropping": {"frameRate":
{ "numerator": 24, "denominator": 1}}}}'
{
  "items": [
    {
      "metadata": {
        "smartCropping": {
          "crop": {
            "centerPoint": {
              "scale": 10000,
              "xPosition": 2176,
              "yPosition": 6250
            }
          }
        }
      },
      "pts": 0,
      "timecode": null
    },
  ],
}
```

```

{
  "metadata": {
    "smartCropping": {
      "crop": {
        "centerPoint": {
          "scale": 10000,
          "xPosition": 2176,
          "yPosition": 6250
        }
      }
    }
  },
  "pts": 41,
  "timecode": null
},
{
  "metadata": {
    "smartCropping": {
      "crop": {
        "centerPoint": {
          "scale": 10000,
          "xPosition": 2208,
          "yPosition": 6238
        }
      }
    }
  },
  "pts": 83,
  "timecode": null
},
.
.
.
{
  "metadata": {
    "smartCropping": {
      "crop": {
        "centerPoint": {
          "scale": 10000,
          "xPosition": 2873,
          "yPosition": 5781
        }
      }
    }
  }
}

```

```
    },  
    "pts": 1000,  
    "timecode": null  
  }  
]  
}
```

Using the metadata

For smart crop, Elemental Inference identifies a *region of interest* in each frame. Elemental Inference then generates metadata that identifies the centerpoint in that region. You can develop a solution that uses this metadata to crop and scale the video. The centerpoint provides you with a reference point for the cropping and scaling algorithms that you develop.

The centerpoint is identified using three pieces of data:

- scale is a reference for calculating the positions as a percentage.
- X position is the position of the centerpoint on the X-axis, from the top left corner of the video frame. Always a positive number.
- Y position is the position of the centerpoint on the Y-axis, from the top left corner of the video frame. Always a positive number.

You can use this data to calculate the centerpoint pixel position in output video of any resolution. The formulas for finding the centerpoint are:

$(X \text{ position}) \times \text{width of output video} / \text{scale}$

$(Y \text{ position}) \times \text{height of output video} / \text{scale}$

Example 1

For example, if the output video is 1920 x 1080, then the following applies to the first piece of data in the metadata example:

- The X pixel position is $2176 \times 1920 / 10000 = \text{pixel } 417.792$ or 418 rounded up
- The Y pixel position is $6250 \times 1080 / 10000 = \text{pixel } 675$

Example 2

Or if the output video is 1280 x 720, then the following applies:

- The X pixel position is $2176 \times 1280 / 10000 = \text{pixel } 278.528$ or 279 rounded up
- The Y pixel position is $6250 \times 720 / 10000 = \text{pixel } 450$

Metadata for graphic composition

When you configure graphic composition on a smart crop output (see [the section called “Configuring smart crop”](#)), the smart crop metadata includes a `graphics` list in addition to the `crop centerPoint`. The list contains one entry for each template group that Elemental Inference evaluated for the frame.

Each graphic reports whether it is present in the frame. When `isPresent` is true, the entry also includes the bounding box that locates the graphic within the frame. When `isPresent` is false, Elemental Inference omits the position and size fields.

The following table describes the fields in each graphic composition entry.

Field	Always present	Description
<code>name</code>	Yes	The name of the template group. This matches the name you configured, so that you can identify which graphic was detected.
<code>isPresent</code>	Yes	Whether the graphic was detected in this frame. When this value is false, the position and size fields are omitted.
<code>xPosition</code>	No	The X position of the top-left corner of the bounding box, measured from the top-left corner of the video frame and normalized against scale. Always a positive number.
<code>yPosition</code>	No	The Y position of the top-left corner of the bounding box, measured from the top-left corner of the video frame and normalized against scale. Always a positive number.
<code>width</code>	No	The width of the bounding box, as a fraction of the frame width, normalized against scale.

Field	Always present	Description
height	No	The height of the bounding box, as a fraction of the frame height, normalized against scale.
scale	No	The reference value used to normalize the position and size fields, so that you can calculate them as a percentage of the frame dimensions.

The following example shows smart crop metadata in which Elemental Inference detected a scoreboard graphic:

```
{
  "metadata": {
    "smartCropping": {
      "crop": {
        "centerPoint": {
          "scale": 10000,
          "xPosition": 2176,
          "yPosition": 6250
        }
      },
      "graphics": [
        {
          "name": "scoreboard",
          "isPresent": true,
          "xPosition": 1602,
          "yPosition": 1201,
          "width": 2250,
          "height": 1264,
          "scale": 10000
        }
      ]
    }
  },
  "pts": 0,
  "timecode": null
}
```

Use the returned name to identify the graphic and to look up the action that your downstream encoder or application must take for it. Use the bounding box to locate the graphic within the frame.

xPosition and *yPosition* give the top-left corner of the bounding box. You calculate pixel values from the normalized values by multiplying by the output video dimension and dividing by *scale*. For an output video that is *W* pixels wide and *H* pixels high:

- Left edge (pixels) = $xPosition \times W / scale$
- Top edge (pixels) = $yPosition \times H / scale$
- Width (pixels) = $width \times W / scale$, and height (pixels) = $height \times H / scale$

Metadata for smart subtitles

For smart subtitles, Elemental Inference returns the metadata as a TTML (Timed Text Markup Language) document encapsulated in the JSON response. The TTML contains the transcribed text with timing information that corresponds to the requested time range.

The following `awscurl` command shows how to query for smart subtitles metadata:

```
# Query the first 5 seconds of subtitles metadata
$ awscurl --service "elemental-inference" --region <region> \
  -X POST 'https://<data-endpoint>/v1/feed/<feed-id>/input/0/metadata' \
  -H "Content-Type: application/json" \
  -d '{"outputName": "subtitles", "timeSpecification": { "ptsBased": { "startPts": 0, "endPts": 5000, "timescale": 1000 }}, "parameters": { "subtitling": { "format": "TTML" }}}'
```

The response contains a TTML document with subtitle cues timed to the requested range. Each subtitle cue includes a start time, end time, and the transcribed text.

Using the metadata

The TTML subtitles returned by Elemental Inference can be used in the following ways:

- Embed the subtitles directly into your video player as a subtitle track.
- Convert the TTML to other subtitle formats such as WebVTT or SRT for compatibility with different players and platforms.
- Use the timed text for downstream processing such as search indexing or content analysis.

Querying contextual metadata

For contextual metadata, Elemental Inference returns content classifications that include IAB Content Taxonomy v3.1 category IDs, GARM brand safety ratings, and descriptive labels. The response contains one item for each shot and each scene that overlaps the requested time range, ordered by presentation timestamp (PTS). The fields that Elemental Inference returns depend on the `summaryGeneration` and `extendedAnalysis` settings of the output. For more information, see [the section called “Configuring contextual metadata”](#).

The following `awscurl` command shows how to query for contextual metadata. The `parameters` field is required, and must contain a `contextualMetadata` object that matches the type of the output that you name in `outputName`.

```
# Query contextual metadata
$ awscurl --service "elemental-inference" --region <region> \
  -X POST 'https://<data-endpoint>/v1/feed/<feed-id>/input/0/metadata' \
  -H "Content-Type: application/json" \
  -d '{"outputName": "contextual-metadata", "timeSpecification": { "ptsBased":
{ "startPts": 0, "endPts": 5000, "timescale": 1000 }}, "parameters":
{ "contextualMetadata": {}}}
```

The response contains contextual classification results. The following example shows one scene-level item and one shot-level item from a tennis broadcast, for a feed that has summary generation and extended analysis enabled, so the items include every contextual metadata field:

```
{
  "items": [
    {
      "pts": 95000,
      "metadata": {
        "contextualMetadata": {
          "type": "SCENE",
          "startPts": 60000,
          "iabTaxonomy": {
            "version": "V3_1",
            "categories": [
              {"uniqueId": "539", "path": ["Sports", "Tennis"]}
            ]
          },
          "garm": {
            "suitability": {
```

```

    "categories": [
      {"category": "ADULT_EXPLICIT_SEXUAL", "flagged": false},
      {"category": "ARMS_AMMUNITION", "flagged": false},
      {"category": "CRIME_HARMFUL_ACTS", "flagged": false},
      {"category": "DEATH_INJURY_MILITARY_CONFLICT", "flagged": false},
      {"category": "DEBATED_SENSITIVE_SOCIAL_ISSUES", "flagged": false},
      {"category": "ILLEGAL_DRUGS_TOBACCO_ALCOHOL", "flagged": false},
      {"category": "HATE_SPEECH_ACTS_OF_AGGRESSION", "flagged": false},
      {"category": "OBSCENITY_PROFANITY", "flagged": false},
      {"category": "ONLINE_PIRACY", "flagged": false},
      {"category": "SPAM_HARMFUL_CONTENT", "flagged": false},
      {"category": "TERRORISM", "flagged": false}
    ]
  },
  "objects": ["tennis racket", "tennis ball", "net", "scoreboard graphic"],
  "actions": ["serving", "returning serve", "running", "celebrating"],
  "people": ["athlete", "umpire", "spectator"],
  "environments": ["stadium", "tennis court"],
  "mood": {
    "anger": 1.2,
    "anticipation": 7.9,
    "joy": 5.4,
    "trust": 5.7,
    "fear": 2.4,
    "surprise": 4.6,
    "sadness": 1.6,
    "disgust": 1.0
  },
  "domains": ["Sports"],
  "brands": ["AnyCompany", "Example Sports Network"],
  "onScreenTexts": ["EXAMPLE OPEN", "PLAYER A 6 4 30", "PLAYER B 3 4 15",
    "BREAK POINT"],
  "summary": "Player A wins a long rally to earn a break point, and the crowd
    applauds."
  }
},
{
  "pts": 95000,
  "metadata": {
    "contextualMetadata": {
      "type": "SHOT",
      "startPts": 78000,

```

```

    "iabTaxonomy": {
      "version": "V3_1",
      "categories": [
        {"uniqueId": "539", "path": ["Sports", "Tennis"]}
      ]
    },
    "garm": {
      "suitability": {
        "categories": [
          {"category": "ADULT_EXPLICIT_SEXUAL", "flagged": false},
          {"category": "ARMS_AMMUNITION", "flagged": false},
          {"category": "CRIME_HARMFUL_ACTS", "flagged": false},
          {"category": "DEATH_INJURY_MILITARY_CONFLICT", "flagged": false},
          {"category": "DEBATED_SENSITIVE_SOCIAL_ISSUES", "flagged": false},
          {"category": "ILLEGAL_DRUGS_TOBACCO_ALCOHOL", "flagged": false},
          {"category": "HATE_SPEECH_ACTS_OF_AGGRESSION", "flagged": false},
          {"category": "OBSCENITY_PROFANITY", "flagged": false},
          {"category": "ONLINE_PIRACY", "flagged": false},
          {"category": "SPAM_HARMFUL_CONTENT", "flagged": false},
          {"category": "TERRORISM", "flagged": false}
        ]
      }
    },
    "objects": ["tennis racket", "tennis ball", "net", "scoreboard graphic"],
    "actions": ["serving", "returning serve", "running"],
    "people": ["athlete", "umpire"],
    "environments": ["tennis court"],
    "mood": {
      "anger": 1.1,
      "anticipation": 8.2,
      "joy": 4.3,
      "trust": 5.6,
      "fear": 2.3,
      "surprise": 3.3,
      "sadness": 1.5,
      "disgust": 1.0
    },
    "domains": ["Sports"],
    "brands": ["AnyCompany"],
    "onScreenTexts": ["PLAYER A 6 4 30", "PLAYER B 3 4 15"],
    "summary": "Player B serves, and Player A returns the ball down the line."
  }
}
}

```

```
]
}
```

Using the metadata

Contextual metadata provides content classifications that you can use for contextual ad targeting. The response contains the following structure:

- **pts** – The presentation timestamp of the metadata item, in the timebase of the media.
- **timecode** – The timecode of the metadata item, when the source media carries timecode information.
- **contextualMetadata** – The top-level object containing classification results.
 - **type** – The granularity of the classification. SHOT for a single continuous camera take, or SCENE for a group of related consecutive shots. Elemental Inference returns both shot-level and scene-level items for a time range, so a shot item and the scene item that contains it can both appear in the response.
 - **startPts** – The presentation timestamp at which the shot or scene begins. For a scene, this value can be earlier than the pts of the item and earlier than the start of the requested time range.
 - **iabTaxonomy** – IAB Content Taxonomy classifications. This field is omitted when Elemental Inference does not match any category.
 - **version** – The taxonomy version used (currently V3_1).
 - **categories** – An array of matched content categories. Each category includes:
 - **path** – The hierarchical category path (for example, ["Sports", "Basketball"]).
 - **uniqueId** – The IAB taxonomy unique identifier for the category.
 - **garm** – GARM (Global Alliance for Responsible Media) brand safety classifications.

The 11 GARM brand safety categories are the following: ADULT_EXPLICIT_SEXUAL, ARMS_AMMUNITION, CRIME_HARMFUL_ACTS, DEATH_INJURY_MILITARY_CONFLICT, DEBATED_SENSITIVE_SOCIAL_ISSUES, ILLEGAL_DRUGS_TOBACCO_ALCOHOL, HATE_SPEECH_ACTS_OF_AGGRESSION, OBSCENITY_PROFANITY, ONLINE_PIRACY, SPAM_HARMFUL_CONTENT, and TERRORISM.

- **suitability** – Brand suitability assessment.
 - **categories** – An array of GARM brand safety categories. Each category includes:

- **category** – The GARM category name (for example, ARMS_AMMUNITION, HATE_SPEECH_ACTS_OF_AGGRESSION, ADULT_EXPLICIT_SEXUAL).
- **flagged** – Whether the content is flagged for this category (true or false).
- **risk** – The risk level for the category (FLOOR, HIGH, MEDIUM, or LOW). Elemental Inference includes this field only when flagged is true.
- **objects** – Labels for the notable objects that Elemental Inference detects in the shot or scene, such as basketball or scoreboard. For a scene, the list is the union of the labels from the shots in the scene. Elemental Inference omits this field when it detects no objects. Elemental Inference doesn't return this field when the output is configured with summaryGeneration set to DISABLED.
- **actions** – Labels for the notable actions that Elemental Inference detects in the shot or scene, such as jumping or cheering. For a scene, the list is the union of the labels from the shots in the scene. Elemental Inference omits this field when it detects no actions. Elemental Inference doesn't return this field when the output is configured with summaryGeneration set to DISABLED.
- **people** – Labels for the roles of the people in the shot or scene, such as athlete, anchor, or referee. The labels describe roles, not the names of individuals. For a scene, the list is the union of the labels from the shots in the scene. Elemental Inference omits this field when it identifies no roles. Elemental Inference doesn't return this field when the output is configured with extendedAnalysis set to DISABLED.
- **environments** – Labels for the kinds of settings that Elemental Inference identifies in the shot or scene, such as stadium, newsroom, or kitchen. The labels describe the type of place, not a named venue or geographic location. For a scene, the list is the union of the labels from the shots in the scene. Elemental Inference omits this field when it identifies no settings. Elemental Inference doesn't return this field when the output is configured with extendedAnalysis set to DISABLED.
- **mood** – The emotional tone of the shot or scene, based on the eight primary emotions of Plutchik's wheel of emotions. The field contains the anger, anticipation, joy, trust, fear, surprise, sadness, and disgust fields. Each field is a number from 1 to 10 that indicates the intensity of that emotion, where 1 is the lowest intensity and 10 is the highest. Elemental Inference always returns this field.
- **domains** – Labels for the content genres or subject areas of the shot or scene, such as Sports, News, or Entertainment. For a scene, the list is the union of the labels from the shots in the scene. Elemental Inference omits this field when it identifies no domains.

- **brands** – The names of brands or logos that Elemental Inference clearly identifies in the shot or scene. For a scene, the list is the union of the labels from the shots in the scene. Elemental Inference omits this field when it identifies no brands. Elemental Inference doesn't return this field when the output is configured with `extendedAnalysis` set to `DISABLED`.
- **onScreenTexts** – The lines of legible text that appear on screen during the shot or scene, such as scoreboard or caption text, copied exactly as displayed. For a scene, the list is the union of the labels from the shots in the scene. Elemental Inference omits this field when it detects no legible text. Elemental Inference doesn't return this field when the output is configured with `extendedAnalysis` set to `DISABLED`.
- **summary** – A short natural-language description of what happens in the shot or scene. Elemental Inference doesn't return this field when the output is configured with `summaryGeneration` set to `DISABLED`. For more information, see [the section called “Configuring contextual metadata”](#).

Integration with AWS Elemental MediaTailor

You can integrate Elemental Inference contextual metadata with AWS Elemental MediaTailor to enrich ad requests with content classifications. Your ad-decision server can then target ads based on what is happening in the content.

This integration uses a MediaTailor Function of type `AWS_SERVICE_REQUEST` to call the Elemental Inference `GetMetadata` API during ad breaks. AWS Elemental MediaLive decorates SCTE-35 markers with Elemental Inference query parameters, and MediaTailor parses these markers to construct authenticated requests to Elemental Inference.

How it works

1. Elemental Inference analyzes your content and produces shot-level and scene-level IAB Content Taxonomy and GARM brand safety classifications.
2. AWS Elemental MediaLive decorates SCTE-35 ad break markers with Elemental Inference query parameters (feed endpoint, region, and timing information).
3. At each ad break, MediaTailor triggers the configured Function, which sends an authenticated request to `GetMetadata` to retrieve classifications for the content window.
4. The Function output expressions extract IAB and GARM signals and pass them to the ad-decision server.

Prerequisites

Before you configure this integration, you need the following:

- An Elemental Inference feed with a contextual metadata output enabled.
- A resource-based policy on the feed that grants MediaTailor permission to call GetMetadata. For more information, see [Feed policies](#).
- An AWS Elemental MediaLive channel with contextual metadata enrichment enabled and associated with the feed.

Configuring MediaTailor

For complete instructions on creating the `AWS_SERVICE_REQUEST` Function, attaching it to a playback configuration, and configuring output expressions, see [Contextual ad targeting with Elemental Inference](#) in the *AWS Elemental MediaTailor User Guide*.

If the GetMetadata request fails or times out, MediaTailor proceeds with the ad request without contextual metadata. The ad-decision server falls back to its default targeting logic.

Using the Elemental Inference playground

The Elemental Inference playground lets you preview smart cropping on a recorded MP4 video file in Amazon S3, without setting up a live feed. Use it to see how smart cropping reframes your content before you build a live workflow.

When you submit a playground job, it runs as an AWS Elemental MediaConvert job in your account. During the job, MediaConvert creates an Elemental Inference feed to analyze the video, and then deletes the feed when the job ends. MediaConvert writes the reframed output – a 1080x1920 portrait MP4 with H.264 video and stereo AAC audio – to the Amazon S3 location that you choose.

Smart cropping in the playground is the same feature described in the AWS Elemental MediaConvert documentation. For how smart cropping works and the list of AWS Regions where it is available, see [Smart Cropping with Elemental Inference](#) in the AWS Elemental MediaConvert User Guide.

Topics

- [How the playground works](#)
- [Permissions](#)
- [Creating a playground job](#)
- [Viewing job results](#)
- [Pricing](#)
- [Troubleshooting](#)

How the playground works

The playground uses two resources in your account:

- An IAM service role named `ElementalInferencePlaygroundServiceRole`. You create this role from the **Create job** page before your first job. The role applies to every AWS Region.
- The first time you submit a job in an AWS Region, the console creates a dedicated MediaConvert queue named `ElementalInferencePlaygroundQueue`. This queue allows a maximum of one concurrent feed. As a result, playground jobs run one at a time and use no more than one feed from your Elemental Inference quota.

We recommend that you don't delete this queue, so that your job history remains available.

After these resources exist, each job that you submit does the following:

- The console submits an MediaConvert job.
- MediaConvert briefly creates and then deletes an Elemental Inference feed to analyze the video.
- MediaConvert writes the reframed output to your Amazon S3 location.

Jobs appear in the **Playground Jobs** table with a status of **Queued**, **Processing**, **Complete**, **Canceled**, or **Error**.

Permissions

The playground uses a service role to run each job. It also requires certain permissions for the IAM identity that you use to sign in.

Service role

The console creates the service role `ElementalInferencePlaygroundServiceRole` when you choose **Create service role** on the **Create job** page. You can create the role after you enter both the input and output locations, because its permissions are scoped to those buckets. Its trust policy allows AWS Elemental MediaConvert to assume the role (`sts:AssumeRole`). Its permissions are limited to the following:

- Elemental Inference – `CreateFeed`, `AssociateFeed`, `PutMedia`, `GetMetadata`, `DeleteFeed`, and `TagResource`
- Amazon S3 – `GetObject` on the input bucket and `PutObject` on the output bucket

If a later job uses input or output buckets that the role doesn't cover, the console prompts you to choose **Update role permissions**. Updating the role scopes it to the buckets that you selected for the new job, and removes access to the buckets it covered before. A queued job, or a duplicate of an earlier job, that uses those earlier buckets then needs the role updated again.

User permissions

In addition to permission to use the Elemental Inference console, the IAM identity that you use to sign in needs the following permissions to use the playground:

- MediaConvert – GetQueue, CreateQueue, CreateJob, ListJobs, GetJob
- iam:PassRole on the service role (MediaConvert requires this to run a job as the role)
- IAM read permissions – GetRole, ListAttachedRolePolicies, GetPolicy, GetPolicyVersion
- To let the console create the service role – iam:CreateRole, iam:CreatePolicy, iam:AttachRolePolicy
- To let the console update the service role – iam:CreatePolicyVersion, iam:DeletePolicyVersion
- Amazon S3 – ListAllMyBuckets, ListBucket, and GetObject on the input and output buckets (the console plays both videos from Amazon S3)

Creating a playground job

To submit a playground job, use the following procedure.

1. Open the Elemental Inference console at <https://console.aws.amazon.com/elemental-inference/>.
2. In the navigation pane, choose **Playground**, and then choose **Create job**.
3. Under **Input settings**, for **Input S3 URI**, enter the Amazon S3 URI of an MP4 file, or choose **Browse** and use **Choose a file** to select one. **Browse** lists only buckets in the console's current AWS Region. To use a bucket in another Region, enter its URI. To preview only part of the video, optionally enter **Start time** and **End time** in HH:MM:SS format.
4. Under **Output settings**, for **Destination**, enter the Amazon S3 folder URI where the output is written, or choose **Browse** and use **Choose a location**.
5. Under **AI feature**, review the feature that the job applies. **Smart Cropping** is the only AI feature available in the playground.
6. Under **Service role**, if the service role doesn't exist yet, choose **Create service role**. If it exists but doesn't cover the buckets you selected, choose **Update role permissions**.
7. Choose **Create job**. The job appears in the **Playground Jobs** table.

Viewing job results

After you submit a job, you can view its progress and results in the Elemental Inference console.

- In the **Playground Jobs** table, choose a job to open **Job details**.
- When the job is complete, the **Original video** and the **Smart crop output** play side by side.
- The following actions are available:
 - **Duplicate job** opens the **Create job** page with this job's settings filled in. Nothing is submitted until you choose **Create job**.
 - **View in MediaConvert** opens the job in the MediaConvert console.
 - **Back to jobs** returns to the list.
- If your browser can't play the video because its format or codec isn't supported, the job still succeeded. View the output in MediaConvert, or download it from Amazon S3.

Pricing

Playground jobs incur both AWS Elemental Inference and AWS Elemental MediaConvert charges, and you're charged only when a job completes successfully. There are no charges when a job fails. For pricing details, see [AWS Elemental Inference pricing](#) and [AWS Elemental MediaConvert pricing](#).

Troubleshooting

If you have trouble using the playground, review the following issues and solutions.

MediaConvert queue limit reached

If your account has reached its MediaConvert queue limit, the console can't create the playground queue. Delete an unused MediaConvert queue, or request a queue limit increase, and then try again.

The service role doesn't cover the selected bucket

If the service role doesn't grant access to the input or output location you chose, choose **Update role permissions** before you submit the job. The update removes the role's access to buckets that it covered before.

A video can't be previewed

If the output's format or codec isn't supported for browser playback, the preview can't play it. This doesn't affect the job; the file was processed successfully. Open it in MediaConvert, or download it from Amazon S3.

A job stays in Queued

Because the playground queue runs one job at a time, a new job waits in **Queued** until capacity is available, and then begins processing.

Work with existing Elemental Inference feeds

You can make the following changes to an existing feed:

- Edit a feed. You can change the configuration of its outputs, add outputs up to the maximum allowed in one feed, remove outputs, and enable or disable outputs. For information about the status of outputs, see [the section called "Status of an output"](#).
- Archive a feed.
- Delete a feed.

You can make these changes using the Elemental Inference console or the AWS CLI.

Topics

- [Editing, archiving, and deleting feeds using the console](#)
- [Revising using the CLI](#)

Editing, archiving, and deleting feeds using the console

To manage an existing feed, open the Elemental Inference console at <https://console.aws.amazon.com/elemental-inference/>. In the navigation pane, choose **Feeds**, and then select the feed to open its details page.

On the feed details page, take one of the following actions.

- To edit a feed – Choose **Actions, Edit**. The **Edit feed** page opens with the same sections as the create-feed page, except tags. To change tags, use the **Tags** tab on the feed details page. On the **Edit feed** page, each output can also be enabled or disabled, or removed with **Remove**. Make your changes, and then choose **Save**. The **Feed outputs** tab is read-only. To change a feed's outputs, edit the feed.

If you add a feature (an output), make sure that you have room in the [enabled outputs quota](#) for Elemental Inference. The list of quotas is sorted alphabetically. Look for quotas that don't start with "Request rate for".

An output that was created by an association, for example by an MediaLive channel, is managed by the associated resource. Editing it might affect the associated service.

- To archive a feed – Choose **Actions**, **Archive**, and then confirm. Archiving removes the feed's association. Archived feeds can no longer be updated or handle traffic, but you can still delete them. You can archive only active feeds that have an association.
- To delete a feed – Choose **Actions**, **Delete**, and then confirm. This permanently deletes the feed. You can't undo this action.

You can also archive or delete feeds directly from the **Feeds** list. Select one or more feeds, and then choose **Actions**.

 Note

Archived and deleted feeds can't be edited.

Revising using the CLI

To add or delete features in a feed, use the `AssociateFeed` or `UpdateFeed` action. `AssociateFeed` is a PATCH operation. `UpdateFeed` is a PUT operation.

For information about the differences between these two operations and about which operation to use for specific changes, see the information about these operations in the <https://docs.aws.amazon.com/elemental-inference/latest/APIReference/Welcome> AWS Elemental Inference API Reference.

Managing feed policies

A feed policy is a resource-based policy that you attach to an Elemental Inference feed to grant cross-account access. When you attach a policy to a feed, other AWS accounts and services can call `GetMetadata` on that feed without needing IAM credentials in your account.

A common use case for feed policies is integration with AWS Elemental MediaTailor. After you attach a policy that grants MediaTailor access, MediaTailor retrieves contextual metadata from the feed at ad break time and forwards it to the ad-decision server for contextual ad targeting.

Topics

- [Prerequisites](#)
- [Policy requirements](#)
- [Managing a policy using the console](#)
- [Managing policies using the CLI](#)
- [Example policy for MediaTailor access](#)
- [Security considerations](#)

Prerequisites

Before you can attach a policy to a feed, you must have the following:

- An existing Elemental Inference feed. For instructions on creating a feed, see [the section called "Create the feed"](#).
- The feed ID. You can find this in the Elemental Inference console on the feed details page, or from the response of the `CreateFeed` or `GetFeed` API operations.
- The AWS account ID of the service or account that you want to grant access.

Policy requirements

Elemental Inference validates a feed policy when you attach it, and rejects the request with a validation error if the policy does not meet the following requirements:

- The policy must include the `Version` and `Statement` elements.

- Each statement must include the `Sid`, `Effect`, `Principal`, `Action`, and `Resource` elements. The `Sid` value must be unique within the policy.
- The only supported action is `elemental-inference:GetMetadata`.
- The `Principal` element cannot be a wildcard (*).
- The policy cannot use the `NotAction`, `NotPrincipal`, or `NotResource` elements, and cannot specify a principal group.
- The policy document can be at most 2,048 characters.

The `Condition` element is optional. For guidance on the conditions to include, see [the section called "Security considerations"](#).

Managing a policy using the console

You can attach, update, and remove a feed's resource policy on the **Permissions** tab of the feed details page. When no policy is set, the **Resource policy** section shows the message **No resource policy configured**. Until you add a policy, only your AWS account can read the feed's metadata.

1. Open the Elemental Inference console at <https://console.aws.amazon.com/elemental-inference/>, and choose **Feeds**. Then select the feed.
2. Choose the **Permissions** tab. If the feed has a policy, the **Resource policy** section shows it in an inline JSON editor. If the feed doesn't have a policy, choose **Add resource policy** to open the editor.
3. To build a policy, edit the JSON directly, or choose **Add new statement** to insert a statement. You can add an example statement that grants the MediaTailor service read access to the feed's metadata. For an example policy, see [the section called "Example policy for MediaTailor access"](#).
4. To attach the policy, choose **Save**. To discard your changes, choose **Cancel**.
5. To remove the policy, choose **Delete policy**, and then confirm. Deleting the policy revokes any cross-account or service access that it granted.

Note

You can also set a resource policy when you create a feed. On the create-feed page, the **Resource policy** section provides **Add resource policy** and **Remove resource policy**.

Managing policies using the CLI

You can use the Elemental Inference API to attach, retrieve, and delete feed policies.

Attaching a policy (PutFeedPolicy)

Use PutFeedPolicy to attach or replace a resource-based policy on a feed. If a policy already exists on the feed, this operation replaces it.

```
aws elemental-inference put-feed-policy \
  --id <feed-id> \
  --policy '{"Version":"2012-10-17","Statement":
[{"Sid":"AllowMediaTailorGetMetadata","Effect":"Allow","Principal":
{"Service":"mediatailor.amazonaws.com"},"Action":"elemental-
inference:GetMetadata","Resource":"arn:aws:elemental-inference:<region>:<account-
id>:feed/<feed-id>","Condition":{"StringEquals":{"aws:SourceAccount":"<account-
id>"},"ArnLike":{"aws:SourceArn":"arn:aws:mediatailor:<region>:<account-
id>:playbackConfiguration/*"}}}]}'
```

Retrieving a policy (GetFeedPolicy)

Use GetFeedPolicy to retrieve the resource-based policy attached to a feed.

```
aws elemental-inference get-feed-policy \
  --id <feed-id>
```

The response contains the JSON policy document:

```
{
  "policy": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Sid\":
\"AllowMediaTailorGetMetadata\",\"Effect\":\"Allow\",\"Principal\":{\"Service\":
\"mediatailor.amazonaws.com\"},\"Action\":\"elemental-inference:GetMetadata\",
\"Resource\":\"arn:aws:elemental-inference:us-west-2:111122223333:feed/abc123\",
\"Condition\":{\"StringEquals\":{\"aws:SourceAccount\":\"111122223333\"},\"ArnLike\":
{\"aws:SourceArn\":\"arn:aws:mediatailor:us-west-2:111122223333:playbackConfiguration/*
\"}}}]}"
}
```

Deleting a policy (DeleteFeedPolicy)

Use `DeleteFeedPolicy` to remove the resource-based policy from a feed. After you delete the policy, accounts and services that the policy previously authorized can no longer access the feed.

```
aws elemental-inference delete-feed-policy \  
  --id <feed-id>
```

Example policy for MediaTailor access

The following example policy grants AWS Elemental MediaTailor permission to call `GetMetadata` on the specified feed. The policy uses two conditions to prevent the *confused deputy* problem:

- `aws:SourceAccount` – Ensures only MediaTailor operating on behalf of your account can access the feed.
- `aws:SourceArn` – Further restricts access to specific playback configurations. Replace the wildcard (*) with a specific playback configuration name for tighter scoping.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "AllowMediaTailorGetMetadata",  
      "Effect": "Allow",  
      "Principal": {  
        "Service": "mediatailor.amazonaws.com"  
      },  
      "Action": "elemental-inference:GetMetadata",  
      "Resource": "arn:aws:elemental-inference:region:account-id:feed/feed-id",  
      "Condition": {  
        "StringEquals": {  
          "aws:SourceAccount": "account-id"  
        },  
        "ArnLike": {  
          "aws:SourceArn": "arn:aws:mediatailor:region:account-id:playbackConfiguration/*"  
        }  
      }  
    }  
  ]  
}
```

```
]
}
```

Replace the following placeholder values:

- *region* – The AWS Region where the feed and playback configuration exist (for example, us-west-2).
- *account-id* – Your AWS account ID (the account that owns both the feed and the MediaTailor playback configuration).
- *feed-id* – The ID of the Elemental Inference feed.

Note

For opt-in AWS Regions, use the regionalized service principal `mediatailor.region.amazonaws.com` instead of `mediatailor.amazonaws.com`.

Security considerations

When creating feed policies, follow these security best practices:

- **Always include a source account condition** – Include the `aws:SourceAccount` condition in every statement that grants access to an AWS service principal. This prevents the *confused deputy* problem, where a service could be tricked into accessing your feed on behalf of an unauthorized account. Where possible, also include `aws:SourceArn` to restrict access to specific resources in your account.
- **Do not use wildcard principals** – Elemental Inference rejects policies with wildcard (*) principals. Specify the exact AWS service or account principal that requires access.
- **Scope permissions to specific actions** – Grant only the actions that the consuming service needs. For MediaTailor integration, `elemental-inference:GetMetadata` is the only supported action.
- **Scope the resource** – Specify the exact feed ARN in the Resource field rather than using a wildcard. This ensures the policy grants access only to the intended feed.

Monitoring Elemental Inference activity

You can monitor activity in Elemental Inference on the AWS Elemental Inference console or using the AWS CLI or an AWS API or SDK.

In addition, implements the monitoring capabilities of several AWS services, identified in the following topics.

Topics

- [Lifecycle of an AWS Elemental Inference workflow](#)
- [Monitoring AWS Elemental Inference on the console](#)
- [Monitoring AWS Elemental Inference using the AWS CLI](#)
- [Monitoring AWS Elemental Inference with Amazon CloudWatch](#)
- [Monitoring AWS Elemental Inference events in Amazon EventBridge](#)
- [Logging Elemental Inference API calls with AWS CloudTrail](#)

Lifecycle of an AWS Elemental Inference workflow

When you use Elemental Inference, you create a feed, include one or more outputs, and associate a resource. Each output represents one Elemental Inference feature, such as smart crop. Each resource represents the source media that works with the feed.

Lifecycle and status of a feed

The typical lifecycle of a feed is as follows:

CREATING -> AVAILABLE -> ACTIVE -> ARCHIVED -> DELETING -> DELETED

All of these statuses are one-way. Most importantly, you can't unarchive a feed. Typically the status changes as follows:

- CREATING to AVAILABLE. This is always the initial transition.
- AVAILABLE means that the feed doesn't have a resource associated with it.
- AVAILABLE to ACTIVE. After you associate the resource, the feed becomes ACTIVE.

ACTIVE means that you can stream media to the feed, using PutMedia.

ACTIVE doesn't mean that you've called PutMedia and the feed is processing the media. Elemental Inference doesn't monitor streaming because each PutMedia call occurs in milliseconds, which means it is impossible to track in a meaningful way.

- ACTIVE to ARCHIVED. When you no longer need the feed, you can archive it in one of these ways:
 - On the console, display the feed details and choose **Archive**.
 - On the CLI, use the DisassociateFeed operation, which indirectly sets the status to ARCHIVED.

When the feed is ARCHIVED, you can't use it any more. Your quota for feeds decreases by 1.

- When you are ready, delete the feed. The feed will disappear from the array of lists, so you won't actually see it with a DELETED status.

Status of a resource

A resource doesn't have a status. Keep in mind that the resource is really just a way of identifying that the feed is intended for a specific source media.

Status of an output

Related to the feed status, the output also has a status:

- ENABLED means that Elemental Inference will generate metadata for the output when it is processing the source media.
- DISABLED means that it won't generate new metadata.

If you create an output using the console, its initial status is ENABLED. If you create it using the CLI, you choose the initial status.

You can change the output status when a feed is any of the statuses before ARCHIVED. You can change the status at any time, including when the source media is being streamed to the feed (in other words, you've called PutMedia).

You can't change the status of the output after the feed is ARCHIVED. There is never any need to do this.

Lifecycle of metadata

Elemental Inference produces metadata for a feature type (for example, smart crop) when media is being streamed to the feed and the output for that feature is **ENABLED**.

Elemental Inference retains metadata for 7 days, for both **ENABLED** and **DISABLED** outputs.

It continually discards data that is older than 7 days, for both **ENABLED** and **DISABLED** outputs.

Monitoring AWS Elemental Inference on the console

You can monitor a feed using the Elemental Inference console.

1. On the Elemental Inference console, in the navigation pane, choose **Feeds**.
2. The **Feeds** page shows a list of your feeds. Each line in the list provides basic information about the feed, including its status. For information about statuses, see [the section called “Lifecycle of a feed”](#).
3. To view more details about a feed, choose the name of that feed. The **Feed details** page appears. Information appears, as described in the following sections.

General details panel and feed association panel

Feed ID

The ID of the Elemental Inference feed. The ID is identical to the last portion of the ARN of the feed.

ARN

The ARN of the feed.

Status of the feed

These statuses are listed in lifetime order, from **CREATING** to **ARCHIVED**. Note the following:

- A newly created feed typically transitions immediately from **CREATING** to **AVAILABLE** to **ACTIVE**. **ACTIVE** means that the feed is associated with its resource.
- When Elemental Inference deletes a feed, its status changes to **DELETED**, then after a short period, it changes to **ARCHIVED**. There is no way to change the status of a feed that is **DELETED** or **ARCHIVED**.

Monitoring tab

The **Monitoring** tab is the first tab on the feed details page.

Until the feed has an association, it is not active. While the feed is not active, the **Monitoring** tab shows a message that prompts you to add an association so that you can monitor the feed.

To add an association from the feed details page, choose **Add association**.

When the feed is active or archived, the **Monitoring** tab shows the **Feed dashboard**. The dashboard is a set of Amazon CloudWatch graphs that show live data for the feed:

- **PutMedia request count**
- **PutMedia request latency**
- **GetMetadata request latency**
- **Feature processing latency** – One graph, with a line for each feature (event clipping, smart cropping, smart subtitling, and contextual metadata)

Feed outputs tab

In this tab, one panel appears for each feature that you have enabled in the channel. Each panel includes the following information:

- The output status. For more information about status, see [the section called “Lifecycle of a feed”](#).
- From association: A value of true means that the output was created using the AssociateFeed operation.

If your organization uses AWS Elemental MediaLive to set up Elemental Inference features in a channel, a value of true indicates that you used MediaLive to create the feed and the output.

Preview metadata

Each output card on the **Feed outputs** tab includes an expandable **Preview metadata** section. This section shows the metadata that the output has recently produced. Preview is available for smart cropping, event clipping, smart subtitling, and contextual metadata outputs.

The feed must be active to preview metadata. While the feed is not active, the console shows the message *Feed must be active to preview metadata*.

By default, the preview reads a window of the most recent metadata, just behind the live edge. To set how far back to read, choose a **Time range: Last 10 seconds** (the default), **Last 20 seconds**, or **Last 30 seconds**.

To read from a specific point on the media timeline instead, turn on **Use a fixed start PTS**. PTS (presentation timestamp) is the media-timeline position that metadata is stamped with. For more information, choose **What is PTS?** in the console. For example, use this option for video-on-demand content that starts near PTS 0. Enter a **Start PTS** value in timescale units. The window length control then changes to **Duration**.

You can also set the following options:

- **Input** – The feed input to read from (0 or 1).
- **Timescale** – The number of timescale units (ticks) per second used for PTS values. The default is 90000.

To load the metadata, choose **Apply**. Results appear as a **Table** or as **JSON**. The columns that appear depend on the output type.

Monitoring AWS Elemental Inference using the AWS CLI

To monitor feeds and outputs using an AWS API, use the `GetFeed` operation of Elemental Inference. For more information, see [GetFeed](#) in the *AWS Elemental Inference API Reference*.

For information about how the feed and outputs change status through the lifetime of the feed, see [the section called “Lifecycle of a feed”](#).

Monitoring AWS Elemental Inference with Amazon CloudWatch

You can monitor AWS Elemental Inference using CloudWatch. CloudWatch collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met.

For more information about using CloudWatch with data, see the [Amazon CloudWatch User Guide](#).

For information about the metrics that CloudWatch produces from CloudWatch data, see the sections that follow.

Topics

- [Components of a metric](#)
- [Pricing to view Elemental Inference metrics](#)
- [Service availability and API metrics](#)
- [Elemental Inference performance metrics](#)

Components of a metric

AWS Elemental Inference collects data that is the basis for metrics. It collects these *datapoints* every second and sends them immediately to Amazon CloudWatch. You can use CloudWatch to generate *metrics* for these datapoints.

A metric is a collection of datapoints that has had an aggregation (a *statistic*) applied and that has a *period* and a *time range*. For example, you can request the Dropped frames metric as a sum (the statistic) for a 1-minute period over 10 minutes (the time range). This result of this request is 10 metrics (because the range divided by the period is 10).

Elemental Inference supports all the statistics offered by CloudWatch. However, some statistics aren't useful for Elemental Inference metrics. In the description of metrics later in this chapter, we include the recommended statistics for each metric.

Each Elemental Inference metric includes one or two specific sets of dimensions.

Pricing to view Elemental Inference metrics

For information about charges to view metrics on the CloudWatch console or to retrieve metrics using a CloudWatch API, see the [Amazon CloudWatch User Guide](#).

Service availability and API metrics

API Latency

The duration of API calls from request initiation to response completion.

Details

- Name: ApiLatency
- Supported dimension sets: Feed

- Recommended statistic: p50, p95, p99, Average
- Units: Milliseconds
- Meaning of zero: Elemental Inference throttled requests because of request volume was too high. Typically, API requests fail with error code 429.
- Meaning of no datapoints: There were no API calls on the specified feed.

API Request Count

The number of API calls made.

Details

- Name: ApiRequestCount
- Supported dimension sets:
 - Feed
 - Feed, StatusCode: to monitor API requests that result in a specific category HTTP response codes.
- Recommended statistic: Sum
- Meaning of zero: No API requests were made to the specified feed during the time period.
- Meaning of no datapoints: There were no API calls on the specified feed.

Elemental Inference performance metrics

GetMetadata request count

The number of GetMetadata API requests made to retrieve metadata.

Details

- Name: GetMetadataRequestCount
- Supported dimension sets:
 - Feed
 - Feed, StatusCode: to monitor API requests that result in a specific category HTTP response codes.
- Recommended statistic: Sum

- Units: Count
- Meaning of zero: This metric will never emit zero.
- Meaning of no datapoints: There were no GetMetadata API calls on the specified feed.

Processing fault count

The total number of processing failures.

Details

- Name: ProcessingFaultCount
- Supported dimension sets: Feed
- Recommended statistic: Sum
- Units: Count
- Meaning of zero: All video segments submitted to the feed were processed successfully without any failures.
- Meaning of no datapoints: No video segments were submitted. (PutMedia hasn't been called.)

Processing latency

The end-to-end time from when a video segment is submitted using PutMedia until the metadata is available to be retrieved using GetMetadata.

Details

Name: ProcessingLatency

Supported dimension sets:

- Feed
- Feed, Feature

Recommended statistics: p50, p95, p99, Average

Units: Milliseconds

Meaning of zero: There is probably an issue with the source media. For example, if the timestamp is wrong, it's possible that the latency will seem to be 0.

Meaning of no datapoints: No video segments were submitted (PutMedia hasn't been called.)

PutMedia request count

The number of PutMedia API requests made to submit video segments for processing.

Details

- Name: PutMediaRequestCount
- Supported dimension sets:
 - Feed,
 - Feed, StatusCode: to monitor API requests that result in a specific category HTTP response codes.

Recommended statistic: Sum

- Units: Count
- Meaning of zero: This metric will never emit zero.
- Meaning of no datapoints: There were no PutMedia API calls on the specified feed.

Monitoring AWS Elemental Inference events in Amazon EventBridge

Elemental Inference automatically turns information about event clipping metadata into events in EventBridge. You can use Amazon EventBridge to manage these events. For example, you can create event rules and deliver the events in emails or SMS messages. You can deliver events to a number of destinations.

For information about the specific events in Elemental Inference that are sent to EventBridge, see the sections that follow.

For information about working with AWS Elemental Inference events in EventBridge, see [Amazon EventBridge user guide](#).

Topics

- [Clip Metadata Generated event](#)

Clip Metadata Generated event

This event is published when Elemental Inference detects an interesting event in a stream that you are pushing to Elemental Inference. The event includes the following information:

- **resource**: The ARN of the Elemental Inference feed that this clipping event belongs to.
- **timescale**
- **startPts** and **endPts**: The start and end presentation time stamp in the source video. This range identifies the start and end of the interesting event.
- **description**: The field is optional.
- **tags**

The following message is an example of this event.

```
{
  "version": "0",
  "id": "12345678-1234-1234-1234-123456789012",
  "detail-type": "Clip Metadata Generated",
  "source": "aws.elemental-inference",
  "account": "111122223333",
  "time": "2026-03-09T08:41:11Z",
  "region": "us-west-2",
  "resources": [
    "arn:aws:elemental-inference:us-west-2:feed/vbphju6m7nohlpcs3sd"
  ],
  "detail": {
    "timescale": 90000,
    "startPts": 159574109479500,
    "endPts": 159574109560500,
    "description": "",
    "tags": [
      "goal"
    ]
  }
}
```

Logging Elemental Inference API calls with AWS CloudTrail

AWS Elemental Inference is integrated with AWS CloudTrail. *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

Elemental Inference information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Elemental Inference, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see [Viewing Events with CloudTrail Event History](#).

For an ongoing record of events in your AWS account, including events for Elemental Inference, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- [Overview for Creating a Trail](#)
- [CloudTrail Supported Services and Integrations](#)
- [Configuring Amazon SNS Notifications for CloudTrail](#)
- [Receiving CloudTrail Log Files from Multiple Regions](#) and [Receiving CloudTrail Log Files from Multiple Accounts](#)

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail userIdentity Element](#).

Understanding Elemental Inference log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry. The example shows the entry for one API call. The call is made by the identity specified in `userIdentity`, in this case a user who has assumed the Administrator role with the session name `santosp`. The call was a `CreateFeed` operation coming from the AWS CLI (as specified in `userAgent`) running on a computer with the IP address `203.0.113.33`

```
{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROEXAMPLEID:santosp",
    "arn": "arn:aws:sts::111122223333:assumed-role/Administrator/santosp",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROEXAMPLEID",
        "arn": "arn:aws:iam::111122223333:role/Administrator",
        "accountId": "111122223333",
        "userName": "Administrator"
      },
      "attributes": {
        "creationDate": "2024-11-14T19:00:28Z",
        "mfaAuthenticated": "false"
      }
    }
  },
  "eventTime": "2024-11-14T19:00:29Z",
  "eventSource": "elemental-inference.amazonaws.com",
  "eventName": "CreateFeed",
  "awsRegion": "us-west-2",
```

```

"sourceIPAddress": "203.0.113.33",
"userAgent": "aws-cli/2.32.22 md/awscrt#0.29.1 ua/2.1 os/
linux#6.1.159-182.297.amzn2023.x86_64 md/arch#x86_64 lang/python#3.9.25 md/
pyimpl#CPython cfg/retry-mode#standard md/installer#source md/distrib#amzn.2023 md/
prompt#off md/command#elemental-inference.create-feed",
"requestParameters": {
  "outputs": [
    {
      "name": "out1",
      "outputConfig": {
        "cropping": {}
      },
      "status": "ENABLED"
    }
  ],
  "name": "live-studio-feed",
  "x-amzn-client-token": "1111aaaa-e840-4416-8ee8-a95d5ed7c031"
},
"responseElements": {
  "outputs": [
    {
      "name": "out1",
      "outputConfig": {
        "cropping": {}
      },
      "status": "ENABLED"
    }
  ],
  "dataEndpoints": [
    "https://abc123example.elemental-inference-data.us-west-2.amazonaws.com/"
  ],
  "name": "live-studio-feed",
  "id": "abc123example",
  "arn": "arn:aws:elemental-inference:us-west-2:111122223333:feed/abc123example",
  "status": "CREATING",
  "tags": {}
},
"requestID": "d2f882ac-1a9d-11e9-a0e5-afe6a8c88993",
"eventID": "ebbe0290-7a1b-4053-a219-367404e0fe96",
"readOnly": false,
"resources": [
  {
    "accountId": "111122223333",
    "type": "AWS::Elemental-Inference::Feed",

```

```
    "ARN": "arn:aws:elemental-inference:us-west-2:111122223333:feed/abc123example"
  }
],
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "111122223333",
"eventCategory": "Management"
}
```

Security in AWS Elemental Inference

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS compliance programs](#). To learn about the compliance programs that apply to AWS Elemental Inference, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Elemental Inference. The following topics show you how to configure Elemental Inference to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Elemental Inference resources.

Topics

- [Data protection in AWS Elemental Inference](#)
- [Identity and Access Management for AWS Elemental Inference](#)
- [Compliance validation for AWS Elemental Inference](#)
- [Resilience in AWS Elemental Inference](#)
- [Infrastructure security in AWS Elemental Inference](#)

Data protection in AWS Elemental Inference

The AWS [shared responsibility model](#) applies to data protection in AWS Elemental Inference. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of

the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see [Data Privacy FAQ](#). For information about data protection in Europe, see the [General Data Protection Regulation \(GDPR\) Center](#).

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see [Working with CloudTrail trails](#) in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-3](#).

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Elemental Inference or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Identity and Access Management for AWS Elemental Inference

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Elemental Inference resources. IAM is an AWS service that you can use with no additional charge.

Topics

- [Audience](#)
- [Authenticating with identities](#)
- [Managing access using policies](#)

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access feature
- **Service administrator** - determine user access and submit permission requests
- **IAM administrator** - write policies to manage access

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see [How to sign in to your AWS account](#) in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see [AWS Signature Version 4 for API requests](#) in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see [Tasks that require root user credentials](#) in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see [What is IAM Identity Center?](#) in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An [IAM user](#) is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see [Require human users to use federation with an identity provider to access AWS using temporary credentials](#) in the *IAM User Guide*.

An [IAM group](#) specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see [Use cases for IAM users](#) in the *IAM User Guide*.

IAM roles

An [IAM role](#) is an identity with specific permissions that provides temporary credentials. You can assume a role by [switching from a user to an IAM role \(console\)](#) or by calling an AWS CLI or AWS API operation. For more information, see [Methods to assume a role](#) in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see [Cross account resource access in IAM](#) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see [Overview of JSON policies](#) in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Define custom IAM permissions with customer managed policies](#) in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see [Choose between managed policies and inline policies](#) in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must [specify a principal](#) in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see [Permissions boundaries for IAM entities](#) in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see [Service control policies](#) in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see [Resource control policies \(RCPs\)](#) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see [Session policies](#) in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see [Policy evaluation logic](#) in the *IAM User Guide*.

Compliance validation for AWS Elemental Inference

Our new AWS sign-up experience is not designed for regulated workloads. If you're using our new AWS sign-up experience, but you want to use AWS for regulated workloads, you can [sign up for AWS \(advanced\)](#) or [activate advanced features](#) for your AWS environment.

To learn whether an AWS service is within the scope of specific compliance programs, see [AWS services in Scope by Compliance Program](#) and choose the compliance program that you are interested in. For general information, see [AWS Compliance Programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. For more information about your compliance responsibility when using AWS services, see [AWS Security Documentation](#).

Resilience in AWS Elemental Inference

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

Infrastructure security in AWS Elemental Inference

As a managed service, AWS Elemental Inference is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see [AWS Cloud Security](#). To design your AWS environment using the best practices for infrastructure security, see [Infrastructure Protection](#) in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Elemental Inference through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Document history for user guide

The following table describes the documentation for this release of AWS Elemental Inference

- **API version: latest**

Change	Description	Date
Playground and console updates	Added the Elemental Inference playground for previewing smart cropping on recorded video. Updated the console procedures for creating, editing, archiving, and deleting feeds; managing feed resource policies; monitoring feeds; and managing dictionaries.	September 28, 2026
Contextual metadata and feed policies	Added support for contextual metadata, which uses AI to generate shot-level and scene-level content classifications (IAB Content Taxonomy v3.1 categories and GARM brand safety ratings) and descriptive labels for contextual ad targeting . Added feed policies for granting cross-account access to feeds.	July 21, 2026
New information	New information has been added to all sections of this guide.	April 8, 2026

New service and guide	This is the initial release of the AWS Elemental Inference service and the AWS Elemental Inference User Guide.	February 24, 2026
Infrastructure security	The information in this section has been revised. Specifically, we now require TLS 1.2 and we recommend TLS 1.3.	June 24, 2023
Data protection	The information in this section has been revised. Specifically, we now require TLS 1.2 and we recommend TLS 1.3.	June 24, 2023
AWS Identity and Access Management	Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM .	February 14, 2023