《用户指南》

AWS Elemental MediaStore

AWS Elemental MediaStore: 《用户指南》

Table of Contents

终止支持通知：2025 年 11 月 13 日，我 AWS 将停止对 AWS MediaStore Elemental 的支持。2025 年 11 月 13 日之后，您将无法再访问 MediaStore 控制台或 MediaStore 资源。有关更多信息，请访问此博客文章。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。

什么是 AWS Elemental MediaStore？

AWS Elemental MediaStore 是一项视频制作和存储服务，可提供直播制作所需的高性能和即时一致性。借 MediaStore助，您可以将视频资产作为容器中的对象进行管理，以构建基于云的可靠媒体工作流程。

要使用该服务，可以将对象从源（如编码器或数据源）上传到您在 MediaStore 中创建的容器。

MediaStore 当您需要强一致性、低延迟读取和写入以及处理大量并发请求的能力时，是存储碎片视频文件的绝佳选择。如果您不提供直播视频，可以考虑改用 Amazon Simple Storage Service (Amazon S3)。

主题

AWS Elemental 的 MediaStore 概念和术语

ARN

Amazon 资源名称。

Body

要上传到对象中的数据。

（字节）范围

要寻址的对象数据的子集。有关更多信息，请参阅 HTTP 规范中的范围。

容器

包含对象的命名空间。容器有一个可用于写入和检索对象以及附加访问策略的终端节点。

终端节点

MediaStore 服务的入口点，以 HTTPS 根网址的形式给出。

ETag

实体标签，这是对象数据的哈希。

文件夹

容器的分区。文件夹可以包含对象和其他文件夹。

Item

用于指代对象和文件夹的术语。

对象

资产，类似于 Amazon S3 对象。对象是 MediaStore 中存储的基本实体。该服务接受所有文件类型。

源服务

MediaStore 被视为发起服务，因为它是媒体内容交付的分发点。

路径

对象或文件夹的唯一标识符，用于指示对象或文件夹在容器中的位置。

零件

对象的数据（数据块）的子集。

策略

IAM 策略。

资源

您可以在 AWS 中使用的实体。每个 AWS 资源都会分配一个作为唯一标识符的 Amazon 资源名称 (ARN)。在中 MediaStore，这是资源及其 ARN 格式：

- 容器：`aws:mediastore:region:account-id:container/:containerName`

相关服务

- Amazon CloudFront 是一项全球内容分发网络 (CDN) 服务，可将数据和视频安全地传送给您的观众。使用 CloudFront 以最佳的性能分发内容。有关更多信息，请参阅 Amazon CloudFront 开发者指南。
- CloudFormation是一项可帮助您建模和设置 AWS 资源的服务。您可以创建一个描述所需的所有 AWS 资源（如 MediaStore 容器）的模板，并 CloudFormation 负责为您配置和配置这些资源。您无

需单独创建和配置 AWS 资源，也不需要弄清楚哪些资源依赖于什么； CloudFormation 可以处理所有这些。有关更多信息，请参阅 AWS CloudFormation 《用户指南》。

- AWS CloudTrail是一项服务，可让您监控对账户的 CloudTrail API 进行的调用，包括 AWS 管理控制台和其他服务发出的调用。 AWS CLI有关更多信息，请参阅 AWS CloudTrail 《用户指南》。
- Amazon CloudWatch 是一项监控 AWS 云资源和您运行的应用程序的服务 AWS。使用 “CloudWatch 事件” 来跟踪容器和中对象状态的变化 MediaStore。有关更多信息，请参阅 Amazon CloudWatch 文档。
- AWS Identity and Access Management (IAM) 是一项 Web 服务，可帮助您安全地控制用户对 AWS 资源的访问权限。使用 IAM 控制谁可以使用您的 AWS 资源（身份验证）以及用户可以通过哪些方式使用哪些资源（授权）。有关更多信息，请参阅 设置 AWS Elemental MediaStore。
- Amazon Simple Storage Service (Amazon S3)是一种对象存储，旨在从任何地方存储和检索任意数量的数据。有关更多信息，请参阅 Amazon S3 文档。

访问 AWS Elemental MediaStore

您可以使用以下任何一种方法 MediaStore 进行访问：

- AWS 管理控制台-本指南中的程序说明了如何使用 AWS 管理控制台执行任务 MediaStore。要 MediaStore 使用控制台进行访问，请执行以下操作：

```
https://<region>.console.aws.amazon.com/mediastore/home
```

- AWS Command Line Interface – 有关更多信息，请参阅 AWS Command Line Interface 用户指南。要 MediaStore 使用 CLI 端点进行访问，请执行以下操作：

```
aws mediastore
```

- MediaStore API — 如果您使用的编程语言不适用于 SDK，请参阅 AWS Elemental MediaStore API 参考以了解有关 API 操作以及如何发出 API 请求的信息。要 MediaStore 使用 REST API 端点进行访问，请执行以下操作：

```
https://mediastore.<region>.amazonaws.com
```

- AWS SDKs — 如果您使用的是 AWS 提供软件开发工具包的编程语言，则可以使用软件开发工具包进行访问 MediaStore。 SDKs 简化身份验证，轻松与开发环境集成，并提供对 MediaStore 命令的便捷访问。有关更多信息，请参阅用于 Amazon Web Services 的工具。

- 适用于 Windows 的 AWS 工具 PowerShell — 有关更多信息，请参阅AWS Tools for PowerShell 用户指南。

AWS Elemental 的定价 MediaStore

与其他 AWS 产品一样，没有合同或最低使用承诺 MediaStore。在内容进入服务后，将向您收取每 GB 的引入费用以及存储在服务中的每 GB 内容的月度费用。有关更多信息，请参阅 AWS Elemental MediaStore 定价。

AWS Elemental 的区域和终端节点 MediaStore

为了减少应用程序中的数据延迟，请 MediaStore 提供区域终端节点来提出您的请求：

```
https://mediastore.<region>.amazonaws.com
```

要查看可用的 AWS 区域的完整列表，请参阅 MediaStore AWS 一般参考中的 AWS Elemental MediaStore 终端节点和配额。

设置 AWS Elemental MediaStore

本节将指导您完成配置用户以访问 AWS Element MediaStore al 所需的步骤。有关身份和访问管理的背景信息和其他信息 MediaStore，请参阅适用于 AWS Elemental 的身份和访问管理 MediaStore。

要开始使用 AWS Elemental MediaStore，请完成以下步骤。

主题

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明操作。

   在注册时，将接到电话或收到短信，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为最佳安全实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。您可以随时前往 https://aws.amazon.com/并选择“我的账户”，查看您当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS 管理控制台在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的 Signing in as the root user。
2. 为您的根用户启用多重身份验证（MFA）。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 Enabling AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 Create a permission set。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 Add groups。

AWS Elemental 入门 MediaStore

本入门教程向您展示了如何使用 AWS Elemental MediaStore 创建容器和上传对象。

主题

第 1 步：访问 AWS Elemental MediaStore

设置 AWS 账户并创建用户和角色后，即可登录 AWS Element MediaStore al 的控制台。

访问 AWS Elemental MediaStore

- 登录 AWS 管理控制台 并打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。

> **Note**
>
> 可以使用已为此账户创建的任何 IAM 凭证登录。有关创建 IAM 凭证的信息，请参阅设置 AWS Elemental MediaStore。

步骤 2：创建容器

您可以在 AWS Elemental 中使用容器 MediaStore 来存储您的文件夹和对象。可以使用容器为相关对象分组，方式与在文件系统中使用目录为文件分组如出一辙。创建容器时不会产生费用，仅当将对象上传到容器时需要付费。

创建容器

1. 在 Containers (容器) 页面上，选择 Create container (创建容器)。
2. 对于 Container name (容器名称)，键入容器的名称。有关更多信息，请参阅 容器命名规则。

3. 选择 创建容器。AWS Elemental MediaStore 将新容器添加到容器列表中。最初，容器状态为 Creating (正在创建)，之后状态变为 Active (活跃)。

步骤 3：上传对象

您可以将对象（每个对象最多 25 MB）上传到容器或容器内的文件夹中。要将对象上传到文件夹，可以指定至文件夹的路径。如果该文件夹已经存在，AWS Elemental 会将该对象 MediaStore 存储在文件夹中。如果文件夹不存在，则该服务将创建文件夹，然后将对象存储在其中。

> **Note**
>
> 对象文件名只能包含字母、数字、句点 (.)、下划线 (_)、波形符 (~) 或连字符 (-)。

上传对象

1. 在 Containers (容器) 页面上，选择刚创建的容器的名称。将出现容器的详细信息页面。
2. 选择 Upload object (上传对象)。
3. 对于 Target path (目标路径)，键入文件夹的路径。例如 `premium/canada`。如果路径中的任何文件夹尚不存在，AWS Elemental MediaStore 会自动创建它们。
4. 对于 Object (对象)，选择 Browse (浏览)。
5. 导航到相应文件夹，然后选择要上传的对象。
6. 选择 Open (打开)，然后选择 Upload (上传)。

步骤 4：访问对象

可以将对象下载到指定终端节点。

1. 在 Containers (容器) 页面上，选择包含要下载的对象的容器的名称。
2. 如果要下载的对象位于子文件夹中，则继续选择文件夹名称，直到看到该对象。
3. 选择对象的名称。
4. 在对象的详细信息页面上，选择 Download (下载)。

AWS Elemental 中的容器 MediaStore

您可以使用中的容器 MediaStore 来存储您的文件夹和对象。相关对象可在容器中进行分组，方式与您在文件系统中使用目录对文件进行分组如出一辙。创建容器时不会产生费用，仅当将对象上传到容器时需要付费。有关费用的更多信息，请参阅 AWS Elemental MediaStore 定价。

主题

容器命名规则

当您选择容器的名称时，请记住以下要求：

- 此名称在当前 Amazon Web Services 区域的当前账户中必须是唯一的。
- 名称可包含大写字母、小写字母、数字和下划线 (_)。
- 名称长度必须介于 1 到 255 个字符之间。
- 名称区分大小写。例如，您可以将容器命名为 `myContainer`，将文件夹命名为 `mycontainer`，因为这些名称是唯一的。
- 容器在创建后不可重命名。

创建容器

您可以为每个 Amazon Web Services 账户创建最多 100 个容器。您可以创建任意数量的文件夹，只要它们在一个容器中嵌套不超过 10 个级别。此外，您还可以向每个容器上传任意数量的对象。

> **Tip**
>
> 您也可以使用 CloudFormation 模板自动创建容器。该 CloudFormation 模板管理五个 API 操作的数据：创建容器、设置访问日志记录、更新默认容器策略、添加跨源资源共享 (CORS) 策略以及添加对象生命周期策略。有关更多信息，请参阅 AWS CloudFormation 《用户指南》。

创建容器（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择 Create container (创建容器)。
3. 对于 Container (容器) 名称，输入容器的名称。有关更多信息，请参阅 容器命名规则。
4. 选择 创建容器。AWS Elemental MediaStore 将新容器添加到容器列表中。最初，容器状态为 Creating (正在创建)，之后状态变为 Active (活跃)。

创建容器 (AWS CLI)

- 在中 AWS CLI，使用以下`create-container`命令：

```
aws mediastore create-container --container-name ExampleContainer --region us-
west-2
```

以下示例显示了返回值：

```
{
    "Container": {
        "AccessLoggingEnabled": false,
        "CreationTime": 1563557265.0,
        "Name": "ExampleContainer",
        "Status": "CREATING",
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer"
    }
}
```

查看有关容器的详细信息

容器的详细信息包括容器策略、终端节点、ARN 和创建时间。

查看容器的详细信息（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择容器的名称。

此时将显示容器详细信息页面。此页面分为两个部分：

- Objects (对象) 部分，其中列出了容器中的对象和文件夹。
- Container (容器) 策略部分，其中显示了与此容器关联的基于资源的策略。有关资源策略的信息，请参阅容器策略。

查看容器的详细信息 (AWS CLI)

- 在中 AWS CLI，使用以下`describe-container`命令：

```
aws mediastore describe-container --container-name ExampleContainer --region us-west-2
```

以下示例显示了返回值：

```
{
    "Container": {
        "CreationTime": 1563558086.0,
        "AccessLoggingEnabled": false,
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
        "Status": "ACTIVE",
        "Name": "ExampleContainer",
        "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com"
    }
}
```

查看容器列表

您可以查看与您的账户关联的所有容器的列表。

查看容器列表（控制台）

- 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。

  此时将显示 Containers (容器) 页面，其中列出了与您的账户关联的所有容器。

查看容器列表 (AWS CLI)

- 在中 AWS CLI，使用`list-containers`命令。

```
aws mediastore list-containers --region us-west-2
```

以下示例显示了返回值：

```
{
    "Containers": [
        {
            "CreationTime": 1505317931.0,
            "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleLiveDemo",
            "AccessLoggingEnabled": false,
            "Name": "ExampleLiveDemo"
        },
        {
            "CreationTime": 1506528818.0,
            "Endpoint": "https://fffggghhhiiijj.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
            "AccessLoggingEnabled": false,
            "Name": "ExampleContainer"
        }
    ]
}
```

删除容器

您只能在容器中没有对象时才能将其删除。

删除容器（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。

2. 在 Containers (容器) 页面上，选择容器名称左侧的选项。
3. 选择删除。

删除容器 (AWS CLI)

- 在中 AWS CLI，使用以下`delete-container`命令：

```
aws mediastore delete-container --container-name=ExampleLiveDemo --region us-west-2
```

此命令没有返回值。

AWS Elemental 中的政策 MediaStore

您可以将以下一项或多项策略应用于您的 AWS Elemental 容器 MediaStore ：

- 容器策略-设置对容器内所有文件夹和对象的访问权限。 MediaStore 设置默认策略，允许用户对容器执行所有 MediaStore 操作。此策略指定必须通过 HTTPS 执行所有操作。创建容器后，您可以编辑容器策略。
- 跨源资源共享 (CORS) 策略-允许来自一个域的客户端 Web 应用程序与另一个域中的资源进行交互。 MediaStore 未设置默认 CORS 策略。
- 指标政策- MediaStore 允许向 Amazon 发送指标 CloudWatch。 MediaStore 未设置默认指标策略。
- 对象生命周期策略-控制对象在 MediaStore 容器中保留多长时间。 MediaStore未设置默认的对象生命周期策略。

AWS Elemental 中的容器策略 MediaStore

每个容器都有一个基于资源的策略，该策略管理对该容器中的所有文件夹和对象的访问权限。默认策略会自动附加到所有新容器，允许访问容器上的所有 AWS Elemental MediaStore 操作。它指定此访问具有对于操作需要 HTTPS 的条件。在创建容器之后，您可以编辑附加到该容器的策略。

您还可以指定对象生命周期策略，此策略用于管理容器中对象的过期时间。在对象达到您指定的最长时间之后，服务将从容器中删除对象。

主题

- 查看容器策略
- 编辑容器策略
- 示例容器策略

查看容器策略

您可以使用控制台或 AWS CLI 查看容器的基于资源的策略。

查看容器策略（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择容器名称。

此时将显示容器详细信息页面。策略将显示在 Container policy (容器策略) 部分中。

查看容器策略 (AWS CLI)

- 在中 AWS CLI，使用以下get-container-policy命令：

```
aws mediastore get-container-policy --container-name ExampleLiveDemo --region us-
west-2
```

以下示例显示了返回值：

```
{
  "Policy": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "PublicReadOverHttps",
        "Effect": "Allow",
        "Principal": {
          "AWS": "arn:aws:iam::111122223333:root",
        },
        "Action": [
          "mediastore:GetObject",
          "mediastore:DescribeObject",
        ],
        "Resource": "arn:aws:mediastore:us-
west-2:111122223333:container/ExampleLiveDemo/*",
        "Condition": {
          "Bool": {
            "aws:SecureTransport": "true"
          }
        }
      }
    ]
  }
}
```

编辑容器策略

您可以编辑默认容器策略中的权限，也可以创建替换默认策略的新策略。新策略最长需要五分钟就能生效。

编辑容器策略（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择容器名称。
3. 选择编辑策略。有关显示如何设置不同权限的示例，请参阅the section called “示例容器策略”。
4. 进行适当的更改，然后选择 Save (保存)。

编辑容器策略 (AWS CLI)

1. 创建一个定义容器策略的文件：

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadOverHttps",
      "Effect": "Allow",
      "Action": ["mediastore:GetObject", "mediastore:DescribeObject"],
      "Principal": "*",
      "Resource": "arn:aws:mediastore:us-
west-2:111122223333:container/ExampleLiveDemo/*",
      "Condition": {
        "Bool": {
            "aws:SecureTransport": "true"
        }
      }
    }
  ]
}
```

2. 在中 AWS CLI，使用以下put-container-policy命令：

```
aws mediastore put-container-policy --container-name ExampleLiveDemo --
policy file://ExampleContainerPolicy.json --region us-west-2
```

此命令没有返回值。

示例容器策略

以下示例显示了针对不同用户组构建的容器策略。

主题

示例容器策略：默认

当您创建容器时，AWS Elemental MediaStore 会自动附加以下基于资源的策略：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "MediaStoreFullAccess",
            "Action": [
                "mediastore:*"
            ],
            "Principal": {
                "AWS": "arn:aws:iam::333333333333:root"
```

```
            },
            "Effect": "Allow",
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        }
    ]
}
```

该策略已内置于该服务中，因此您无需创建它。但是，如果默认策略中的权限与您要用于容器的权限不一致，则可以编辑容器上的策略。

分配到所有新容器的默认策略允许访问容器上的所有 MediaStore 操作。它指定此访问具有对于操作需要 HTTPS 的条件。

示例容器策略：通过 HTTPS 的公共读取访问权限

此示例策略允许用户通过 HTTPS 请求检索对象。它允许任何人通过安全 SSL/TLS 连接进行读取访问：经过身份验证的用户和匿名用户（未登录的用户）。该语句具有名称 PublicReadOverHttps。它允许在任何对象（资源路径的末尾由 * 指定）上访问 GetObject 和 DescribeObject 操作。它指定此访问具有对于操作需要 HTTPS 的条件：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadOverHttps",
            "Effect": "Allow",
            "Action": [
                "mediastore:GetObject",
                "mediastore:DescribeObject"
            ],
            "Principal": "*",
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
```

```
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        }
    ]
}
```

示例容器策略：通过 HTTP 或 HTTPS 的公共读取访问权限

此示例策略允许在任何对象（在资源路径的末尾由 * 指定）上访问 GetObject 和 DescribeObject 操作。它对任何人都允许读取访问权限，包括所有经过身份验证的用户和匿名用户（未登录的用户）：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadOverHttpOrHttps",
            "Effect": "Allow",
            "Action": [
                "mediastore:GetObject",
                "mediastore:DescribeObject"
            ],
            "Principal": "*",
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "false"
                }
            }
        }
    ]
}
```

示例容器策略：已启用 HTTP 的跨账户读取访问权限

此实例策略允许用户通过 HTTP 请求检索对象。它对具有跨账户访问权限的经过身份验证的用户允许此访问权限。该对象不需要托管在带有 SSL/TLS 证书的服务器上：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccountReadOverHttpOrHttps",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::333333333333:root"
            },
            "Action": [
                "mediastore:GetObject",
                "mediastore:DescribeObject"
            ],
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        }
    ]
}
```

示例容器策略：通过 HTTPS 的跨账户读取访问权限

此示例策略允许在由指定 <其他账号> 的根用户所拥有的任何对象（资源路径的末尾由 * 指定）上访问 `GetObject` 和 `DescribeObject` 操作。它指定此访问具有对于操作需要 HTTPS 的条件：

JSON

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

```
    "Statement": [
        {
            "Sid": "CrossAccountReadOverHttps",
            "Effect": "Allow",
            "Action": [
                "mediastore:GetObject",
                "mediastore:DescribeObject"
            ],
            "Principal": {
                "AWS": "arn:aws:iam::333333333333:root"
            },
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
            "Condition": {
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        }
    ]
}
```

示例容器策略：对角色的跨账户读取访问权限

此示例策略允许在由 <所有者账号> 所拥有的任何对象（资源路径的末尾由 * 指定）上访问 GetObject 和 DescribeObject 操作。它对 <其他账号> 的任何用户允许此访问权限，前提是该账户已担任在 <角色名称> 中指定的角色：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CrossAccountRoleRead",
      "Effect": "Allow",
      "Action": ["mediastore:GetObject", "mediastore:DescribeObject"],
      "Principal":{
        "AWS": "arn:aws:iam::<other acct number>:role/<role name>"},
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>:container/<container
 name>/*",
    }
  ]
```

```
}
```

示例容器策略：对角色的跨账户完全访问权限

此示例策略允许跨账户访问权限以更新账户中的任何对象，只要用户通过 HTTP 登录即可。它还允许跨账户访问权限以通过 HTTP 或 HTTPS 删除、下载和描述对象，只要这个账户已担任指定的角色：

- 第一个语句是 CrossAccountRolePostOverHttps。它允许在任何对象上访问 PutObject 操作，并且对指定账户的任何用户允许此访问权限，前提是该账户已担任在 <角色名称> 中指定的角色。它指定此访问具有对于操作需要 HTTPS 的条件（此条件在提供对 PutObject 的访问权限时必须始终包含在内）。

  换言之，具有跨账户访问权限的任何委托人都可以访问 PutObject，但只能通过 HTTPS 进行访问。
- 第二个语句是 CrossAccountFullAccessExceptPost。它允许在任何对象上访问除 PutObject 之外的所有操作。它对指定账户的任何用户允许此访问权限，前提是该账户已担任在 <角色名称> 中指定的角色。此访问没有对于操作需要 HTTPS 的条件。

  换言之，具有跨账户访问权限的任何账户都可以访问 DeleteObject、GetObject 等（PutObject 除外），而且可通过 HTTP 或 HTTPS 执行此操作。

  如果您从第二个语句中未排除 PutObject，则该语句将不会有效（因为如果包含 PutObject，您必须将 HTTPS 显式设置为条件）。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccountRolePostOverHttps",
            "Effect": "Allow",
            "Action": "mediastore:PutObject",
            "Principal": {
                "AWS": "arn:aws:iam::333333333333:role/<role name>"
            },
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
            "Condition": {
```

```
                "Bool": {
                    "aws:SecureTransport": "true"
                }
            }
        },
        {
            "Sid": "CrossAccountFullAccessExceptPost",
            "Effect": "Allow",
            "NotAction": "mediastore:PutObject",
            "Principal": {
                "AWS": "arn:aws:iam::333333333333:role/<role name>"
            },
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*"
        }
    ]
}
```

示例容器策略：限制对特定 IP 地址的访问

此示例策略允许访问对指定容器中的对象执行所有 AWS Elemental MediaStore 操作。但是，请求必须来自条件中指定的 IP 地址范围。

本语句中的条件确定了允许的互联网协议版本 4 (IPv4) IP 地址的198.51.100.* 范围，但有一个例外：198.51.100.188。

Condition 块使用 IpAddress 和 NotIpAddress 条件以及 aws:SourceIp 条件键 (这是 AWS 范围的条件键)。这些aws:sourceIp IPv4 值使用标准的 CIDR 表示法。有关更多信息，请参阅 《IAM 用户指南》中的 IP 地址条件运算符。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AccessBySpecificIPAddress",
            "Effect": "Allow",
            "Action": [
                "mediastore:GetObject",
                "mediastore:DescribeObject"
```

```
            ],
            "Principal": "*",
            "Resource": "arn:aws:mediastore:us-
east-2:333333333333:container/<container name>/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "198.51.100.0/24"
                    ]
                },
                "NotIpAddress": {
                    "aws:SourceIp": "198.51.100.188/32"
                }
            }
        }
    ]
}
```

AWS Elemental 中的跨源资源共享 (CORS) 策略 MediaStore

跨源资源共享 (CORS) 定义了在一个域中加载的客户端 Web 应用程序与另一个域中的资源交互的方式。借助 AWS MediaStore Elemental 中的 CORS 支持，您可以构建丰富的客户端 Web 应用程序，并 MediaStore 有选择地允许跨源访问您的资源。 MediaStore

> **Note**
>
> 如果您使用 Amazon CloudFront 分发具有 CORS 策略的容器中的内容，请务必为 AWS MediaStore Elemental 配置分配（包括编辑缓存行为以设置 CORS 的步骤）。

本部分提供 CORS 概述。副主题描述了如何使用 AWS MediaStore Elemental 控制台启用 CORS，或者使用 REST API 和 AWS 以编程方式启用 MediaStore CORS。 SDKs

主题

- CORS 使用案例方案
- 将 CORS 策略添加到容器
- 查看 CORS 策略
- 编辑 CORS 策略

- 删除 CORS 策略
- 排查 CORS 问题
- 示例 CORS 策略

CORS 使用案例方案

以下是有关使用 CORS 的示例场景：

- 场景 1：假设您在名为的 AWS Elemental MediaStore 容器中分发直播视频。LiveVideo您的用户将从特定源（如 `http://livevideo.mediastore.ap-southeast-2.amazonaws.com`）加载视频清单终端节点 `www.example.com`。您想使用 JavaScript 视频播放器通过未经身份验证`GET`的`PUT`请求访问来自此容器的视频。浏览器通常会 JavaScript阻止允许这些请求，但您可以在容器上设置 CORS 策略以明确启用来自`www.example.com`的这些请求。
- 场景 2：假设您想在 MediaStore 容器中托管与场景 1 中相同的直播，但希望允许来自任何来源的请求。您可以配置 CORS 策略来允许通配符 (*) 源，以便来自任何源的请求可以访问视频。

将 CORS 策略添加到容器

本节介绍如何向 AWS Elemental 容器添加跨源资源共享 (CORS MediaStore) 配置。CORS 允许在一个域中加载的客户端 Web 应用程序与另一个域中的资源进行交互。

要将容器配置为允许跨源请求，请将 CORS 策略添加到容器。CORS 策略定义用于标识源（允许访问容器）、每个源支持的操作（HTTP 方法）和其他操作特定信息的规则。

将 CORS 策略添加到容器后，容器策略（控制对容器的访问权限）将继续适用。

添加 CORS 策略（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择要为其创建 CORS 策略的容器的名称。

   此时将显示容器详细信息页面。
3. 在 Container CORS policy (容器 CORS 策略) 部分中，选择 Create CORS policy (创建 CORS 策略)。
4. 插入 JSON 格式的策略，然后选择 Save (保存)。

添加 CORS 策略 (AWS CLI)

1. 创建一个定义 CORS 策略的文件：

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "HEAD"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "MaxAgeSeconds": 3000
  }
]
```

2. 在中 AWS CLI，使用`put-cors-policy`命令。

```
aws mediastore put-cors-policy --container-name ExampleContainer --cors-policy
 file://corsPolicy.json --region us-west-2
```

此命令没有返回值。

查看 CORS 策略

跨源资源共享 (CORS) 定义了在一个域中加载的客户端 Web 应用程序与另一个域中的资源交互的方式。

查看 CORS 策略（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择要查看其 CORS 策略的容器的名称。

   将出现容器详细信息页面，其中 CORS 策略位于 Container CORS policy (容器 CORS 策略) 部分中。

查看 CORS 策略 (AWS CLI)

- 在中 AWS CLI，使用以下get-cors-policy命令：

```
aws mediastore get-cors-policy --container-name ExampleContainer --region us-west-2
```

以下示例显示了返回值：

```
{
    "CorsPolicy": [
        {
            "AllowedMethods": [
                "GET",
                "HEAD"
            ],
            "MaxAgeSeconds": 3000,
            "AllowedOrigins": [
                "*"
            ],
            "AllowedHeaders": [
                "*"
            ]
        }
    ]
}
```

编辑 CORS 策略

跨源资源共享 (CORS) 定义了在一个域中加载的客户端 Web 应用程序与另一个域中的资源交互的方式。

编辑 CORS 策略（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择要编辑其 CORS 策略的容器的名称。

   此时将显示容器详细信息页面。
3. 在 Container CORS policy (容器 CORS 策略) 部分中，选择 Edit CORS policy (编辑 CORS 策略)。

4. 更改策略，然后选择 Save (保存)。

编辑 CORS 策略 (AWS CLI)

1. 创建一个定义更新 CORS 策略的文件：

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "HEAD"
    ],
    "AllowedOrigins": [
      "https://www.example.com"
    ],
    "MaxAgeSeconds": 3000
  }
]
```

2. 在中 AWS CLI，使用`put-cors-policy`命令。

```
aws mediastore put-cors-policy --container-name ExampleContainer --cors-policy
 file://corsPolicy2.json --region us-west-2
```

此命令没有返回值。

删除 CORS 策略

跨源资源共享 (CORS) 定义了在一个域中加载的客户端 Web 应用程序与另一个域中的资源交互的方式。删除容器中的 CORS 策略将删除跨源请求的权限。

删除 CORS 策略（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择要删除其 CORS 策略的容器的名称。

此时将显示容器详细信息页面。

3. 在 Container CORS policy (容器 CORS 策略) 部分中，选择 Delete CORS policy (删除 CORS 策略)。
4. 选择 Continue (继续) 以确认，然后选择 Save (保存)。

删除 CORS 策略 (AWS CLI)

- 在中 AWS CLI，使用以下`delete-cors-policy`命令：

```
aws mediastore delete-cors-policy --container-name ExampleContainer --region us-west-2
```

此命令没有返回值。

排查 CORS 问题

如果在访问具有 CORS 策略的容器时遇到意外行为，请执行以下步骤以排查问题。

1. 验证 CORS 策略是否附加到容器。

   有关说明，请参阅 the section called “查看 CORS 策略”。
2. 使用选择的工具（如浏览器的开发人员控制台）捕获完整的请求和响应。验证附加到容器的 CORS 策略是否至少包含一个与请求数据匹配的 CORS 规则，如下所示：

   a. 验证请求是否具有 `Origin` 标头。

      如果缺少标头，AWS Elemental MediaStore 不会将请求视为跨源请求，也不会在响应中发回 CORS 响应标头。

   b. 验证请求中的 `Origin` 标头是否与特定 `AllowedOrigins` 中的至少一个 `CORSRule` 元素匹配。

      `Origin` 请求标头中的方案、主机和端口值必须与 `AllowedOrigins` 中的 `CORSRule` 匹配。例如，如果设置 `CORSRule` 以允许源 `http://www.example.com`，则请求中的 `https://www.example.com` 和 `http://www.example.com:80` 源与配置中允许的源都不匹配。

   c. 验证请求中的方法（或对于预检请求，为 `Access-Control-Request-Method` 中指定的方法）是否为相同 `AllowedMethods` 中的 `CORSRule` 元素之一。

d. 对于预检请求，如果请求包含 Access-Control-Request-Headers 标头，请验证对于 CORSRule 标头中的每个值，AllowedHeaders 是否包含 Access-Control-Request-Headers 条目。

示例 CORS 策略

以下示例显示了跨源资源共享 (CORS) 策略。

主题

示例 CORS 策略：任何域的读取访问权限

以下策略允许来自任何域的网页从您的 AWS Elemental MediaStore 容器中检索内容。请求包括来自原始域的所有 HTTP 标头，服务仅响应来自原始域的 HTTP GET 和 HTTP HEAD 请求。在交付新的结果集之前，将缓存 3000 秒的结果。

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "HEAD"
    ],
    "AllowedOrigins": [
      "*"
    ],
    "MaxAgeSeconds": 3000
  }
]
```

示例 CORS 策略：特定域的读取访问权限

以下策略允许网页从https://www.example.com您的 AWS Elemental MediaStore 容器中检索内容。请求包括来自 https://www.example.com 的所有 HTTP 标头，服务仅响应来自 https://

www.example.com 的 HTTP GET 和 HTTP HEAD 请求。在交付新的结果集之前，将缓存 3000 秒的结果。

```
[
  {
    "AllowedHeaders": [
      "*"
    ],
    "AllowedMethods": [
      "GET",
      "HEAD"
    ],
    "AllowedOrigins": [
      "https://www.example.com"
    ],
    "MaxAgeSeconds": 3000
  }
]
```

AWS Elemental 中的对象生命周期策略 MediaStore

对于每个容器，您可以创建一个对象生命周期策略，用于管理对象在容器中应该存储多长时间。当对象达到您指定的最大使用期限时，AWS Elemental MediaStore 会删除这些对象。您可以通过删除不再需要的对象来节省存储成本。

您还可以指定在对象达到一定年限后 MediaStore 应将其移至不频繁访问 (IA) 存储类别。存储在 IA 存储类中的对象与存储在标准存储类中的对象具有不同的存储和检索速率。有关更多信息，请参阅MediaStore 定价。

一个对象生命周期策略包含规则，规则按子文件夹指定对象的生命周期。（您不能将对象生命周期策略分配给单个对象）。对于一个容器，您只能附加一个对象生命周期策略，但您可以给每个对象生命周期策略添加最多 10 个规则。有关更多信息，请参阅 对象生命周期策略的组成。

主题

- 示例对象生命周期策略

对象生命周期策略的组成

对象生命周期策略控制对象在 AWS Elemental MediaStore 容器中保留多长时间。每个对象生命周期策略由一条或多条规则组成，规则指示对象的生命周期。一个规则可以应用到一个文件夹、多个文件夹或整个容器。

对于一个容器，您可以附加一个对象生命周期策略，而且每个对象生命周期策略可以包含最多 10 个规则。您不能将对象生命周期策略分配给单个对象。

对象生命周期策略中的规则

您可以创建三种类型的规则：

- 瞬态数据
- 删除对象
- 生命周期转换

瞬态数据

瞬态数据规则将对象设置为在几秒钟内过期。此类规则仅适用于在策略生效后添加到容器中的对象。将新策略应用 MediaStore 于容器最多需要 20 分钟。

瞬态数据的规则示例如下所示：

```
        {
            "definition": {
                "path": [ {"wildcard": "Football/index*.m3u8"} ],
                "seconds_since_create": [
                    {"numeric": [">", 120]}
                ]
            },
            "action": "EXPIRE"
        },
```

瞬态数据规则有三个部分：

- `path`：始终设置为 `wildcard`。您可以使用此部分来定义您要删除哪些对象。您可以使用一个或多个通配符，以星号 (*) 表示。每个通配符表示 0 个或多个字符的任意组合。例

如，"path": [{"wildcard": "Football/index*.m3u8"}]，适用于 Football 文件夹中与 index*.m3u8 的模式匹配的所有文件（例如 index.m3u8、index1.m3us8 和 index123456.m3u8）。单个规则中最多可以包含 10 条 路径。

- seconds_since_create：始终设置为 numeric。可以指定 1-300 秒的值。也可以将运算符设置为大于 (>) 或大于等于 (>=)。
- action：始终设置为 EXPIRE。

对于瞬态数据规则（对象在几秒钟内过期），在对象过期和删除对象之间没有延迟。

> **Note**
>
> 受瞬态数据规则约束的对象不包括在 list-items 响应中。此外，由于临时数据规则而过期的对象在过期时不会发出 CloudWatch 事件。

删除对象

删除对象规则将对象设置为在几天之内过期。此类规则适用于容器中的所有对象，即使它们在创建策略之前已添加到容器中也是如此。应用新策略最多需要 20 分钟，但从容器中清除对象最多可能需要 24 小时。 MediaStore

删除对象的两条规则示例如下所示：

```
    {
        "definition": {
            "path": [ { "prefix": "FolderName/" }  ],
            "days_since_create": [
                {"numeric": [">" , 5]}
            ]
        },
        "action": "EXPIRE"
    },
    {
        "definition": {
            "path": [ { "wildcard": "Football/*.ts" }  ],
            "days_since_create": [
                {"numeric": [">" , 5]}
            ]
        },
        "action": "EXPIRE"
```

```
        }
```

删除对象规则有三个部分：

- path：设置为 prefix 或 wildcard。您不能在同一规则中混用 prefix 和 wildcard。如果要同时使用两者，则必须为 prefix 创建一条规则，为 wildcard 单独创建一条规则，如上面的示例所示。
  - prefix – 如果要删除特定文件夹中的所有对象，则将路径设置为 prefix。如果该参数为空 ("path": [{ "prefix": "" }],)，则目标是当前容器内的任何位置存储的所有对象。单个规则中最多可以包含 10 条 prefix 路径。
  - wildcard – 如果要基于文件名和/或文件类型删除特定对象，则将路径设置为 wildcard。您可以使用一个或多个通配符，以星号 (*) 表示。每个通配符表示 0 个或多个字符的任意组合。例如，"path": [{"wildcard": "Football/*.ts"}], 适用于 Football 文件夹中与 *.ts 的模式匹配的所有文件（例如 filename.ts、filename1.ts 和 filename123456.ts）。单个规则中最多可以包含 10 条 wildcard 路径。
- days_since_create：始终设置为 numeric。可以指定 1-36,500 天的值。也可以将运算符设置为大于 (>) 或大于等于 (>=)。
- action：始终设置为 EXPIRE。

对于删除对象规则（对象在几天之内过期），在对象过期和删除对象之间可能会略有滞后。但是，一旦对象过期，账单就会立即发生变化。例如，如果生命周期规则指定 10 days_since_create，则在对象超过 10 之后，即使该对象尚未删除，该账户也无需为其支付费用。

生命周期转换

生命周期转换规则会将对象设置为在达到一定期限（以天计算）后移动到不经常访问 (IA) 存储类中。存储在 IA 存储类中的对象与存储在标准存储类中的对象具有不同的存储和检索速率。有关更多信息，请参阅MediaStore 定价。

对象一旦移动到 IA 存储类中，就无法再移回标准存储类。

生命周期转换规则适用于容器中的所有对象，即使它们在创建策略之前已添加到容器中也是如此。应用新策略最多需要 20 分钟，但从容器中清除对象最多可能需要 24 小时。 MediaStore

下面为一个生命周期转换规则示例：

```
        {
            "definition": {
                "path": [
```

```
                {"prefix": "AwardsShow/"}
            ],
            "days_since_create": [
                {"numeric": [">=" , 30]}
            ]
        },
        "action": "ARCHIVE"
    }
```

生命周期转换规则包含三个部分：

- path：设置为 prefix 或 wildcard。您不能在同一规则中混用 prefix 和 wildcard。如果要同时使用两者，则必须为 prefix 创建一个规则，再为 wildcard 创建另一个规则。
  - prefix - 如果要将特定文件夹中的所有对象移动到 IA 存储类，则可以将 path 设置为 prefix。如果该参数为空 ("path": [{ "prefix": "" }],)，则目标是当前容器内的任何位置保存的所有对象。单个规则中最多可以包含 10 条 prefix 路径。
  - wildcard – 如果要根据文件名和/或文件类型将特定对象移动到 IA 存储类，则可以将 path 设置为 wildcard。您可以使用一个或多个通配符，以星号 (*) 表示。每个通配符表示 0 个或多个字符的任意组合。例如，"path": [{"wildcard": "Football/*.ts"}], 适用于 Football 文件夹中与 *.ts 的模式匹配的所有文件（例如 filename.ts、filename1.ts 和 filename123456.ts）。单个规则中最多可以包含 10 条 wildcard 路径。
- days_since_create：始终设置为 "numeric": [">=" , 30]。
- action：始终设置为 ARCHIVE。

示例

假设一个名为 LiveEvents 的容器有四个子文件夹：Football、Baseball、Basketball 和 AwardsShow。分配给 LiveEvents 文件夹的对象生命周期策略可能类似于如下内容：

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"prefix": "Football/"},
                    {"prefix": "Baseball/"}
                ],
                "days_since_create": [
                    {"numeric": [">" , 28]}
```

```
            ]
        },
        "action": "EXPIRE"
    },
    {
        "definition": {
            "path": [ { "prefix": "AwardsShow/" }  ],
            "days_since_create": [
                {"numeric": [">=" , 15]}
            ]
        },
        "action": "EXPIRE"
    },
    {
        "definition": {
            "path": [ { "prefix": "" }  ],
            "days_since_create": [
                {"numeric": [">" , 40]}
            ]
        },
        "action": "EXPIRE"
    },
    {
        "definition": {
            "path": [ { "wildcard": "Football/*.ts" }  ],
            "days_since_create": [
                {"numeric": [">" , 20]}
            ]
        },
        "action": "EXPIRE"
    },
    {
        "definition": {
            "path": [
                {"wildcard": "Football/index*.m3u8"}
            ],
            "seconds_since_create": [
                {"numeric": [">" , 15]}
            ]
        },
        "action": "EXPIRE"
    },
    {
            "definition": {
```

```
                    "path": [
                        {"prefix": "Program/"}
                    ],
                    "days_since_create": [
                        {"numeric": [">=" , 30]}
                    ]
                },
                "action": "ARCHIVE"
            }
    ]
}
```

上述策略指定以下内容：

- 第一条规则指示 AWS MediaStore Elemental 删除存储`LiveEvents/Football`在文件夹和文件夹`LiveEvents/Baseball`中的存储时间超过 28 天的对象。
- 第二个规则指示服务删除存储在 `LiveEvents/AwardsShow` 文件夹中达到或超过 15 天的对象。
- 第三个规则指示服务删除存储在 `LiveEvents` 容器中任何地方超过 40 天的对象。这条规则应用于直接存储在 `LiveEvents` 容器中和存储在该容器四个子文件夹任何一个中的对象。
- 第四个规则指示服务删除 `Football` 文件夹中与模式 `*.ts` 匹配的超过 20 天的对象。
- 第五条规则指示服务在`Football`文件夹中与模式匹配的对象超过15秒`index*.m3u8`后将其删除。MediaStore 在这些文件放入容器后 16 秒钟将其删除。
- 第六条规则指示服务在 `Program` 文件夹中的对象达到 30 天后将其移动到 IA 存储类。

有关对象生命周期策略的更多示例，请参阅示例对象生命周期策略。

为容器添加对象生命周期策略

对象生命周期策略让您可以指定对象在容器中存储的时长。您设置了过期日期，在过期日期之后，AWS Elemental MediaStore 会删除这些对象。服务最长需要 20 分钟才能对容器应用新策略。

有关如何构建生命周期策略的信息，请参阅对象生命周期策略的组成。

> **Note**
>
> 对于删除对象规则（对象在几天之内过期），在对象过期和删除对象之间可能会略有滞后。但是，一旦对象过期，账单就会立即发生变化。例如，如果生命周期规则指定 10

`days_since_create`，则在对象超过 10 之后，即使该对象尚未删除，该账户也无需为其支付费用。

添加对象生命周期策略（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择要为其创建对象生命周期策略的容器的名称。

   此时将显示容器详细信息页面。
3. 在 Object lifecycle policy (对象生命周期策略) 部分中，选择 Create object lifecycle policy (创建对象生命周期策略)。
4. 插入 JSON 格式的策略，然后选择 Save (保存)。

添加对象生命周期策略 (AWS CLI)

1. 创建一个文件，该文件定义对象生命周期策略：

```
{
    "rules": [
         {
            "definition": {
                "path": [
                    {"prefix": "Football/"},
                    {"prefix": "Baseball/"}
                ],
                "days_since_create": [
                    {"numeric": [">" , 28]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [
                    {"wildcard": "AwardsShow/index*.m3u8"}
                ],
                "seconds_since_create": [
                    {"numeric": [">" , 8]}
                ]
```

```
            },
            "action": "EXPIRE"
        }
    ]
}
```

2. 在中 AWS CLI，使用以下`put-lifecycle-policy`命令：

```
aws mediastore put-lifecycle-policy --container-name LiveEvents --lifecycle-
policy file://LiveEventsLifecyclePolicy.json --region us-west-2
```

此命令没有返回值。服务器将指定的策略附加到容器。

查看对象生命周期策略

对象生命周期策略指定对象应在容器中保留的时间。

查看对象生命周期策略（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择要查看其对象生命周期策略的容器的名称。

   随即出现容器详细信息页面，在 Object lifecycle policy (对象生命周期策略) 部分中显示了对象生命周期策略。

查看对象生命周期策略 (AWS CLI)

- 在中 AWS CLI，使用以下`get-lifecycle-policy`命令：

```
aws mediastore get-lifecycle-policy --container-name LiveEvents --region us-west-2
```

以下示例显示了返回值：

```
{
    "LifecyclePolicy": "{
        "rules": [
            {
                "definition": {
                    "path": [
                        {"prefix": "Football/"},
```

```
                        {"prefix": "Baseball/"}
                    ],
                    "days_since_create": [
                        {"numeric": [">" , 28]}
                    ]
                },
             "action": "EXPIRE"
            }
        ]
    }"
}
```

编辑对象生命周期策略

您无法编辑现有的对象生命周期策略。但是，您可以通过上传一个替换策略来更改现有策略。服务最长需要 20 分钟来对容器应用更新的策略。

编辑对象生命周期策略（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择要编辑其对象生命周期策略的容器的名称。

   此时将显示容器详细信息页面。
3. 在 Object lifecycle policy (对象生命周期策略) 部分中，选择 Edit object lifecycle policy (编辑对象生命周期策略)。
4. 更改策略，然后选择 Save (保存)。

编辑对象生命周期策略 (AWS CLI)

1. 创建一个文件，该文件定义更新的对象生命周期策略：

```
{
    "rules": [
         {
            "definition": {
                "path": [
                    {"prefix": "Football/"},
                    {"prefix": "Baseball/"}
                    {"prefix": "Basketball/"}
                ],
```

```
                "days_since_create": [
                    {"numeric": [">" , 28]}
                ]
            },
            "action": "EXPIRE"
        }
    ]
}
```

2. 在中 AWS CLI，使用以下`put-lifecycle-policy`命令：

```
aws mediastore put-lifecycle-policy --container-name LiveEvents --lifecycle-
policy file://LiveEvents2LifecyclePolicy --region us-west-2
```

此命令没有返回值。服务器将指定的策略附加到容器，同时替换之前的策略。

删除对象生命周期策略

当您删除对象生命周期策略时，服务最长需要 20 分钟来将更改应用到容器。

删除对象生命周期策略（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择要删除其对象生命周期策略的容器的名称。

   此时将显示容器详细信息页面。
3. 在 Object lifecycle policy (对象生命周期策略) 部分中，选择 Delete lifecycle policy (删除生命周期策略)。
4. 选择 Continue (继续) 以确认，然后选择 Save (保存)。

删除对象生命周期策略 (AWS CLI)

- 在中 AWS CLI，使用以下`delete-lifecycle-policy`命令：

```
aws mediastore delete-lifecycle-policy --container-name LiveEvents --region us-
west-2
```

此命令没有返回值。

示例对象生命周期策略

以下是示例对象生命周期策略。

主题

示例对象生命周期策略：在几秒内过期

以下策略指定 MediaStore 删除符合以下所有条件的对象：

- 在策略生效之后添加到容器中。
- 存储在 `Football` 文件夹中。
- 具有 `m3u8` 文件扩展名。
- 已在容器中存放超过 20 秒。

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"wildcard": "Football/*.m3u8"}
                ],
                "seconds_since_create": [
                    {"numeric": [ ">", 20 ]}
                ]
            },
            "action": "EXPIRE"
        }
    ]
}
```

对象生命周期策略示例：在几天内过期

以下策略指定 MediaStore 删除符合以下所有条件的对象：

- 存储在 `Program` 文件夹中
- 具有 `ts` 文件扩展名
- 已在容器中存放超过 5 天

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"wildcard": "Program/*.ts"}
                ],
                "days_since_create": [
                    {"numeric": [ ">", 5 ]}
                ]
            },
            "action": "EXPIRE"
        }
    ]
}
```

对象生命周期策略示例：移动到不经常访问存储类

以下策略规定，当对象已有 30 天时，将其 MediaStore 移至不频繁访问 (IA) 存储类别。存储在 IA 存储类中的对象与存储在标准存储类中的对象具有不同的存储和检索速率。

`days_since_create` 字段必须设置为 `"numeric": [">=" ,30]`。

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"prefix": "Football/"},
                    {"prefix": "Baseball/"}
                ],
                "days_since_create": [
```

```
                    {"numeric": [">=" , 30]}
                ]
            },
            "action": "ARCHIVE"
        }
    ]
}
```

示例对象生命周期策略：多个规则

以下策略规定了执行以下 MediaStore 操作：

- 将在 AwardsShow 文件夹中存储达到 30 天的对象移动到不经常访问 (IA) 存储类中。
- 删除文件扩展名为 m3u8 且在 Football 文件夹中存储达到 20 秒的对象
- 删除在 April 文件夹中存储达到 10 天的对象
- 删除文件扩展名为 ts 且在 Program 文件夹中存储达到 5 天的对象

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"prefix": "AwardsShow/"}
                ],
                "days_since_create": [
                    {"numeric": [ ">=" , 30 ]}
                ]
            },
            "action": "ARCHIVE"
        },
        {
            "definition": {
                "path": [
                    {"wildcard": "Football/*.m3u8"}
                ],
                "seconds_since_create": [
                    {"numeric": [ ">", 20 ]}
                ]
            },
            "action": "EXPIRE"
        },
```

```
        {
            "definition": {
                "path": [
                    {"prefix": "April"}
                ],
                "days_since_create": [
                    {"numeric": [ ">", 10 ]}
                ]
            },
            "action": "EXPIRE"
        },
        {
            "definition": {
                "path": [
                    {"wildcard": "Program/*.ts"}
                ],
                "days_since_create": [
                    {"numeric": [ ">", 5 ]}
                ]
            },
            "action": "EXPIRE"
        }
    ]
}
```

示例对象生命周期策略：空容器

以下对象生命周期策略规定，容器中的所有对象（包括文件夹和子文件夹）将在添加到容器 1 天后将其 MediaStore 删除。如果容器在应用此策略之前存放了任何对象，则会在策略生效 1 天后 MediaStore 删除这些对象。服务最长需要 20 分钟才能对容器应用新策略。

```
{
    "rules": [
        {
            "definition": {
                "path": [
                    {"wildcard": "*"}
                ],
                "days_since_create": [
                    {"numeric": [ ">=", 1 ]}
                ]
            },
            "action": "EXPIRE"
```

```
        }
    ]
}
```

AWS Elemental 中的指标策略 MediaStore

对于每个容器，您可以添加指标策略以允许 AWS Elemental MediaStore 向亚马逊发送指标。CloudWatch新策略最长需要 20 分钟就能生效。有关每个 MediaStore 指标的描述，请参阅MediaStore指标。

指标策略包含：

- 在容器级别启用或禁用指标的设置。
- 在对象级别启用指标的规则（从零个到五个的任一数量）。如果策略包含规则，每个规则必须包含以下两项：
  - 定义要包含在组中的对象的对象组。定义可以是路径或文件名，不能超过 900 个字符。有效字符包括：a-z、A-Z、0-9、_（下划线）、=（等号）、:（冒号）、.（句点）、-（连字符）、~（波浪符）、/（正斜线）、*（星号）。接受通配符（*）。
  - 让您可以引用对象组的对象组名称。名称不能超过 30 个字符。有效字符包括：a-z、A-Z、0-9、_（下划线）。

如果一个对象匹配多个规则，则 CloudWatch 显示每个匹配规则的数据点。例如，如果一个对象与两个名为`rule1`和的规则匹配`rule2`，则 CloudWatch 会显示这些规则的两个数据点。第一个维度为`ObjectGroupName=rule1`，第二个维度为 `ObjectGroupName=rule2`。

主题

- 添加指标策略
- 查看指标策略
- 编辑指标策略
- 示例指标策略

添加指标策略

指标策略包含规定 AWS MediaStore Elemental 向亚马逊发送哪些指标的规则。 CloudWatch有关指标策略的示例，请参阅 示例指标策略。

添加指标策略（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择要向其添加指标策略的容器的名称。

   此时将显示容器详细信息页面。
3. 在 Metric policy (指标策略) 部分，选择 Create metric policy (创建指标策略)。
4. 插入 JSON 格式的策略，然后选择 Save (保存)。

查看指标策略

您可以使用控制台或 AWS CLI 查看容器的指标策略。

查看指标策略（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择容器名称。

   此时将显示容器详细信息页面。策略将显示在 Metric policy (指标策略) 部分。

编辑指标策略

指标策略包含规定 AWS MediaStore Elemental 向亚马逊发送哪些指标的规则。 CloudWatch编辑现有指标策略时，新策略最长需要 20 分钟生效。有关指标策略的示例，请参阅 示例指标策略。

编辑指标策略（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择容器名称。
3. 在 Metric policy (指标策略) 部分，选择 Edit metric policy (编辑指标策略)。
4. 进行适当的更改，然后选择 Save (保存)。

示例指标策略

以下示例显示了针对不同使用案例构建的指标策略。

主题

示例指标策略：容器级指标

此示例策略表明 AWS Elemental MediaStore 应 CloudWatch 在容器级别向亚马逊发送指标。例如，包括统计向容器发出的 Put 请求数的 RequestCount 指标。或者，您可以将其设置为 DISABLED。

由于此策略中没有规则，因此 MediaStore 不发送路径级别的指标。例如，您无法看到对此容器中的特定文件夹发出了多少 Put 请求。

```
{
   "ContainerLevelMetrics": "ENABLED"
}
```

示例指标策略：路径级指标

此示例策略表明，AWS Elemental MediaStore 不应 CloudWatch 在容器级别向亚马逊发送指标。而且， MediaStore 应在两个特定文件夹中发送对象的指标：baseball/saturday 和 football/saturday。 MediaStore 请求的指标如下：

- 对该baseball/saturday文件夹的请求的 CloudWatch 维度为ObjectGroupName=baseballGroup。
- 对 football/saturday 文件夹的请求具有维度 ObjectGroupName=footballGroup。

```
{
   "ContainerLevelMetrics": "DISABLED",
   "MetricPolicyRules": [
     {
       "ObjectGroup": "baseball/saturday",
       "ObjectGroupName": "baseballGroup"
     },
     {
       "ObjectGroup": "football/saturday",
       "ObjectGroupName": "footballGroup"
```

```
    }
  ]
}
```

示例指标策略：容器级和路径级指标

此示例策略表明 AWS Elemental MediaStore 应 CloudWatch 在容器级别向亚马逊发送指标。此外，还 MediaStore 应发送两个特定文件夹中对象的指标：`baseball/saturday`和`football/saturday`。MediaStore 请求的指标如下：

- 对该`baseball/saturday`文件夹的请求的 CloudWatch 维度为`ObjectGroupName=baseballGroup`。
- 对该`football/saturday`文件夹的请求有一个 CloudWatch 维度`ObjectGroupName=footballGroup`。

```
{
   "ContainerLevelMetrics": "ENABLED",
   "MetricPolicyRules": [
     {
       "ObjectGroup": "baseball/saturday",
       "ObjectGroupName": "baseballGroup"
     },
     {
       "ObjectGroup": "football/saturday",
       "ObjectGroupName": "footballGroup"
     }
   ]
}
```

示例指标策略：使用通配符的路径级指标

此示例策略表明 AWS Elemental MediaStore 应 CloudWatch 在容器级别向亚马逊发送指标。此外，还 MediaStore 应根据对象的文件名发送对象的指标。通配符表示对象可以存储在容器中的任何位置，而且可以使用任何文件名，但必须以 `.m3u8` 扩展名结尾。

```
{
   "ContainerLevelMetrics": "ENABLED",
   "MetricPolicyRules": [
     {
       "ObjectGroup": "*.m3u8",
```

```
      "ObjectGroupName": "index"
    }
  ]
}
```

示例指标策略：具有重叠规则的路径级指标

此示例策略表明 AWS Elemental MediaStore 应 CloudWatch 在容器级别向亚马逊发送指标。此外，还 MediaStore 应发送两个文件夹的指标：`sports/football/saturday`和`sports/football`。

向该`sports/football/saturday`文件夹发出的 MediaStore 请求的指标 CloudWatch 维度为`ObjectGroupName=footballGroup1`。由于存储在 `sports/football` 文件夹中的对象与两个规则都匹配，因此 CloudWatch 将为这些对象显示两个数据点：一个维度为 `ObjectGroupName=footballGroup1`，另一个维度为 `ObjectGroupName=footballGroup2`。

```
{
   "ContainerLevelMetrics": "ENABLED",
   "MetricPolicyRules": [
     {
       "ObjectGroup": "sports/football/saturday",
       "ObjectGroupName": "footballGroup1"
     },
     {
       "ObjectGroup": "sports/football",
       "ObjectGroupName": "footballGroup2"
     }
   ]
}
```

AWS Elemental 中的文件夹 MediaStore

文件夹在容器内细分。使用文件夹以您在文件系统中创建子文件夹来划分文件夹的相同方式细分您的容器。您可以创建最多 10 个级别的文件夹（不包括容器本身）。

文件夹是可选的；您可以选择将对象直接上传到容器而不是文件夹。但是，文件夹是整理对象的一种简单方式。

要将对象上传到文件夹，可以指定至文件夹的路径。如果该文件夹已经存在，AWS Elemental 会将该对象 MediaStore 存储在文件夹中。如果文件夹不存在，则该服务将创建文件夹，然后将对象存储在其中。

例如，假设您有一个名为 `movies` 的容器，而且上传一个名为 `mlaw.ts` 且路径为 `premium/canada` 的文件。AWS Elemental 将对象 MediaStore 存储在加拿大子文件夹 premium 下。如果这两个文件夹均不存在，则该服务会创建 `premium` 文件夹和 `canada` 子文件夹，然后将对象存储在 `canada` 子文件夹中。如果您仅指定容器 `movies`（无路径），则该服务会将对象直接存储在容器中。

当您删除该文件夹中的最后一个对象时，AWS Elemental 会 MediaStore 自动删除该文件夹。该服务还将删除该文件夹上的任何空文件夹。例如，假设您有一个名为 premium 的文件夹，它不包含任何文件，但包含一个名为 `canada` 的子文件夹。`canada` 子文件夹包含一个名为 `mlaw.ts` 的文件。如果删除文件 `mlaw.ts`，则该服务将同时删除 `premium` 和 `canada` 文件夹。此自动删除操作仅适用于文件夹。该服务不会删除空容器。

主题

文件夹命名规则

当您选择文件夹的名称时，请记住以下要求：

- 名称只能包含以下字符：大写字母（A-Z）、小写字母（a-z）、数字（0-9）、句点（.）、连字符(-)、短划线（-）、下划线（_）、等号（=）、以及冒号（:）。
- 名称必须至少有一个字符。不允许使用空文件夹名称（例如`folder1//folder3/`）。

- 名称区分大小写。例如，您可以在相同的容器或文件夹中具有名为 myFolder 的文件夹和名为 myfolder 的文件夹，因为这些名称是唯一的。
- 名称仅在其父容器或文件夹中必须唯一。例如，您可以在以下两个不同的容器中创建一个名为 myfolder 的文件夹：movies/myfolder 和 sports/myfolder。
- 名称可与其父容器名称相同。
- 在创建文件夹后，不能对其重命名。

创建文件夹

您可以在上传对象时创建文件夹。要将对象上传到文件夹，可以指定至文件夹的路径。如果该文件夹已经存在，AWS Elemental 会将该对象 MediaStore 存储在文件夹中。如果文件夹不存在，则该服务将创建文件夹，然后将对象存储在其中。

有关更多信息，请参阅 the section called “上传对象”。

删除文件夹

您只能在文件夹为空时将其删除；无法删除包含对象的文件夹。

当您删除该文件夹中的最后一个对象时，AWS Elemental 会 MediaStore 自动删除该文件夹。该服务还将删除该文件夹上的任何空文件夹。例如，假设有一个名为 premium 的文件夹，它不包含任何文件，但包含一个名为 canada 的子文件夹。canada 子文件夹包含一个名为 mlaw.ts 的文件。如果删除文件 mlaw.ts，则该服务将同时删除 premium 和 canada 文件夹。此自动删除操作仅适用于文件夹。该服务不会删除空容器。

有关更多信息，请参阅 删除对象。

AWS Elemental 中的对象 MediaStore

AWS Elemental MediaStore 资产被称为对象。可以将对象上传到容器或容器内的文件夹。

在中 MediaStore，您可以上传、下载和删除对象：

- 上传 – 将对象添加到容器或文件夹。这不同于创建对象。必须先在本地创建对象，然后才能将其上传到 MediaStore。
- 下载-将对象从中复制 MediaStore 到其他位置。这不会从中移除对象 MediaStore。
- 删除 – 从 MediaStore 中彻底删除对象。您可以逐个删除对象，也可以添加对象生命周期策略以在指定的持续时间后自动删除容器内的对象。

MediaStore 接受所有文件类型。

主题

上传对象

您可以将对象上传到容器或容器内的文件夹。要将对象上传到文件夹，可以指定至文件夹的路径。如果该文件夹已经存在，AWS Elemental 会将该对象 MediaStore 存储在文件夹中。如果文件夹不存在，则该服务将创建文件夹，然后将对象存储在其中。有关文件夹的更多信息，请参阅AWS Elemental 中的文件夹 MediaStore。

您可以使用 MediaStore 控制台或 AWS CLI 上传对象。

MediaStore 支持对对象进行分块传输，通过在对象仍在上传时可供下载，从而减少延迟。要使用此功能，请将对象的上传可用性设置为 `streaming`。您可以在使用 API 上传对象时设置此标头的值。如果您未在请求中指定此标头，则会`standard`为数据元的上传可用性 MediaStore 分配默认值。

标准上传可用性的对象大小不得超过 25MB，流上传可用性的对象大小不得超过 10MB。

> **Note**
>
> 对象文件名只能包含字母、数字、句点 (.)、下划线 (_)、波形符 (~)、连字符 (-)、等号 (=) 和冒号 (:)。

上传对象（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择容器的名称。将出现容器的详细信息面板。
3. 选择 Upload object (上传对象)。
4. 对于 Target path (目标路径)，键入文件夹的路径。例如 `premium/canada`。如果所指定路径中的任何文件夹不存在，则该服务将自动创建这些文件夹。
5. 在 Object (对象) 部分中，选择 Browse (浏览)。
6. 导航到相应文件夹，然后选择要上传的对象。
7. 选择 Open (打开)，然后选择 Upload (上传)。

> **Note**
>
> 如果选定文件夹中已存在同名文件，则该服务将用上传的文件替换原始文件。

上传对象 (AWS CLI)

- 在中 AWS CLI，使用`put-object`命令。您也可以包括以下任意参数：`content-type`、`cache-control`（以允许调用方控制对象缓存行为）和 `path`（用于将对象放入容器中的某个文件夹）。

> **Note**
>
> 上传对象后，您将无法编辑 `content-type`、`cache-control` 或 `path`。

```
aws mediastore-data put-object --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --body README.md --path /
```

```
folder_name/README.md --cache-control "max-age=6, public" --content-type binary/
octet-stream --region us-west-2
```

以下示例显示了返回值：

```
{
    "ContentSHA256":
 "74b5fdb517f423ed750ef214c44adfe2be36e37d861eafe9c842cbe1bf387a9d",
    "StorageClass": "TEMPORAL",
    "ETag": "af3e4731af032167a106015d1f2fe934e68b32ed1aa297a9e325f5c64979277b"
}
```

查看对象的列表

您可以使用 AWS Elemental MediaStore 控制台查看存储在容器最顶层或文件夹中的项目（对象和文件夹）。当前容器或文件夹的子文件夹中存储的项目将不会显示出来。无论容器 AWS CLI 内有多少文件夹或子文件夹，您都可以使用来查看容器中的对象和文件夹列表。

查看特定容器中对象的列表（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择容器（包含要查看的文件夹）的名称。
3. 从列表中选择文件夹的名称。

   将显示详细信息页面，其中显示了文件夹中存储的所有文件夹和对象。

查看特定文件夹中对象的列表（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择容器（包含要查看的文件夹）的名称。

   将显示详细信息页面，其中显示了容器中存储的所有文件夹和对象。

查看特定容器中对象和文件夹的列表 (AWS CLI)

- 在中 AWS CLI，使用以下`list-items`命令：

```
aws mediastore-data list-items --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --region us-west-2
```

以下示例显示了返回值：

```
{
    "Items": [
        {
            "ContentType": "image/jpeg",
            "LastModified": 1563571859.379,
            "Name": "filename.jpg",
            "Type": "OBJECT",
            "ETag":
 "543ab21abcd1a234ab123456a1a2b12345ab12abc12a1234abc1a2bc12345a12",
            "ContentLength": 3784
        },
        {
            "Type": "FOLDER",
            "Name": "ExampleLiveDemo"
        }
    ]
}
```

> **Note**
>
> 受 seconds_since_create 规则约束的对象不包括在 list-items 响应中。

查看特定文件夹中对象和文件夹的列表 (AWS CLI)

- 在中 AWS CLI，使用list-items命令，在请求的末尾加上指定的文件夹名称：

```
aws mediastore-data list-items --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path /folder_name --
region us-west-2
```

以下示例显示了返回值：

```
{
```

```
    "Items": [
        {
            "Type": "FOLDER",
            "Name": "folder_1"
        },
        {
            "LastModified": 1563571940.861,
            "ContentLength": 2307346,
            "Name": "file1234.jpg",
            "ETag":
 "111a1a22222a1a1a222abc333a444444b55ab1111ab2222222222ab333333a2b",
            "ContentType": "image/jpeg",
            "Type": "OBJECT"
        }
    ]
}
```

> **Note**
>
> 受 `seconds_since_create` 规则约束的对象不包括在 `list-items` 响应中。

查看对象的详细信息

在您上传对象后，AWS Elemental 会 MediaStore 存储修改日期、内容长度、 ETag （实体标签）和内容类型等详细信息。要了解如何使用某个对象的元数据，请参阅 MediaStore 与 HTTP 缓存的交互。

查看对象的详细信息（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择容器（包含要查看的对象）的名称。
3. 如果要查看的对象位于文件夹中，则继续选择文件夹名称，直到看到该对象。
4. 选择对象的名称。

   将显示详细信息页面，其中显示了有关对象的信息。

查看对象的详细信息 (AWS CLI)

- 在中 AWS CLI，使用以下`describe-object`命令：

```
aws mediastore-data describe-object --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path /folder_name/
file1234.jpg --region us-west-2
```

以下示例显示了返回值：

```
{
    "ContentType": "image/jpeg",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ContentLength": "2307346",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3"
}
```

下载对象

可以使用控制台下载对象。您可以使用下载对象或仅下载对象的一部分。 AWS CLI

下载对象（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择包含要下载的对象的容器的名称。
3. 如果要下载的对象位于文件夹中，则继续选择文件夹名称，直到看到该对象。
4. 选择对象的名称。
5. 在 Object (对象) 详细信息页面上，选择 Download (下载)。

下载对象 (AWS CLI)

- 在中 AWS CLI，使用以下`get-object`命令：

```
aws mediastore-data get-object --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path=/folder_name/
README.md README.md --region us-west-2
```

以下示例显示了返回值：

```
{
    "ContentLength": "2307346",
```

```
    "ContentType": "image/jpeg",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3",
    "StatusCode": 200
}
```

下载对象的一部分 (AWS CLI)

- 在中 AWS CLI，使用get-object命令并指定范围。

```
aws mediastore-data get-object --endpoint https://
aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path /folder_name/
README.md --range="bytes=0-100" README2.md --region us-west-2
```

以下示例显示了返回值：

```
{
    "StatusCode": 206,
    "ContentRange": "bytes 0-100/2307346",
    "ContentLength": "101",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ContentType": "image/jpeg",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3"
}
```

删除对象

AWS Elemental MediaStore 为从容器中删除对象提供了不同的选项：

- 删除单个对象。不收取费用。
- 清空容器来一次删除容器中的所有对象。由于此过程使用 API 调用，因此收取正常的 API 费用。
- 添加对象生命周期策略来在对象达到特定期限时删除对象。不收取费用。

删除对象

可以使用控制台或 AWS CLI逐个删除对象。或者，您可以添加对象生命周期策略，当对象在容器中达到特定期限后自动删除对象，也可以清空容器来删除该容器中的所有对象。

> **Note**
>
> 当您删除文件夹中唯一的对象时，AWS Elemental MediaStore 会自动删除该文件夹以及该文件夹上方的所有空文件夹。例如，假设有一个名为 `premium` 的文件夹，它不包含任何文件，但包含一个名为 `canada` 的子文件夹。`canada` 子文件夹包含一个名为 `mlaw.ts` 的文件。如果删除文件 `mlaw.ts`，则该服务将同时删除 `premium` 和 `canada` 文件夹。

删除对象（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择包含要删除的对象的容器的名称。
3. 如果要删除的对象位于文件夹中，则继续选择文件夹名称，直到看到该对象。
4. 选择对象名称左侧的选项。
5. 选择删除。

删除对象 (AWS CLI)

- 在中 AWS CLI，使用`delete-object`命令。

  示例：

  ```
  aws mediastore-data --region us-west-2 delete-object --endpoint=https://
  aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com --path=/folder_name/
  README.md
  ```

  此命令没有返回值。

清空容器

您可以清空容器来删除存储在容器中的所有对象。或者，您也可以添加对象生命周期策略，以在对象在容器中存储达到特定期限后自动删除对象，也可以单个删除对象。

清空容器（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择要清空的容器的选项。

3. 选择 Empty container (清空容器)。此时会显示确认消息。
4. 通过在文本字段中输入容器名称来确认要清空该容器，然后选择清空。

AWS Elemental 中的安全 MediaStore

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在云中运行 AWS 服务的基础架构 AWS 云。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 AWS Elemental 的合规计划 MediaStore，请参阅按合规计划提供的范围内的AWS 服务按合规计划划。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用时如何应用分担责任模型 MediaStore。以下主题向您介绍如何进行配置 MediaStore 以满足您的安全和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 MediaStore 资源。

主题

- AWS Elemental 中的数据保护 MediaStore
- 适用于 AWS Elemental 的身份和访问管理 MediaStore
- 登录和监控 AWS Elemental MediaStore
- AWS Elemental 的合规性验证 MediaStore
- AWS Elemental 中的弹性 MediaStore
- AWS Elemental 中的基础设施安全 MediaStore
- 防止跨服务混淆座席

AWS Elemental 中的数据保护 MediaStore

AWS 分担责任模型分担责任模型适用于 AWS Element MediaStore al 中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS 云。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS Security Blog 上的 AWS Shared Responsibility Model and GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 用于 SSL/TLS 与 AWS 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅《AWS CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API MediaStore 或以其他 AWS 服务 方式使用控制台 AWS CLI、API 或时 AWS SDKs。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，强烈建议您不要在 URL 中包含凭证信息来验证对该服务器的请求。

数据加密

MediaStore 使用行业标准 AES-256 算法对静态容器和对象进行加密。我们建议您 MediaStore 使用以下方式保护您的数据：

- 创建容器策略以控制对该容器中所有文件夹和对象的访问权限。有关更多信息，请参阅 the section called “容器策略”。
- 创建跨源资源共享 (CORS) 策略，允许有选择地跨域访问您的资源。 MediaStore 有了 CORS ，您可以允许在一个域中加载的客户端 Web 应用程序与另一个域中的资源交互。有关更多信息，请参阅 the section called “CORS 策略”。

适用于 AWS Elemental 的身份和访问管理 MediaStore

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（拥有权限）使用 MediaStore 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

受众

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅 对 AWS Elemental MediaStore 身份和访问进行故障排除 ）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅 AWS Elemental 如何与 IAM 配 MediaStore 合使用 ）
- IAM 管理员：编写用于管理访问权限的策略（请参阅 AWS Elemental 基于身份的策略示例 MediaStore ）

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center （例如（IAM Identity Center）、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录 用户指南》中的如何登录您的 AWS 账户。

对于编程访问， AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的AWS 签名版本 4。

AWS 账户 root 用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务 和资源。强烈建议您不要使用根用户执行日常任务。有关需要根用户凭证的任务，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户使用与身份提供商的联合身份验证才能 AWS 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Directory Service ，或者 AWS 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

要集中管理访问权限，建议使用 AWS IAM Identity Center。有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

IAM 用户和群组

IAM 用户是对单个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问。

IAM 组指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户的使用案例。

IAM 角色

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon 上运行的应用程序非常有用。 EC2有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略定义了与身份或资源关联时的权限。 AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员可使用策略来指定访问权限，具体做法是定义哪个主体可在何种条件下对哪些资源执行何种操作。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以代入这些角色。IAM 策略定义权限，而不考虑您使用哪种方法来执行操作。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可在何种条件下对哪些资源执行什么操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管式策略（附加到多个身份的独立策略）。要了解如何在托管式策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。您必须在基于资源的策略中指定主体。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

其他策略类型

AWS 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界：设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，请参阅《 IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs)-在中指定组织或组织单位的最大权限 AWS Organizations。有关更多信息，请参阅 AWS Organizations 用户指南中的服务控制策略。
- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略：在为角色或联合用户创建临时会话时，作为参数传递的高级策略。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

AWS Elemental 如何与 IAM 配 MediaStore 合使用

在使用 IAM 管理访问权限之前 MediaStore，请先了解有哪些 IAM 功能可供使用 MediaStore。

您可以在 AWS Elemental 中使用的 IAM 功能 MediaStore

IAM 功能	MediaStore 支持
基于身份的策略	是
基于资源的策略	是
策略操作	是
策略资源	是
策略条件键（特定于服务）	是
ACLs	否
ABAC（策略中的标签）	部分
临时凭证	是
主体权限	是
服务角色	是
服务相关角色	否

要全面了解 MediaStore 以及其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的AWS 服务。

基于身份的策略 MediaStore

支持基于身份的策略：是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

基于身份的策略示例 MediaStore

要查看 MediaStore 基于身份的策略的示例，请参阅。AWS Elemental 基于身份的策略示例 MediaStore

内部基于资源的政策 MediaStore

支持基于资源的策略：是

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

> **Note**
>
> MediaStore 还支持容器策略，这些策略定义了哪些委托人实体（账户、用户、角色和联合用户）可以在容器上执行操作。有关更多信息，请参阅 容器策略。

的政策行动 MediaStore

支持策略操作：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。在策略中包含操作以授予执行关联操作的权限。

要查看 MediaStore 操作列表，请参阅《服务授权参考》 MediaStore中的 AWS Elemental 定义的操作。

正在执行的策略操作在操作前 MediaStore 使用以下前缀：

```
mediastore
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "mediastore:action1",
      "mediastore:action2"
         ]
```

要查看 MediaStore 基于身份的策略的示例，请参阅。AWS Elemental 基于身份的策略示例 MediaStore

的政策资源 MediaStore

支持策略资源：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符（*）来指示此语句应用于所有资源。

```
"Resource": "*"
```

要查看 MediaStore 资源类型及其列表 ARNs，请参阅《服务授权参考》 MediaStore中的 AWS Elemental 定义的资源。要了解您可以使用哪些操作来指定每种资源的 ARN，请参阅 AWS Element al 定义的操作。 MediaStore

MediaStore 容器资源具有以下 ARN：

```
arn:${Partition}:mediastore:${Region}:${Account}:container/${containerName}
```

有关格式的更多信息 ARNs，请参阅 Amazon 资源名称 (ARNs) 和 AWS 服务命名空间。

例如，要在语句中指定 AwardsShow 容器，请使用以下 ARN：

```
"Resource": "arn:aws:mediastore:us-east-1:111122223333:container/AwardsShow"
```

的策略条件密钥 MediaStore

支持特定于服务的策略条件键：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

Condition 元素指定语句何时根据定义的标准执行。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

要查看 MediaStore 条件密钥列表，请参阅《服务授权参考》 MediaStore中的 AWS Elemental 条件密钥。要了解您可以使用条件键的操作和资源，请参阅 AWS Element MediaStore al 定义的操作。

要查看 MediaStore 基于身份的策略的示例，请参阅。AWS Elemental 基于身份的策略示例 MediaStore

ACLs in MediaStore

支持 ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

ABAC with MediaStore

支持 ABAC（策略中的标签）：部分支持

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于称为标签的属性来定义权限。您可以将标签附加到 IAM 实体和 AWS 资源，然后设计 ABAC 策略以允许在委托人的标签与资源上的标签匹配时进行操作。

要基于标签控制访问，您需要使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的使用 ABAC 授权定义权限。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

将临时证书与 MediaStore

支持临时凭证：是

临时证书提供对 AWS 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的临时安全凭证和使用 IAM 的AWS 服务。

的跨服务主体权限 MediaStore

支持转发访问会话（FAS）：是

转发访问会话 (FAS) 使用调用主体的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。有关发出 FAS 请求时的策略详细信息，请参阅转发访问会话。

MediaStore 的服务角色

支持服务角色：是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会中断 MediaStore 功能。只有在 MediaStore 提供操作指导时才编辑服务角色。

的服务相关角色 MediaStore

支持服务相关角色：否

服务相关角色是一种链接到的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅能够与 IAM 搭配使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

AWS Elemental 基于身份的策略示例 MediaStore

默认情况下，用户和角色没有创建或修改 MediaStore 资源的权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略（控制台）。

有关由 MediaStore定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅服务授权参考 MediaStore中的 AWS Elemental 的操作、资源和条件密钥。 ARNs

主题

- 策略最佳实践
- 使用 MediaStore 控制台
- 允许用户查看他们自己的权限

策略最佳实践

基于身份的策略决定了某人是否可以在您的账户中创建、访问或删除 MediaStore 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的AWS 托管式策略或工作职能的AWS 托管式策略。
- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言（JSON）和 IAM 最佳实

践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 验证策略。

- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅 IAM 用户指南中的 IAM 中的安全最佳实操。

使用 MediaStore 控制台

要访问 AWS Elemental MediaStore 控制台，您必须拥有一组最低权限。这些权限必须允许您列出和查看有关您的 MediaStore 资源的详细信息 AWS 账户。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍然可以使用 MediaStore 控制台，还需要将 MediaStore *ConsoleAccess*或*ReadOnly* AWS 托管策略附加到实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
```

```
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

对 AWS Elemental MediaStore 身份和访问进行故障排除

使用以下信息来帮助您诊断和修复在使用 MediaStore 和 IAM 时可能遇到的常见问题。

主题

- 我无权在以下位置执行操作 MediaStore
- 我无权执行 iam：PassRole
- 我想允许我以外的人 AWS 账户 访问我的 MediaStore 资源

我无权在以下位置执行操作 MediaStore

如果您收到错误提示，指明您无权执行某个操作，则必须更新策略以允许执行该操作。

当 `mateojackson` IAM 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `mediastore:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 mediastore:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 mateojackson 用户的策略，以允许使用 mediastore:*GetWidget* 操作访问 *my-example-widget* 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam：PassRole

如果您收到一个错误，表明您无权执行 iam:PassRole 操作，则必须更新策略以允许您将角色传递给 MediaStore。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 marymajor 的 IAM 用户尝试使用控制台在 MediaStore 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 iam:PassRole 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人 AWS 账户 访问我的 MediaStore 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解是否 MediaStore 支持这些功能，请参阅AWS Elemental 如何与 IAM 配 MediaStore 合使用。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

登录和监控 AWS Elemental MediaStore

本部分概括介绍了 AWS Elemental MediaStore 中出于安全目的而提供的日志记录和监控选项。有关 MediaStore 中日志记录和监控的更多信息，请参阅在 AWS Elemental 中进行监控和标记 MediaStore。

监控是维护 AWS 解决方案的可靠性、可用性和性能的重要组成部分。 AWS Elemental MediaStore 您应该从 AWS 解决方案的各个部分收集监控数据，以便在出现多点故障时可以更轻松地进行调试。AWS 提供了多种用于监控您的 MediaStore 资源和应对潜在事件的工具。

亚马逊 CloudWatch 警报

使用 CloudWatch 警报，您可以监视您指定的时间段内的单个指标。如果该指标超过给定的阈值，则会向 Amazon SNS 主题或 AWS Auto Scaling 策略发送通知。 CloudWatch 警报不会调用操作，因为它们处于特定状态。而是必须在状态已改变并在指定的若干个时间段内保持不变后才调用。有关更多信息，请参阅 使用监控 CloudWatch。

AWS CloudTrail 日志

CloudTrail 提供了用户、角色或 AWS 服务在中执行的操作的记录 AWS Elemental MediaStore。使用收集的信息 CloudTrail，您可以确定向哪个请求发出 MediaStore、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。有关更多信息，请参阅 使用 CloudTrail 记录 API 调用。

AWS Trusted Advisor

Trusted Advisor 借鉴了从为成千上万的 AWS 客户提供服务中学到的最佳实践。 Trusted Advisor 检查您的 AWS 环境，然后在有机会节省资金、提高系统可用性和性能或帮助填补安全漏洞时提出建议。所有 AWS 客户都可以获得五张 Trusted Advisor 支票。拥有商业或企业支持计划的客户可以查看所有 Trusted Advisor 支票。

有关更多信息，请参阅 AWS Trusted Advisor。

AWS Elemental 的合规性验证 MediaStore

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅AWS 安全文档。

AWS Elemental 中的弹性 MediaStore

AWS 全球基础设施是围绕 AWS 区域 可用区构建的。 AWS 区域 提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络连接。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错能力和可扩展性。

有关 AWS 区域 和可用区的更多信息，请参阅AWS 全球基础设施。

除了 AWS 全球基础架构外，还 MediaStore 提供多项功能来帮助支持您的数据弹性和备份需求。

AWS Elemental 中的基础设施安全 MediaStore

作为一项托管服务，AWS Elemen MediaStore t AWS al 受到全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅AWS 云安全。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用 MediaStore 通过网络进行访问。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

防止跨服务混淆座席

混淆代理问题是一个安全性问题，即不具有操作执行权限的实体可能会迫使具有更高权限的实体执行该操作。在中 AWS，跨服务模仿可能会导致混乱的副手问题。一个服务（呼叫服务）调用另一项服务（所谓的服务）时，可能会发生跨服务模拟。可以操纵调用服务，使用其权限以在其他情况下该服务不应有访问权限的方式对另一个客户的资源进行操作。为防止这种情况， AWS 提供可帮助您保护所有服务的数据的工具，而这些服务中的服务主体有权限访问账户中的资源。

我们建议在资源策略中使用aws:SourceArn和aws:SourceAccount全局条件上下文密钥来限制 AWS Elemental 向该 MediaStore 资源提供的其他服务的权限。如果您只希望将一个资源与跨服务访问相关联，请使用 aws:SourceArn。如果您想允许该账户中的任何资源与跨服务使用操作相关联，请使用 aws:SourceAccount。

防范混淆代理问题最有效的方法是使用 aws:SourceArn 全局条件上下文键和资源的完整 ARN。如果不知道资源的完整 ARN，或者正在指定多个资源，请针对 ARN 未知部分使用带有通配符字符 (*) 的 aws:SourceArn 全局上下文条件键。例如 arn:aws:*servicename*:*:*123456789012*:*。

如果 aws:SourceArn 值不包含账户 ID，例如 Amazon S3 存储桶 ARN，您必须使用两个全局条件上下文键来限制权限。

的值aws:SourceArn必须是在您的区域和账户中 MediaStore 发布 CloudWatch 日志的配置。

以下示例显示了如何在中使用aws:SourceArn和aws:SourceAccount全局条件上下文键 MediaStore 来防止出现混淆的副手问题。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ConfusedDeputyPreventionExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "mediastore.amazonaws.com"
      },
      "Action": "mediastore:CreateContainer",
      "Resource": [
        "arn:aws:mediastore:us-east-2:333333333333:container/ResourceName/*"
      ],
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:mediastore:*:333333333333:*"
        },
        "StringEquals": {
          "aws:SourceAccount": "333333333333"
        }
      }
    }
  ]
}
```

在 AWS Elemental 中进行监控和标记 MediaStore

监控是维护 AWS Elemental 和其他 AWS 解决方案的可靠性、可用 MediaStore 性和性能的重要组成部分。 AWS 提供以下监控工具 MediaStore，供您监视、报告问题并在适当时自动采取措施：

- AWS CloudTrail捕获由您的账户或代表您的 AWS 账户进行的 API 调用和相关事件，并将日志文件传输到您指定的 Amazon S3 存储桶。您可以识别哪些用户和帐户拨打了电话 AWS、发出呼叫的源 IP 地址以及呼叫发生的时间。有关更多信息，请参阅 AWS CloudTrail 《用户指南》。
- Amazon 会实时 CloudWatch监控您的 AWS 资源和您运行 AWS 的应用程序。您可以收集和跟踪指标，创建自定义的控制平面，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以 CloudWatch 跟踪您的 Amazon EC2 实例的 CPU 使用率或其他指标，并在需要时自动启动新实例。有关更多信息，请参阅 Amazon CloudWatch 用户指南。
- Amazon CloudWatch Events 提供了一系列描述 AWS 资源变化的系统事件。通常， AWS 服务会在几秒钟内向 CloudWatch 事件发送事件通知，但有时可能需要一分钟或更长时间。 CloudWatch 事件支持事件驱动的自动计算，因为您可以编写规则来监视某些事件，并在这些事件发生时在其他 AWS 服务中触发自动操作。有关更多信息，请参阅 Amazon Ev CloudWatch ents 用户指南。
- Amazon Lo CloudWatch gs 使您能够监控、存储和访问来自亚马逊 EC2 实例和其他来源的日志文件。 CloudTrail CloudWatch 日志可以监视日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch 日志用户指南。

您还可以以标签的形式为 MediaStore 容器分配元数据。每个标签都是包含您定义的一个键和值的标记。利用标签可以更轻松地管理、搜索和筛选资源。您可以使用标签在 AWS 管理控制台中整理 AWS 资源，创建所有资源的使用情况和账单报告，并在基础设施自动化活动期间筛选资源。 AWS

主题

使用记录 AWS Elemental MediaStore API 调用 AWS CloudTrail

AWS Elemental MediaStore 与 AWS CloudTrail一项服务集成，可记录用户、角色或 AWS 服务在中执行的操作。 MediaStore CloudTrail 捕获一部分 API 调用 MediaStore 作为事件，包括来自 MediaStore 控制台的调用和对 MediaStore API 的代码调用。如果您创建了跟踪，则可以允许将 CloudTrail 事件持

续传输到 Amazon S3 存储桶，包括的事件 MediaStore。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向哪个请求发出 MediaStore、发出请求的 IP 地址、谁发出了请求、何时发出请求等等。

要了解更多信息 CloudTrail，包括如何配置和启用它，请参阅《AWS CloudTrail 用户指南》。

主题

AWS Elemental 信息 MediaStore位于 CloudTrail

CloudTrail 在您创建 AWS 账户时已在您的账户上启用。当 AWS Elemental 中出现支持的事件活动时 MediaStore，该活动会与其他 AWS 服务 CloudTrail事件一起记录在事件历史记录中。您可以在自己的 AWS 账户中查看、搜索和下载最近发生的事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录您 AWS 账户中的事件，包括的事件 MediaStore，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅以下主题：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

AWS Elemental MediaStore 支持将以下操作作为事件记录在 CloudTrail 日志文件中：

- `CreateContainer`
- `DeleteContainer`
- `DeleteContainerPolicy`
- `DeleteCorsPolicy`
- `DescribeContainer`
- `GetContainerPolicy`

- GetCorsPolicy
- ListContainers
- PutContainerPolicy
- PutCorsPolicy

每个事件或日志条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户凭证还是用户凭证发出的
- 请求是使用角色还是联合用户的临时安全凭证发出的
- 请求是否由其他 AWS 服务发出

有关更多信息，请参阅 CloudTrail userIdentity 元素。

示例：AWS Elemental MediaStore 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。 CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会以任何特定顺序显示。

以下示例显示了演示该CreateContainer操作的 CloudTrail 日志条目：

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "ABCDEFGHIJKL123456789",
        "arn": "arn:aws:iam::111122223333:user/testUser",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "testUser",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2018-07-09T12:55:42Z"
            }
        },
        "invokedBy": "signin.amazonaws.com"
    },
```

```
    "eventTime": "2018-07-09T12:56:54Z",
    "eventSource": "mediastore.amazonaws.com",
    "eventName": "CreateContainer",
    "awsRegion": "ap-northeast-1",
    "sourceIPAddress": "54.239.119.16",
    "userAgent": "signin.amazonaws.com",
    "requestParameters": {
        "containerName": "TestContainer"
    },
    "responseElements": {
        "container": {
            "status": "CREATING",
            "creationTime": "Jul 9, 2018 12:56:54 PM",
            "name": " TestContainer ",
            "aRN": "arn:aws:mediastore:ap-northeast-1:111122223333:container/
TestContainer"
        }
    },
    "requestID":
 "MNCTGH4HRQJ27GRMBVDPIVHEP4LO2BN6MUVHBCPSHOAWNSOKSXCO24B2UEOBBND5DONRXTMFK3TOJ4G7AHWMESI",
    "eventID": "7085b140-fb2c-409b-a329-f567912d704c",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


使用亚马逊监控 AWS Elemental MediaStore CloudWatch

您可以使用监控 AWS Elemental MediaStore CloudWatch，它会收集原始数据并将其处理为可读的指标。 CloudWatch 保存 15 个月的统计数据，以便您可以访问历史信息并更好地了解 Web 应用程序或服务的性能。还可以设置特定阈值监视警报，在达到对应阈值时发送通知或采取行动。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

AWS 提供以下监控工具 MediaStore，供您监视、报告问题并在适当时自动采取措施：

- Amazon CloudWatch Logs 允许您监控、存储和访问来自 AWS Element MediaStore al 等 AWS 服务的日志文件。您可以使用 CloudWatch 日志通过日志数据监控应用程序和系统。例如，CloudWatch 日志可以跟踪应用程序日志中发生的错误数量，并在错误率超过您指定的阈值时向您发送通知。 CloudWatch 日志使用您的日志数据进行监控，因此无需更改代码。例如，您可以监控应用程序日志中的特定字面术语（例如 ValidationException “”），或者计算在特定时间段内发出的`PutObject`请求数量。找到您要搜索的术语后， CloudWatch Logs 会将数据报告给您指定的 CloudWatch 指标。日志数据会在传输期间加密，并且会对静态日志数据加密。

- Amazon CloudWatch Events 提供描述 AWS 资源（例如 MediaStore 对象）变化的系统事件。通常， AWS 服务会在几秒钟内向 CloudWatch 事件发送事件通知，但有时可能需要一分钟或更长时间。您可以设置规则来匹配事件（例如`DeleteObject`请求），并将它们路由到一个或多个目标函数或流。 CloudWatch 事件在发生时就会意识到操作变化。此外，E CloudWatch vents 还会通过发送消息以响应环境、激活功能、进行更改和捕获状态信息来响应这些操作更改并在必要时采取纠正措施。

CloudWatch 日志

访问日志记录提供对容器中的对象发出的请求的详细记录。对于许多应用程序来说访问日志很有用，例如在安全和访问审核方面。他们还可以帮助您了解客户群并了解您的 MediaStore 账单。 CloudWatch 日志分为以下几类：

- 日志流是共享同一来源的一系列日志事件。
- 日志组是一组具有相同保留期、监控和访问控制设置的日志流。在容器上启用访问日志时， MediaStore 会创建一个名称为（如）的日志组`/aws/mediastore/MyContainerName`。您可以定义日志组并指定向各组中放入哪些流。对可属于一个日志组的日志流数没有配额。

默认情况下，日志将无限期保留且永不过期。您可以调整每个日志组的保留策略，保持无限期保留或选择介于一天到 10 年之间的保留期。

为 Amazon 设置权限 CloudWatch

使用 AWS Identity and Access Management (IAM) 创建一个角色来授予 AWS Elemental MediaStore 访问亚马逊的权限。 CloudWatch您必须执行以下步骤才能为您的账户发布 CloudWatch 日志。 CloudWatch 自动发布您账户的指标。

允许 MediaStore 访问 CloudWatch

1. 使用 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在 IAM 控制台的导航窗格中，选择 Policies（策略），然后选择 Create policy（创建策略）。
3. 选择 JSON 选项卡，然后粘贴以下策略：

   JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups",
                "logs:CreateLogGroup"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/mediastore/*"
        }
    ]
}
```

此策略 MediaStore 允许您为 AWS 账户内任何区域中的任何容器创建日志组和日志流。

4. 选择查看策略。
5. 在 Review policy (查看策略) 页面上，对于 Name (名称)，输入 **`MediaStoreAccessLogsPolicy`**，然后选择 Create policy (创建策略)。
6. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
7. 选择 其他 Amazon Web Services 账户 角色类型。
8. 在账户 ID 中，输入您的 AWS 账户 ID。
9. 选择下一步: 权限。
10. 在搜索框中，输入 **`MediaStoreAccessLogsPolicy`**。
11. 选择您的新策略旁边的复选框，然后选择 Next: Tags (下一步：标记)。
12. 选择 Next: Review (下一步: 审核) 以预览您的新用户。
13. 对于角色名称，输入 **`MediaStoreAccessLogs`**，然后选择创建角色。
14. 在确认信息中，选择您刚刚创建的角色的名称 (**`MediaStoreAccessLogs`**)。
15. 在角色的 Summary (摘要) 页上，选择 Trust relationship (信任关系) 选项卡。
16. 选择编辑信任关系。

17. 在策略文档中，将委托人更改为 MediaStore 服务。它应如下所示：

```
"Principal": {
   "Service": "mediastore.amazonaws.com"
},
```

整个策略的内容现在应如下所示：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "mediastore.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

18. 选择更新信任策略。

为容器启用访问日志记录

默认情况下，AWS Elemental MediaStore 不收集访问日志。当您在容器上启用访问日志时，MediaStore 会将存储在该容器中的对象的访问日志发送给 Amazon CloudWatch。访问日志记录提供对容器中存储的任何对象发出的请求的详细记录。这些信息可以包括请求类型、请求中指定的资源以及处理请求的时间和日期。

> **Important**
>
> 在 MediaStore 容器中启用访问日志记录不会额外收取费用。但是，服务提交给您的任何日志文件都会产生普通存储费用。（您可以随时删除日志文件。）AWS 不会因日志文件的传输而收取数据传输费，但会按正常数据传输费率对访问日志文件收费。

启用访问日志记录 (AWS CLI)

- 在中 AWS CLI，使用以下`start-access-logging`命令：

```
aws mediastore start-access-logging --container-name LiveEvents --region us-west-2
```

此命令没有返回值。

对容器禁用访问日志记录

当您在容器上禁用访问日志时，AWS Elemental MediaStore 会停止向亚马逊发送访问日志。CloudWatch这些访问日志不会保存，也不可供检索。

禁用访问日志记录 (AWS CLI)

- 在中 AWS CLI，使用以下`stop-access-logging`命令：

```
aws mediastore stop-access-logging --container-name LiveEvents --region us-west-2
```

此命令没有返回值。

对 AWS Elemental 中的访问日志进行故障排除 MediaStore

当 AWS Elemental MediaStore 访问日志未出现在亚马逊上时 CloudWatch，请参阅下表，了解可能的原因和解决方案。

> **Note**
>
> 请务必启用 “ AWS CloudTrail 日志” 以协助故障排除过程。

症状	该问题可能是...	尝试这项操作...
即使启用了 CloudTrail 日志，您也看不到任何 CloudTrail 事件。	该 IAM 角色不存在或具有不正确的名称、权限或信任策略。	使用正确的名称、权限和信任策略创建一个角色。请参阅the section called “为设置权限 CloudWatch”。

症状	该问题可能是...	尝试这项操作...
您提交了一个 DescribeContainer API 请求，但响应显示此 AccessLoggingEnabled 参数的值为 False。此外，您还没有看到有关 MediaStoreAccessLogs 角色成功进行 DescribeLogGroup 、CreateLogGroup 、DescribeLogStream 或 CreateLogStream 调用的任何 CloudTrail 事件。	该 IAM 角色不存在或具有不正确的名称、权限或信任策略。	使用正确的名称、权限和信任策略创建一个角色。请参阅the section called “为设置权限 CloudWatch”。
	容器上未启用访问日志记录。	启用容器的访问日志记录。请参阅the section called “启用访问日志记录”。

在 CloudTrail 控制台上，您会看到一个事件，其中包含与该MediaStoreAccessLogs 角色相关的访问被拒绝错误。该 CloudTrail 事件可能包括以下几行：
 	
`"eventSource": "logs.amazonaws.com",`
 	
`"errorCode": "AccessDenied",`
 	
`"errorMessage": "User: arn:aws:sts::111122223333:assumed-role/MediaStoreAccessLogs/MediaStoreAccessLogsSession is not authorized to perform: logs:DescribeLogGroups on resource: arn:aws:logs:us-west-2:111122223333:log-group::log-stream:",`
	IAM 角色没有 AWS Element MediaStoreal 的正确权限。	更新 IAM 角色来获得正确的权限和信任策略。请参阅the section called “为设置权限 CloudWatch”。

症状	该问题可能是...	尝试这项操作...
您没有看到整个容器或多个容器的任何日志。	您的账户可能已超过每个区域每个账户的日志组 CloudWatch 配额。请参阅 Amazon 日志用户指南中的 CloudWatch 日志组配额。	在 CloudWatch 控制台上，确定您的账户是否已达到日志组 CloudWatch 配额。如有必要，您可以请求提高配额。
你会看到一些登录日志 CloudWatch，但不是所有你期望看到的日志。	您的账户可能已超过每个地区每个账户每秒交易的 CloudWatch 配额。请参阅《Amazon CloudWatch 日志用户指南》PutLogEvents 中的配额。	申请增加每个区域每个账户每秒 CloudWatch 交易的配额。

访问日志格式

访问日志文件由一系列 JSON 格式的日志记录组成，其中每个日志记录代表一个请求。日志中字段的顺序可能会变化。以下是示例日志，其中包括两个日志记录：

```
{
  "Path": "/FootballMatch/West",
  "Requester": "arn:aws:iam::111122223333:user/maria-garcia",
  "AWSAccountId": "111122223333",
  "RequestID":
 "aaaAAA111bbbBBB222cccCCC333dddDDD444eeeEEE555fffFFF666gggGGG777hhhHHH888iiiIII999jjjJJJ",
  "ContainerName": "LiveEvents",
  "TotalTime": 147,
  "BytesReceived": 1572864,
  "BytesSent": 184,
  "ReceivedTime": "2018-12-13T12:22:06.245Z",
  "Operation": "PutObject",
  "ErrorCode": null,
  "Source": "192.0.2.3",
  "HTTPStatus": 200,
```

```
  "TurnAroundTime': 7,
  "ExpiresAt": "2018-12-13T12:22:36Z"
}
 {
  "Path": "/FootballMatch/West",
  "Requester": "arn:aws:iam::111122223333:user/maria-garcia",
  "AWSAccountId": "111122223333",
  "RequestID":
 "ddDDD444eeeEEE555fffFFF666gggGGG777hhhHHH888iiiIII999jjjJJJ000cccCCC333bbbBBB222aaaAAA",
  "ContainerName": "LiveEvents",
  "TotalTime": 3,
  "BytesReceived": 641354,
  "BytesSent": 163,
  "ReceivedTime": "2018-12-13T12:22:51.779Z",
  "Operation": "PutObject",
  "ErrorCode": "ValidationException",
  "Source": "198.51.100.15",
  "HTTPStatus": 400,
  "TurnAroundTime": 1,
  "ExpiresAt": null
}
```

以下列表介绍日志记录字段：

AWSAccount我是

用于发出请求的账户的账户 ID。 AWS

BytesReceived

MediaStore 服务器接收的请求正文中的字节数。

BytesSent

MediaStore 服务器发送的响应正文中的字节数。此值通常与服务器响应包含的 `Content-Length` 标头的值相同。

ContainerName

接收请求的容器的名称。

ErrorCode

MediaStore 错误代码（例如`InternalServerError`）。如果没有发生任何错误，则显示 `-` 字符。即使状态代码为 200 也可能出现错误代码（服务器开始流式处理响应后，指示已关闭连接或错误）。

ExpiresAt

对象的到期日期和时间。此值基于应用于容器的生命周期策略transient data rule中设置的到期时间。该值是 ISO-8601 日期时间，基于为此请求提供服务的主机的系统时钟。如果生命周期策略没有适用于该对象的临时数据规则，或者没有对容器应用生命周期策略，则此字段的值为null。此字段仅适用于以下操作：PutObject、GetObject、DescribeObject、和DeleteObject。

HTTPStatus

响应的数字 HTTP 状态代码。

操作

已执行的操作，如 PutObject 或 ListItems。

路径

容器中存储对象的路径。如果操作没有使用路径参数，则会显示 - 字符。

ReceivedTime

收到请求的日期时间。该值是 ISO-8601 日期时间，基于为此请求提供服务的主机的系统时钟。

请求者

用于发出请求的账户的 Amazon 资源名称 (ARN) 。对于未经身份验证的请求，此值为 anonymous。如果在身份验证完成之前请求失败，则日志中可能会丢失此字段。对于此类请求，ErrorCode 可能会标识授权问题。

RequestID

由 AWS Elemental 生成的字符串 MediaStore ，用于唯一标识每个请求。

来源

请求者或进行调用的 AWS 服务的服务委托人的显式 Internet 地址。如果中间代理和防火墙隐藏发送请求的计算机的地址，值将设为空。

TotalTime

从服务器的角度传输请求的毫秒数。该值的测量从服务收到请求的时间开始，到发出响应的最后一个字节的时间结束。该值从服务器的角度来测量，因为从客户端角度测得的值受网络延迟影响。

TurnAroundTime

处理您的请求所 MediaStore 花费的毫秒数。该值计算从收到您的请求的最后一个字节到发出响应的第一个字节的时间。

日志中字段的顺序可能会发生变化。

日志记录状态更改将逐渐生效

容器的日志记录状态更改需要一定时间才能实际影响日志文件的传输。例如，如果您为容器 A 启用了日志记录，则可能记录在以下时间内发送的一些请求，而不会记录其他请求。如果您禁用容器 B 的日志记录，在接下来的一个小时里可能有一些日志被继续传输，而其他则可能不会。在所有情况下，新的设置最终都将生效，而您无需执行任何更多操作。

最大努力服务器日志传输

访问日志记录会以最大努力进行传输。针对已正确配置了日志记录的容器的大多数请求会导致传输一条日志记录。大多数日志记录将在记录后的几小时内传输，但可以更频繁地传输这些记录。

因此，不能保证访问日志记录的完整性和即时性。特殊请求的日志记录可能会在实际处理了请求之后进行传输，也可能根本不会传输。访问日志的用途在于向您提供有关容器流量性质方面的信息。丢失日志记录的情况十分少见，但是访问日志记录不旨在完整记录所有请求。

根据访问日志记录功能的最大努力性质，在 AWS 门户上提供的使用率报告（AWS 管理控制台上的账单和成本管理报告）中可能有一个或多个访问请求不会出现在传输的访问日志中。

访问日志格式的编程注意事项

有时我们可能会通过添加新字段来扩展访问日志格式。必须写入可解析访问日志的代码以处理额外的未知字段。

CloudWatch 活动

Amazon CloudWatch Events 使您能够实现 AWS 服务自动化，并自动响应系统事件，例如应用程序可用性问题或资源更改。您可以编写简单规则来指示您关注的事件，并指示要在事件匹配规则时执行的自动化操作。

> **Important**
>
> 通常， AWS 服务会在几秒钟内向 CloudWatch 事件发送事件通知，但有时可能需要一分钟或更长时间。

将文件上传到容器或从容器中移除时， CloudWatch 服务中会连续触发两个事件：

1. the section called “对象状态更改事件”

2. the section called “容器状态更改事件”

有关订阅这些活动的信息，请参阅 Amazon CloudWatch。

可自动触发的操作包括：

- 调用函数 AWS Lambda
- 调用 Amazon EC2 运行命令
- 将事件中继到 Amazon Kinesis Data Streams
- 激活 AWS Step Functions 状态机
- 通知 Amazon SNS 主题或队列 AWS SMS

在 AWS Elemental 中使用 CloudWatch 事件的一些示例 MediaStore 包括：

- 创建容器时激活 Lambda 函数。
- 删除对象时通知 Amazon SNS 主题。

有关更多信息，请参阅 Amazon Ev CloudWatch ents 用户指南。

主题

- AWS Elemental MediaStore 对象状态更改事件
- AWS Elemental MediaStore 容器状态更改事件

AWS Elemental MediaStore 对象状态更改事件

当对象的状态更改（对象已上传或删除）时，将发布此事件。

> **Note**
>
> 由于临时数据规则而过期的对象在过期时不会发出 CloudWatch事件。

有关订阅此活动的信息，请参阅 Amazon CloudWatch。

对象已更新

```
{
```

```
  "version": "1",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "MediaStore Object State Change",
  "source": "aws.mediastore",
  "account": "111122223333",
  "time": "2017-02-22T18:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:mediastore:us-east-1:111122223333:MondayMornings/Episode1/
Introduction.avi"
  ],
  "detail": {
    "ContainerName": "Movies",
    "Operation": "UPDATE",
    "Path":"TVShow/Episode1/Pilot.avi",
    "ObjectSize":123456,
    "URL": "https://a832p1qeaznlp9.files.mediastore-us-west-2.com/Movies/
MondayMornings/Episode1/Introduction.avi"
  }
}
```

对象已删除

```
{
  "version": "1",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "MediaStore Object State Change",
  "source": "aws.mediastore",
  "account": "111122223333",
  "time": "2017-02-22T18:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:mediastore:us-east-1:111122223333:Movies/MondayMornings/Episode1/
Introduction.avi"
  ],
  "detail": {
    "ContainerName": "Movies",
    "Operation": "REMOVE",
    "Path":"Movies/MondayMornings/Episode1/Introduction.avi",
    "URL": "https://a832p1qeaznlp9.files.mediastore-us-west-2.com/Movies/
MondayMornings/Episode1/Introduction.avi"
  }
}
```

AWS Elemental MediaStore 容器状态更改事件

当容器的状态更改（容器已添加或删除）时，将发布此事件。有关订阅此活动的信息，请参阅 Amazon CloudWatch。

容器已创建

```
{
  "version": "1",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "MediaStore Container State Change",
  "source": "aws.mediastore",
  "account": "111122223333",
  "time": "2017-02-22T18:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:mediastore:us-east-1:111122223333:container/Movies"
  ],
  "detail": {
    "ContainerName": "Movies",
    "Operation": "CREATE"
    "Endpoint": "https://a832p1qeaznlp9.mediastore-us-west-2.amazonaws.com"
  }
}
```

容器已删除

```
{
  "version": "1",
  "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
  "detail-type": "MediaStore Container State Change",
  "source": "aws.mediastore",
  "account": "111122223333",
  "time": "2017-02-22T18:43:48Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:mediastore:us-east-1:111122223333:container/Movies"
  ],
  "detail": {
    "ContainerName": "Movies",
    "Operation": "REMOVE"
  }
```

```
}
```

使用 MediaStore 亚马逊指标监控 AWS Elemental CloudWatch

您可以使用监控 AWS Elemental MediaStore CloudWatch，它会收集原始数据并将其处理为可读的指标。 CloudWatchkeeps 统计数据保存 15 个月，这样您就可以访问历史信息并更好地了解您的 Web 应用程序或服务的性能。还可以设置特定阈值监视警报，在达到对应阈值时发送通知或采取行动。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

对于 AWS Elemental MediaStore，当该指标达到特定阈值时，您可能需要关注`BytesDownloaded`并向自己发送一封电子邮件。

使用 CloudWatch 控制台查看指标

指标的分组首先依据服务命名空间，然后依据每个命名空间内的各种维度组合。

1. 登录 AWS 管理控制台 并打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择指标。
3. 在所有指标下，选择 AWS/ MediaStore 命名空间。
4. 选择指标维度查看指标。例如，选择 `Request metrics by container` 查看已发送到容器的不同类型请求的指标。

要查看指标，请使用 AWS CLI

- 在命令提示符处输入下面的命令：

```
aws cloudwatch list-metrics --namespace "AWS/MediaStore"
```

AWS Elemental 指标 MediaStore

下表列出了 AWS Elemental MediaStore 发送到的指标。 CloudWatch

> **Note**
>
> 要查看指标，您必须在容器中添加指标策略 MediaStore 以允许向 Amazon 发送指标 CloudWatch。

指标	描述
RequestCount	对 MediaStore 容器发出的 HTTP 请求总数，按操作类型（Put、Get、Delete、Describe、List）划分。 单位：计数 有效维度： • 容器名称 • 对象组名称 • 请求类型 有效统计数据：Sum
4xxErrorCount	向其发出的 HTTP 请求数量 MediaStore 导致了 4xx 错误。 单位：计数 有效维度： • 容器名称 • 对象组名称 • 请求类型 有效统计数据：Sum
5xxErrorCount	向其发出的 HTTP 请求数量 MediaStore 导致了 5xx 错误。 单位：计数 有效维度： • 容器名称 • 对象组名称 • 请求类型

指标	描述
	有效统计数据：Sum
BytesUploaded	为向 MediaStore 容器发出的请求上传的字节数（请求包含正文）。 单位：字节 有效维度： • 容器名称 • 对象组名称 有效统计数据：平均值（每个请求的字节数）、总和（每个周期的字节数）、样本数、最小值（与 P0.0 相同）、最大值（与 p100 相同）、p0.0 到 p99.9 的任何百分位数
BytesDownloaded	为向 MediaStore 容器发出的请求下载的字节数（响应包含正文）。 单位：字节 有效维度： • 容器名称 • 对象组名称 有效统计数据：平均值（每个请求的字节数）、总和（每个周期的字节数）、样本数、最小值（与 P0.0 相同）、最大值（与 p100 相同）、p0.0 到 p99.9 的任何百分位数

指标	描述
TotalTime	从服务器的角度传输请求的毫秒数。该值是从 MediaStore 收到您的请求到它发送响应的最后一个字节的时间开始计算的。该值从服务器的角度来测量，因为从客户端角度测得的值受网络延迟影响。 单位：毫秒 有效维度： • 容器名称 • 对象组名称 • 请求类型 有效统计数据：平均值、最小值（与 P0.0 相同）、最大值（与 p100 相同）、p0.0 到 p100 的任何百分位数
TurnaroundTime	处理您的请求所 MediaStore 花费的毫秒数。该值是从 MediaStore 收到请求的最后一个字节到发送响应的第一个字节的时间开始计算的。 单位：毫秒 有效维度： • 容器名称 • 对象组名称 • 请求类型 有效统计数据：平均值、最小值（与 P0.0 相同）、最大值（与 p100 相同）、p0.0 到 p100 的任何百分位数

指标	描述
ThrottleCount	向其发出的 HTTP 请求数量 MediaStore 受到限制。 单位：计数 有效维度： • 容器名称 • 对象组名称 • 请求类型 有效统计数据：Sum

标记 AWS MediaStore Elemental 资源

标签是您分配或分配给 AWS 资源的自定义属性标签。 AWS 每个 标签具有两个部分：

- 标签键（例如，CostCenter、Environment 或 Project）。标签键区分大小写。
- 一个称为标签值 的可选字段（例如，111122223333 或 Production）。省略标签值与使用空字符串效果相同。与标签键一样，标签值区分大小写。

标签可帮助您：

- 识别和整理您的 AWS 资源。许多 AWS 服务支持标记，因此，您可以将同一标签分配给来不同服务的资源，以指示这些资源是相关的。例如，您可以为 AWS Elemental 分配与分配给 MediaStore *container*输入的标签相同的标签。 AWS Elemental MediaLive
- 跟踪您的 AWS 成本。您可以在 AWS 账单与成本管理 控制面板上激活这些标签。AWS 使用标签对您的成本进行分类，并向您提供每月成本分配报告。有关更多信息，请参阅 AWS Billing 《用户指南》 中的使用成本分配标签。

以下各节提供了有关 AWS Element MediaStore al 标签的更多信息。

AWS Elemental 中支持的资源 MediaStore

AWS Elemental 中的以下资源 MediaStore 支持标记：

- *container*

有关添加和管理标签的信息，请参阅管理标签。

AWS Elemental MediaStore 不支持 AWS Identity and Access Management (IAM) 的基于标签的访问控制功能。

标签命名和使用约定

以下基本命名和使用惯例适用于在 AWS Elemental MediaStore 资源中使用标签：

- 每个资源最多可以有 50 个标签。
- 对于每个资源，每个标签键都必须是唯一的，每个标签键只能有一个值。
- 最大标签键长度为 128 个 Unicode 字符 (采用 UTF-8 格式)。
- 最大标签值长度为 256 个 Unicode 字符 (采用 UTF-8 格式)。
- 允许使用的字符包括可用 UTF-8 格式表示的字母、数字和空格，以及以下字符：. : + = @ _ / - (连字符) 。Amazon EC2 资源允许使用任何字符。
- 标签键和值区分大小写。最佳实践是，决定利用标签的策略并在所有资源类型中一致地实施该策略。例如，决定是使用 `Costcenter`、`costcenter` 还是 `CostCenter`，以及是否对所有标签使用相同的约定。避免将类似的标签用于不一致的案例处理。
- 该`aws:`前缀禁止用于标记；它是保留供 AWS 使用的。无法编辑或删除带此前缀的标签键或值。具有此前缀的标签不计入每个资源的标签数配额。

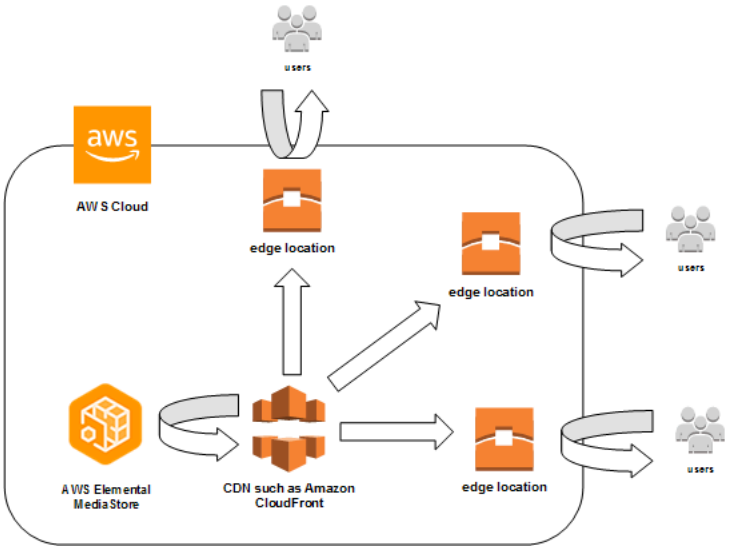
管理标签

标签由资源上的 `Key` 和 `Value` 属性构成。您可以使用 AWS CLI 或 MediaStore API 来添加、编辑或删除这些属性的值。有关使用标签的信息，请参阅《AWS Elemental MediaStore API 参考》中的以下章节：

- CreateContainer
- ListTagsForResource
- 资源
- TagResource
- UntagResource

与内容交付网络合作 (CDNs)

您可以使用诸如亚马逊之类的内容分发网络 (CDN) CloudFront 来提供您存储在 AWS MediaStore Elemental 中的内容。CDN 是一组全球分布的服务器，可缓存视频等内容。当用户请求您的内容时，CDN 会将请求路由至延迟最低的边缘站点。如果您的内容已缓存在该边缘站点中，CDN 将立即传送它。如果您的内容当前不在该边缘位置，CDN 会从您的来源（例如您的 MediaStore 容器）检索内容并将其分发给用户。



主题

- 允许亚马逊 CloudFront 访问您的 AWS Elemental MediaStore 容器
- AWS Elemental 与 HT MediaStore TP 缓存的交互

允许亚马逊 CloudFront 访问您的 AWS Elemental MediaStore 容器

您可以使用亚马逊 CloudFront 来提供您存储在 AWS Element MediaStore al 容器中的内容。您可以通过下列方式之一来执行此操作：

- 使用来源访问控制 (OAC)-（推荐）如果您 AWS 区域 支持 OAC 功能，请使用此选项。 CloudFront
- 使用共享密钥-如果您 AWS 区域 不支持 OAC 功能，请使用此选项。 CloudFront

使用来源访问控制 (OAC)

您可以使用亚马逊的源站访问控制 (OAC) 功能，通过提高安全性 CloudFront 来保护 AWS MediaStore Elemental 源代码的安全。您可以对 MediaStore源请求启用AWS 签名版本 4 (Sigv4)，并设置何时以

及是否 CloudFront 应该对 CloudFront 请求进行签名。您可以 CloudFront 通过控制台 APIs、SDK 或 CLI 访问的 OAC 功能，使用该功能无需支付额外费用。

有关将 OAC 功能与配合使用的更多信息 MediaStore，请参阅《亚马逊 CloudFront 开发者指南》中的限制对 MediaStore 源的访问。

使用共享密钥

如果您 AWS 区域 不支持亚马逊的 OAC 功能 CloudFront，则可以将策略附加到您的 AWS MediaStore Elemental 容器，授予读取权限或更高的读取权限。 CloudFront

> **Note**
>
> 如果您 AWS 区域 支持 OAC 功能，我们建议您使用该功能。以下过程要求您配置 MediaStore 和 CloudFront 使用共享密钥，以限制对 MediaStore 容器的访问。为了遵循最佳安全实践，此手动配置需要定期轮换密钥。在 MediaStore 源上启用 OAC，您可以指示使用 Sigv4 CloudFront 对请求进行签名并将其转发到 MediaStore 进行签名匹配，从而无需使用和轮换密钥。这可确保在提供媒体内容之前自动验证请求，从而使媒体内容的交付变得更加 CloudFront 简单 MediaStore 和安全。

允许 CloudFront 访问您的容器（控制台）

1. 打开 MediaStore 控制台，网址为https://console.aws.amazon.com/mediastore/。
2. 在 Containers (容器) 页面上，选择容器名称。

   此时将显示容器详细信息页面。
3. 在 “容器政策” 部分，附上授予 Amazon 读取权限或更高权限的策略 CloudFront。

   Example

   以下示例策略类似于通过 HTTPS 进行公共读取访问的示例策略，它符合这些要求，因为它允许任何通过 HTTPS 提交域访问请求的用户执行 `GetObject` 和 `DescribeObject` 命令。此外，以下示例策略可以更好地保护您的工作流程，因为它仅在请求通过 HTTPS 连接发出且包含正确的 Referer 标头时才允许 CloudFront 访问 MediaStore 对象。
4. 在 Container CORS policy (容器 CORS 策略) 部分中，分配一个允许适当的访问级别的策略。

> **Note**
>
> 只有在您希望提供对基于浏览器的播放器的访问权限时，才需要 CORS 策略。

5. 记下以下详细信息：
   - 分配到您的容器的数据终端节点。您可以在 容器 页面的 信息 部分中找到此信息。在中 CloudFront，数据端点被称为源域名。
   - 存储对象的容器中的文件夹结构。在中 CloudFront，这被称为起点路径。请注意，此设置为可选。有关源路径的更多信息，请参阅《Amazon CloudFront 开发者指南》。
6. 在中 CloudFront，创建一个配置为提供来自 AWS Element MediaStore al 的内容的分发。您将需要在上一步中收集的信息。

将策略附加到 MediaStore 容器后，您必须配置为仅 CloudFront 对源请求使用 HTTPS 连接，并添加具有正确机密值的自定义标头。

配置 CloudFront 为通过 HTTPS 连接访问您的容器，并使用 Referer 标头的密钥值（控制台）

1. 打开控制 CloudFront 台。
2. 在起源页面上，选择您的 MediaStore起源。
3. 选择编辑。
4. 对于协议，仅选择 HTTP。
5. 在添加自定义标题部分中，选择添加标题。
6. 在“名称”中，选择 Referer。对于该值，请使用您在容器策略中使用的相同*<secretValue>*字符串。
7. 选择“保存”，然后让更改进行部署。

AWS Elemental 与 HT MediaStore TP 缓存的交互

AWS Elemental MediaStore 存储对象，以便像亚马逊这样的内容交付网络 (CDNs) 可以正确、高效地缓存这些对象。 CloudFront当最终用户或 CDN 从中检索对象时 MediaStore，该服务会返回影响该对象缓存行为的 HTTP 标头。（HTTP 1.1 缓存行为的标准可在RFC2616 第 13 节中找到。）这些标头包括：

- **ETag**（不可自定义）– 实体标签标头是 MediaStore 发送的响应的唯一标识符。符合标准 CDNs 的 Web 浏览器使用此标签作为缓存对象的密钥。 MediaStore 上传时会自动ETag为每个对象生成。您可以查看对象的详细信息以确定其 ETag 值。
- **Last-Modified**（不可自定义）— 此标题的值表示修改对象的日期和时间。 MediaStore 上传对象时会自动生成此值。
- **Cache-Control**（可自定义）– 此标头的值控制 CDN 应在对象缓存多久后检查该对象是否经过修改。使用 CLI 或 AP I 将对象上传到 MediaStore 容器时，可以将此标头设置为任何值。HTTP/1.1 文档中介绍了完整的有效值集。如果您在上传对象时未设置此值，则在检索对象时 MediaStore 不会返回此标头。

  Cache-Control 标头的常见用例是指定缓存对象的持续时间。例如，假设您有一个经常被编码器覆盖的视频清单文件。您可以将 `max-age` 设置为 10 以指示对象应仅缓存 10 秒。或者假设您存储了一个永远不会被覆盖的视频段。您可以将此对象的 `max-age` 设置为 31536000，以缓存大约 1 年。

有条件请求

有条件地请求到 MediaStore

MediaStore 对条件请求（使用请求标头，例如`If-Modified-Since`和`If-None-Match`，如中所述 RFC7232）和无条件请求的响应方式相同。这意味着，当 MediaStore 收到有效`GetObject`请求时，即使客户端已经拥有该对象，服务也始终会返回该对象。

有条件地请求到 CDNs

CDNs 代表提供内容的人 MediaStore 可以通过返回来处理有条件的请求`304 Not Modified`，如RFC7232 第 4.1 节所述。这指示不需要传输完整的对象内容，因为请求者已有一个与有条件请求匹配的对象。

CDNs （以及其他符合 HTTP/1.1 的缓存）根据源服务器转发的`ETag`和`Cache-Control`标头做出这些决定。要控制向 MediaStore 源服务器 CDNs 查询重复检索对象更新的频率，请在将这些对象上传到时为这些对象设置`Cache-Control`标头 MediaStore。

将此服务与 AWS SDK 配合使用

AWS 软件开发套件 (SDKs) 可用于许多流行的编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

SDK 文档	代码示例
适用于 C++ 的 AWS SDK	适用于 C++ 的 AWS SDK 代码示例
AWS CLI	AWS CLI 代码示例
适用于 Go 的 AWS SDK	适用于 Go 的 AWS SDK 代码示例
适用于 Java 的 AWS SDK	适用于 Java 的 AWS SDK 代码示例
适用于 JavaScript 的 AWS SDK	适用于 JavaScript 的 AWS SDK 代码示例
适用于 Kotlin 的 AWS SDK	适用于 Kotlin 的 AWS SDK 代码示例
适用于 .NET 的 AWS SDK	适用于 .NET 的 AWS SDK 代码示例
适用于 PHP 的 AWS SDK	适用于 PHP 的 AWS SDK 代码示例
AWS Tools for PowerShell	AWS Tools for PowerShell 代码示例
适用于 Python (Boto3) 的 AWS SDK	适用于 Python (Boto3) 的 AWS SDK 代码示例
适用于 Ruby 的 AWS SDK	适用于 Ruby 的 AWS SDK 代码示例
适用于 Rust 的 AWS SDK	适用于 Rust 的 AWS SDK 代码示例
适用于 SAP ABAP 的 AWS SDK	适用于 SAP ABAP 的 AWS SDK 代码示例
适用于 Swift 的 AWS SDK	适用于 Swift 的 AWS SDK 代码示例

有关特定于此服务的示例，请参阅使用的代码示 MediaStore 例 AWS SDKs。

示例可用性

找不到所需的内容？ 通过使用此页面底部的提供反馈链接请求代码示例。

使用的代码示 MediaStore 例 AWS SDKs

以下代码示例说明如何 MediaStore 使用 AWS 软件开发套件 (SDK)。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

- 使用的基本示 MediaStore 例 AWS SDKs
  - MediaStore 使用的操作 AWS SDKs
    - CreateContainer与 AWS SDK 或 CLI 配合使用
    - DeleteContainer与 AWS SDK 或 CLI 配合使用
    - 与 AWS SDK DeleteObject 配合使用
    - DescribeContainer与 AWS SDK 或 CLI 配合使用
    - GetObject与 AWS SDK 或 CLI 配合使用
    - ListContainers与 AWS SDK 或 CLI 配合使用
    - PutObject与 AWS SDK 或 CLI 配合使用

使用的基本示 MediaStore 例 AWS SDKs

以下代码示例说明如何使用 with 的基础 AWS Elemental MediaStore 知识 AWS SDKs。

示例

- MediaStore 使用的操作 AWS SDKs
  - CreateContainer与 AWS SDK 或 CLI 配合使用
  - DeleteContainer与 AWS SDK 或 CLI 配合使用
  - 与 AWS SDK DeleteObject 配合使用
  - DescribeContainer与 AWS SDK 或 CLI 配合使用
  - GetObject与 AWS SDK 或 CLI 配合使用
  - ListContainers与 AWS SDK 或 CLI 配合使用
  - PutObject与 AWS SDK 或 CLI 配合使用

MediaStore 使用的操作 AWS SDKs

以下代码示例演示了如何使用执行单个 MediaStore操作 AWS SDKs。每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

以下示例仅包括最常用的操作。有关完整列表，请参阅 AWS Elemental MediaStore API 参考。

示例

- CreateContainer与 AWS SDK 或 CLI 配合使用
- DeleteContainer与 AWS SDK 或 CLI 配合使用
- 与 AWS SDK DeleteObject 配合使用
- DescribeContainer与 AWS SDK 或 CLI 配合使用
- GetObject与 AWS SDK 或 CLI 配合使用
- ListContainers与 AWS SDK 或 CLI 配合使用
- PutObject与 AWS SDK 或 CLI 配合使用

`CreateContainer`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateContainer`。

CLI

AWS CLI

创建容器

以下 `create-container` 示例创建一个新的空容器。

```
aws mediastore create-container --container-name ExampleContainer
```

输出：

```
{
    "Container": {
        "AccessLoggingEnabled": false,
        "CreationTime": 1563557265,
        "Name": "ExampleContainer",
        "Status": "CREATING",
```

```
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer"
    }
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的创建容器。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateContainer中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>
```

```
                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }


    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
String containerName) {
        try {
            CreateContainerRequest containerRequest =
CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
```

```
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateContainer中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteContainer`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteContainer`。

CLI

AWS CLI

删除容器

以下 `delete-container` 示例删除指定容器。您只能在容器中没有对象时才能将其删除。

```
aws mediastore delete-container \
    --container-name=ExampleLiveDemo
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的删除容器。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteContainer中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.model.CreateContainerRequest;
import software.amazon.awssdk.services.mediastore.model.CreateContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateContainer {
    public static long sleepTime = 10;

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>

                Where:
                   containerName - The name of the container to create.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        createMediaContainer(mediaStoreClient, containerName);
        mediaStoreClient.close();
    }
```

```
    public static void createMediaContainer(MediaStoreClient mediaStoreClient,
String containerName) {
        try {
            CreateContainerRequest containerRequest =
CreateContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            CreateContainerResponse containerResponse =
mediaStoreClient.createContainer(containerRequest);
            String status = containerResponse.container().status().toString();
            while (!status.equalsIgnoreCase("Active")) {
                status = DescribeContainer.checkContainer(mediaStoreClient,
containerName);
                System.out.println("Status - " + status);
                Thread.sleep(sleepTime * 1000);
            }

            System.out.println("The container ARN value is " +
containerResponse.container().arn());
            System.out.println("Finished ");

        } catch (MediaStoreException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteContainer中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

与 AWS SDK **DeleteObject** 配合使用

以下代码示例演示了如何使用 DeleteObject。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.DeleteObjectRequest;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:    <completePath> <containerName>

                Where:
                   completePath - The path (including the container) of the item
 to delete.
                   containerName - The name of the container.
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));

        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        deleteMediaObject(mediaStoreData, completePath);
        mediaStoreData.close();
    }

    public static void deleteMediaObject(MediaStoreDataClient mediaStoreData,
String completePath) {
        try {
            DeleteObjectRequest deleteObjectRequest =
DeleteObjectRequest.builder()
                    .path(completePath)
                    .build();

            mediaStoreData.deleteObject(deleteObjectRequest);

        } catch (MediaStoreDataException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
DescribeContainerRequest.builder()
```

```
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        mediaStoreClient.close();
        return response.container().endpoint();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteObject中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeContainer`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeContainer`。

CLI

AWS CLI

查看容器的详细信息

以下 `describe-container` 示例显示指定容器的详细信息。

```
aws mediastore describe-container \
    --container-name ExampleContainer
```

输出：

```
{
    "Container": {
        "CreationTime": 1563558086,
        "AccessLoggingEnabled": false,
        "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
        "Status": "ACTIVE",
        "Name": "ExampleContainer",
```

```
        "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com"
    }
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的查看容器的详细信息。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeContainer中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DescribeContainer {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <containerName>
```

```
                Where:
                   containerName - The name of the container to describe.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String containerName = args[0];
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        System.out.println("Status is " + checkContainer(mediaStoreClient,
 containerName));
        mediaStoreClient.close();
    }

    public static String checkContainer(MediaStoreClient mediaStoreClient, String
 containerName) {
        try {
            DescribeContainerRequest describeContainerRequest =
 DescribeContainerRequest.builder()
                    .containerName(containerName)
                    .build();

            DescribeContainerResponse containerResponse =
 mediaStoreClient.describeContainer(describeContainerRequest);
            System.out.println("The container name is " +
 containerResponse.container().name());
            System.out.println("The container ARN is " +
 containerResponse.container().arn());
            return containerResponse.container().status().toString();

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeContainer中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`GetObject`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetObject`。

CLI

AWS CLI

下载对象

以下 `get-object` 示例将对象下载到指定端点。

```
aws mediastore-data get-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --path=/folder_name/README.md README.md
```

输出：

```
{
    "ContentLength": "2307346",
    "ContentType": "image/jpeg",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3",
    "StatusCode": 200
}
```

下载对象的一部分

以下 `get-object` 示例将对象的一部分下载到指定端点。

```
aws mediastore-data get-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --path /folder_name/README.md \
    --range="bytes=0-100" README2.md
```

输出：

```
{
    "StatusCode": 206,
    "ContentRange": "bytes 0-100/2307346",
    "ContentLength": "101",
    "LastModified": "Fri, 19 Jul 2019 21:32:20 GMT",
    "ContentType": "image/jpeg",
    "ETag": "2aa333bbcc8d8d22d777e999c88d4aa9eeeeee4dd89ff7f555555555555da6d3"
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的下载对象。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考GetObject中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:    <completePath> <containerName> <savePath>

                Where:
                   completePath - The path of the object in the container (for
 example, Videos5/sampleVideo.mp4).
                   containerName - The name of the container.
                   savePath - The path on the local drive where the file is
 saved, including the file name (for example, C:/AWS/myvid.mp4).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        String savePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        getMediaObject(mediaStoreData, completePath, savePath);
        mediaStoreData.close();
    }
```

```
    public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
 completePath, String savePath) {

        try {
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .path(completePath)
                    .build();

            // Write out the data to a file.
            ResponseInputStream<GetObjectResponse> data =
 mediaStoreData.getObject(objectRequest);
            byte[] buffer = new byte[data.available()];
            data.read(buffer);

            File targetFile = new File(savePath);
            OutputStream outStream = new FileOutputStream(targetFile);
            outStream.write(buffer);
            System.out.println("The data was written to " + savePath);

        } catch (MediaStoreDataException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetObject](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListContainers`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListContainers`。

CLI

AWS CLI

查看容器列表

以下 `list-containers` 示例显示与您的账户关联的所有容器的列表。

```
aws mediastore list-containers
```

输出：

```
{
    "Containers": [
        {
            "CreationTime": 1505317931,
            "Endpoint": "https://aaabbbcccdddee.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleLiveDemo",
            "AccessLoggingEnabled": false,
            "Name": "ExampleLiveDemo"
        },
        {
            "CreationTime": 1506528818,
            "Endpoint": "https://fffggghhhiiijj.data.mediastore.us-
west-2.amazonaws.com",
            "Status": "ACTIVE",
            "ARN": "arn:aws:mediastore:us-west-2:111122223333:container/
ExampleContainer",
            "AccessLoggingEnabled": false,
            "Name": "ExampleContainer"
        }
    ]
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的查看容器列表。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ListContainers中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.Container;
import software.amazon.awssdk.services.mediastore.model.ListContainersResponse;
import software.amazon.awssdk.services.mediastore.model.MediaStoreException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListContainers {

    public static void main(String[] args) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        listAllContainers(mediaStoreClient);
        mediaStoreClient.close();
```

```
    }

    public static void listAllContainers(MediaStoreClient mediaStoreClient) {
        try {
            ListContainersResponse containersResponse =
 mediaStoreClient.listContainers();
            List<Container> containers = containersResponse.containers();
            for (Container container : containers) {
                System.out.println("Container name is " + container.name());
            }

        } catch (MediaStoreException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListContainers中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`PutObject`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `PutObject`。

CLI

AWS CLI

上传对象

以下 `put-object` 示例将对象上传到指定容器。您可以指定在容器中保存对象的文件夹路径。如果该文件夹已经存在， AWS Elemental 会将该对象 MediaStore 存储在文件夹中。如果该文件夹不存在，则该服务将创建一个文件夹，然后将对象存储在其中。

```
aws mediastore-data put-object \
    --endpoint https://aaabbbcccdddee.data.mediastore.us-west-2.amazonaws.com \
    --body README.md \
    --path /folder_name/README.md \
    --cache-control "max-age=6, public" \
```

```
    --content-type binary/octet-stream
```

输出：

```
{
    "ContentSHA256":
 "74b5fdb517f423ed750ef214c44adfe2be36e37d861eafe9c842cbe1bf387a9d",
    "StorageClass": "TEMPORAL",
    "ETag": "af3e4731af032167a106015d1f2fe934e68b32ed1aa297a9e325f5c64979277b"
}
```

有关更多信息，请参阅 AWS Elemental MediaStore 用户指南中的上传对象。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考PutObject中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectRequest;
import
 software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import software.amazon.awssdk.services.mediastoredata.model.PutObjectResponse;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import
 software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import java.io.File;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PutObject {
    public static void main(String[] args) throws URISyntaxException {
        final String USAGE = """

                To run this example, supply the name of a container, a file
 location to use, and path in the container\s

                Ex: <containerName> <filePath> <completePath>
                """;

        if (args.length < 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String containerName = args[0];
        String filePath = args[1];
        String completePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        putMediaObject(mediaStoreData, filePath, completePath);
        mediaStoreData.close();
    }

    public static void putMediaObject(MediaStoreDataClient mediaStoreData, String
 filePath, String completePath) {
        try {
            File myFile = new File(filePath);
            RequestBody requestBody = RequestBody.fromFile(myFile);

            PutObjectRequest objectRequest = PutObjectRequest.builder()
```

```
                    .path(completePath)
                    .contentType("video/mp4")
                    .build();

            PutObjectResponse response = mediaStoreData.putObject(objectRequest,
 requestBody);
            System.out.println("The saved object is " +
 response.storageClass().toString());

        } catch (MediaStoreDataException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getEndpoint(String containerName) {

        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutObject中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅将此服务与 AWS SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

AWS Elemental 中的配额 MediaStore

服务配额控制台提供有关 AWS Elemental MediaStore 配额的信息。除了查看默认配额外，还可以使用服务配额控制台为可调整的配额请求提高配额。

下表描述了 AWS Element MediaStore al 中的配额（以前称为限制）。限额是您的 Amazon Web Services 账户使用的服务资源或操作的最大数量。

> **Note**
>
> 要为您的账户中的单个容器分配配额，请联系 Amazon Web Services Support 或您的账户经理。此选项可以帮助您在容器之间划分账户级别的限制，以防止一个容器耗尽您的全部配额。

资源或操作	默认配额	评论
容器	100	您可在此账户中创建的容器的最大数量。
文件夹级别	10	您可在容器中创建的文件夹级别的最大数量。您可以创建任意数量的文件夹，只要它们在一个容器中嵌套不超过 10 个级别。
文件夹	无限制	您可以创建任意数量的文件夹，只要它们在一个容器中嵌套不超过 10 个级别。
对象大小	25MB	单个对象的最大文件大小。
对象	无限制	您可以将任意数量的对象上传到账户中的文件夹或容器。
DeleteObject API 请求的速率	100	每秒可发出的操作请求的最大数量。额外的请求将被阻止。 您可以请求提高配额。
DescribeObject API 请求的速率	1000	每秒可发出的操作请求的最大数量。额外的请求将被阻止。 您可以请求提高配额。

资源或操作	默认配额	评论
标准上传可用性的 GetObjectAPI 请求率	1000	每秒可发出的操作请求的最大数量。额外的请求将被阻止。 您可以请求提高配额。
流式上传可用性的 GetObjectAPI 请求率	25	每秒可发出的操作请求的最大数量。额外的请求将被阻止。 您可以请求提高配额。
ListItems API 请求的速率	5	每秒可发出的操作请求的最大数量。额外的请求将被阻止。 您可以请求提高配额。
分块传输编码 PutObjectAPI 请求的速率（也称为流式上传可用性）	10	每秒可发出的操作请求的最大数量。额外的请求将被阻止。 您可以请求提高配额。在请求中，指定请求的 TPS 和平均对象大小。
标准上传可用性的 PutObjectAPI 请求率	100	每秒可发出的操作请求的最大数量。额外的请求将被阻止。 您可以请求提高配额。在请求中，指定请求的 TPS 和平均对象大小。
指标策略中的规则	10	指标策略中可包含的最大规则数。
对象生命周期策略中的规则	10	您可以在对象生命周期策略中包含的最大规则数量。

AWS Elemental 相关信息 MediaStore

下表列出了您在使用 AWS Element MediaStore al 时会发现有用的相关资源。

- 课程和研讨会 — 指向基于角色的课程和专业课程的链接，以及自定进度的实验室，可帮助您提高 AWS 技能并获得实践经验。
- AWS 开发者中心 — 浏览教程、下载工具并了解 AWS 开发者活动。
- AWS 开发者工具-指向开发者工具 SDKs、IDE 工具包以及用于开发和管理 AWS 应用程序的命令行工具的链接。
- 入门资源中心 — 了解如何设置你的 AWS 账户、加入 AWS 社区和启动你的第一个应用程序。
- 动手教 step-by-step程 — 按照教程启动您的第一个应用程序 AWS。
- AWS 白皮书 — 指向技术 AWS 白皮书完整列表的链接，这些白皮书涵盖架构、安全和经济学等主题，由 AWS 解决方案架构师或其他技术专家撰写。
- AWS 支持 中心 — 创建和管理 AWS 支持 案例的中心。还包括指向其他有用资源的链接，例如论坛 FAQs、技术、服务运行状况和 AWS Trusted Advisor。
- 支持— 提供有关 支持快速响应支持渠道信息的主要网页 one-on-one，该渠道可帮助您在云中构建和运行应用程序。
- 联系我们 – 用于查询有关 AWS 账单、账户、事件、滥用和其他问题的中央联系点。
- AWS 网站条款 — 有关我们的版权和商标、您的帐户、许可证和网站访问权限以及其他主题的详细信息。

用户指南文档历史记录

下表描述了此版本的 AWS Element MediaStore al 的文档。要获得本文档的更新通知，您可以订阅 RSS 源。

变更	说明	日期
终止支持通知	终止支持通知：2025 年 11 月 13 日，我 AWS 将停止对 AWS MediaStore Elemental 的支持。2025 年 11 月 13 日之后，您将无法再访问 MediaStore 控制台或 MediaStore 资源。有关更多信息，请访问此博客文章。	2024 年 11 月 12 日
源站访问控制 (OAC) 改进	添加了有关如何在 AWS Elemental 中使用 OA MediaStore C 的信息。	2023 年 4 月 17 日
配额更新	更正了的配额值和描述`Rules in a Metric Policy`。	2022 年 10 月 25 日
ExpiresAt field	访问日志现在包含一个`ExpiresAt` 字段，该字段根据容器生命周期策略中的临时数据规则指示对象的过期日期和时间。	2020 年 7 月 16 日
生命周期转换规则	现在，您可以向对象生命周期策略中添加生命周期转换规则，将对象设置为在达到一定期限后移动到不经常访问 (IA) 存储类中。	2020 年 4 月 20 日
清空容器	您现在可以一次删除容器中的所有对象。	2020 年 4 月 7 日

对 Amazon CloudWatch 指标的支持	您可以设置指标策略来决定将哪些指标 MediaStore 发送到 CloudWatch哪里。	2020 年 3 月 30 日
删除对象规则中的通配符	在对象生命周期策略中，您现在可以在删除对象规则中使用通配符。这样一来，您就可以根据文件名或扩展名，来指定您希望服务在一定天数后删除的文件。	2019 年 12 月 20 日
对象生命周期策略	现在，您可以在对象生命周期策略中添加一条规则，该规则以年龄（以秒为单位）指示到期时间。	2019 年 9 月 13 日
CloudFormation 支持	现在，您可以使用 CloudFormation 模板自动创建容器。该 CloudFormation 模板管理五个 API 操作的数据：创建容器、设置访问日志记录、更新默认容器策略、添加跨源资源共享 (CORS) 策略以及添加对象生命周期策略。	2019 年 5 月 17 日
流式上传可用性的配额	对于具有流式上传可用性（对象的分块传输）的对象，`PutObject` 操作不得超过 10TPS，而 `GetObject` 操作不得超过 25TPS。	2019 年 4 月 8 日
对象分块传输	添加对对象分块传输的支持。此功能可让您指定可在对象上传完成之前下载此对象。	2019 年 4 月 5 日

访问日志记录	AWS Elemental MediaStore 现在支持访问日志，它可以提供对容器中对象发出的请求的详细记录。	2019 年 2 月 25 日
对象生命周期策略	添加对对象生命周期测量的支持，这些策略管理当前容器内对象的过期时间。	2018 年 12 月 12 日
增大的对象大小配额	现在，对象大小的配额为 25MB。	2018 年 10 月 10 日
增大的对象大小配额	现在，对象大小的配额为 20MB。	2018 年 9 月 6 日
AWS CloudTrail 整合	CloudTrail 集成内容已更新，以适应 CloudTrail 服务的最新更改。	2018 年 7 月 12 日
CDN 协作	添加了有关如何将 AWS Elemental MediaStore 与内容分发网络 (CDN)（例如亚马逊）配合使用的信息。CloudFront	2018 年 4 月 14 日
CORS 配置	AWS Elemental MediaStore 现在支持跨源资源共享 (CORS)，允许在一个域中加载的客户端 Web 应用程序与另一个域中的资源进行交互。	2018 年 2 月 7 日
新服务和指南	这是视频制作和存储服务 AWS Elemental 和 AWS MediaStore Elemental 用户指南的初始版本。 MediaStore	2017 年 11 月 27 日

Note

- AWS 媒体服务不是为应用程序或需要故障安全性能的情况而设计或使用的，例如生命安全操作、导航或通信系统、空中交通管制或生命支持机器，在这些机器中，服务的不可用、中断或故障可能导致死亡、人身伤害、财产损失或环境破坏。

AWS 词汇表

有关最新 AWS 术语，请参阅《AWS 词汇表 参考资料》中的AWS 词汇表。