《用户指南》

AWS元素 MediaStore

AWS元素 MediaStore: 《用户指南》

Table of Contents

什么是 MediaStore？ 1
概念和术语 1
相关服务 2
访问 MediaStore 3
定价 4
区域和终端节点 4
设置 AWS Elemental MediaStore 5
注册获取 AWS 账户 5
创建具有管理访问权限的用户 5
开始使用 7
第 1 步：访问 AWS Elemental MediaStore 7
步骤 2：创建容器 7
步骤 3：上传对象 8
步骤 4：访问对象 8
容器 9
容器命名规则 9
创建容器 9
查看容器详细信息 10
查看容器列表 11
删除容器 13
策略 14
容器策略 14
查看容器策略 14
编辑容器策略 16
示例容器策略 17
CORS 策略 23
使用案例方案 23
添加 CORS 策略 24
查看 CORS 策略 25
编辑 CORS 策略 26
删除 CORS 策略 27
故障排除 28
示例 CORS 策略 28
对象生命周期策略 30

对象生命周期策略的组成 30
添加对象生命周期策略 36
查看对象生命周期策略 37
编辑对象生命周期策略 38
删除对象生命周期策略 40
示例对象生命周期策略 40
指标策略 44
添加指标策略 45
查看指标策略 45
编辑指标策略 46
示例指标策略 46
文件夹 50
文件夹命名规则 50
创建文件夹 51
删除文件夹 51
对象 52
上传对象 52
查看列表 54
查看对象详细信息 56
下载对象 57
删除对象 58
删除一个对象 58
清空容器 59
安全性 61
数据保护 61
数据加密 62
Identity and Access Management 62
受众 63
使用身份进行身份验证 63
使用策略管理访问 66
AWSElemental 是如何使用 MediaStore 的 IAM 68
基于身份的策略示例 74
故障排除 76
日记账记录和监控 78
亚马逊 CloudWatch 警报 78
AWS CloudTrail 日志 78

AWS Trusted Advisor 78
合规性验证 78
弹性 79
基础架构安全性 80
防止跨服务混淆座席 80
监控和标记 82
使用 CloudTrail 记录 API 调用 83
CloudTrail 中的 MediaStore 信息 83
示例：日志文件条目 84
使用 CloudWatch 进行监控 85
CloudWatch Logs 86
CloudWatch Events 94
CloudWatch metrics（CloudWatch 指标） 98
Tagging 102
AWS Elemental MediaStore 中支持的资源 103
标签命名和使用约定 103
管理标签 104
使用 CDN 105
允许 CloudFront 访问您的容器 105
使用来源访问控制 (OAC) 106
使用共享密钥 106
MediaStore 与 HTTP 缓存的互动 108
有条件请求 109
与... 合作 AWS SDKs 110
代码示例 111
基础知识 111
操作 112
配额 133
相关信息 135
文档历史记录 136
AWS 术语表 139
..... cxl

什么是 AWS Elemental MediaStore？

AWS Elemental MediaStore 是一项视频创作和存储服务，可提供实时创作所需的高性能和即时一致性。使用 MediaStore，可以在容器中将视频资产作为对象进行管理，以构建可靠、基于云的媒体工作流。

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。

当需要强一致性、低延迟读取和写入以及处理大量并发请求的能力时，MediaStore 是存储零碎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 MediaStore 的概念和术语
- 相关服务
- 访问 AWS Elemental MediaStore
- AWS Elemental MediaStore 的定价
- AWS Elemental MediaStore 区域和终端节点

AWS Elemental MediaStore 的概念和术语

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。

端点

进入 MediaStore 服务的入口点，指定为 HTTPS 根 URL。

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。

项目

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 被视为源服务，因为它是媒体内容传送的分配点。

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。

部分

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 MediaStore 中，这是资源及其 ARN 格式：

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

相关服务

- Amazon CloudFront 是一项全球内容传送网络 (CDN) 服务，可将数据和视频安全地传输给您的观看者。使用 CloudFront 以最佳的性能分发内容。有关更多信息，请参阅 Amazon CloudFront 开发人员指南。
- AWS CloudFormation 是一项服务，可帮助您建模和设置 AWS 资源。您创建一个描述您所需的所有 AWS 资源（如 MediaStore 容器）的模板，并且 AWS CloudFormation 将负责为您设置和配置这些

资源。您无需单独创建和配置 AWS 资源并了解 what; AWS CloudFormation 句柄处理所有这些工作时所依赖的内容。有关更多信息，请参阅 AWS CloudFormation《用户指南》。

- AWS CloudTrail 是一种服务，可用于监控对账户的 CloudTrail API 的调用，包括通过 Amazon Web Services 管理控制台、AWS CLI 和其他服务发出的调用。有关更多信息，请参阅 AWS CloudTrail《用户指南》。
- Amazon CloudWatch 是一项针对AWS云资源以及在AWS 上运行的应用程序的监控服务。使用 CloudWatch Events 可跟踪 MediaStore 中容器和对象状态的更改。有关更多信息，请参阅 Amazon CloudWatch 文档。
- AWS Identity and Access Management (IAM) 是一种 Web 服务，可帮助您安全地控制用户对 AWS 资源的访问权限。通过 IAM 可以控制哪些人可以使用您的 AWS 资源（身份验证）以及用户可以使用的资源和采用的方式（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

您可以使用下面任何方式访问 MediaStore：

- Amazon Web Services 管理控制台 – 本指南中的过程阐述如何使用 Amazon Web Services 管理控制台执行 MediaStore 任务。使用控制台访问 MediaStore：

```
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。要使用 CLI 端点访问 MediaStore：

```
aws mediastore
```

- MediaStore API – 如果要使用开发工具包不可用的编程语言，请参阅 AWS Elemental MediaStore API 参考 以了解有关 API 操作及如何发出 API 请求的信息。要使用 REST API 终端节点访问 MediaStore，请执行以下操作：

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

- AWS 开发工具包 – 如果要使用 AWS 提供的开发工具包可用的编程语言，则可以使用开发工具包访问 MediaStore。开发工具包可简化身份验证，与开发环境轻松集成，并有助于轻松访问 MediaStore 命令。有关更多信息，请参阅用于 Amazon Web Services 的工具。

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

AWS Elemental MediaStore 的定价

与其他 AWS 产品一样，在使用 MediaStore 时，您无需签订合同或承诺最低使用量。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 MediaStore 区域和终端节点

为减少应用程序中的数据延迟，MediaStore 提供了区域终端节点来发出请求。

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

要查看提供 MediaStore 的 Amazon Web Services 区域的完整列表，请参阅 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访问 AWS Elemental所需的步骤。 MediaStore有关身份和访问管理的背景信息和其他信息 MediaStore，请参阅AWSElemental 的 Identity and Access 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
- 创建具有管理访问权限的用户

注册获取 AWS 账户

如果你没有 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. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当你注册时 AWS 账户，一个 AWS 账户根用户已创建。root 用户可以访问所有内容 AWS 服务 以及账户中的资源。作为安全最佳实践，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

在你注册之后 AWS 账户，保护你的 AWS 账户根用户，启用 AWS IAM Identity Center，然后创建一个管理用户，这样你就不会使用 root 用户来执行日常任务。

保护你的 AWS 账户根用户

1. 登录 。AWS Management Console以账户所有者的身份选择 Root 用户并输入你的 AWS 账户 电子邮件地址。在下一页上，输入您的密码。

   有关使用 root 用户登录的帮助，请参阅中以 root 用户身份登录 AWS 登录 用户指南。
2. 为您的 root 用户开启多重身份验证 (MFA)。

   有关说明，请参阅为您的MFA设备启用虚拟设备 AWS 账户 用户指南中的 root IAM 用户（控制台）。

创建具有管理访问权限的用户

1. 启用IAM身份中心。

   有关说明，请参阅启用 AWS IAM Identity Center中的 AWS IAM Identity Center 用户指南。
2. 在 IAM Identity Center 中，向用户授予管理访问权限。

   有关使用教程 IAM Identity Center 目录 作为您的身份来源，请参阅使用默认设置配置用户访问权限 IAM Identity Center 目录中的 AWS IAM Identity Center 用户指南。

以具有管理访问权限的用户身份登录

- 要使用您的 Ident IAM ity Center 用户登录URL，请使用您在创建 Ident IAM ity Center 用户时发送到您的电子邮件地址的登录信息。

  有关使用IAM身份中心用户登录的帮助，请参阅登录 AWS 访问中的门户 AWS 登录 用户指南。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个遵循应用最低权限权限的最佳实践的权限集。

   有关说明，请参阅中的创建权限集 AWS IAM Identity Center 用户指南。
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，请参阅中的添加群组 AWS IAM Identity Center 用户指南。

AWS Elemental MediaStore 入门

本入门教程为您演示如何使用 AWS Elemental MediaStore 来创建容器并上传对象。

主题

- 第 1 步：访问 AWS Elemental MediaStore
- 步骤 2：创建容器
- 步骤 3：上传对象
- 步骤 4：访问对象

第 1 步：访问 AWS Elemental MediaStore

设置 Amazon Web Services 账户并创建用户和角色之后，登录 AWS Elemental MediaStore 控制台。

访问 AWS Elemental MediaStore

- 请登录并通过以下网址AWS Management Console打开 MediaStore 控制台：https://console.aws.amazon.com/glue/。

> **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

您可以使用 AWS Elemental MediaStore 中的容器来存储文件夹和对象。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。如果文件夹已存在，则 AWS Elemental MediaStore 会将对象存储在文件夹中。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`。如果所指定路径中的任何文件夹不存在，则 AWS Elemental MediaStore 将自动创建这些文件夹。
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 MediaStore 中的容器

在 MediaStore 中使用容器来存储文件夹和对象。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
- 创建容器
- 查看有关容器的详细信息
- 查看容器列表
- 删除容器

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**
>
> 还可以使用 AWS CloudFormation 模板自动创建容器。该 AWS 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 AWS CLI 中，使用 `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 AWS CLI 中，使用 `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 AWS CLI 中，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://fffgggghhhiiijj.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

您只能在容器没有对象时将其删除。

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 Delete (删除)。

删除容器 (AWS CLI)

- 在 AWS CLI 中，使用 `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 MediaStore 中的政策

您可以将以下一个或多个策略应用于 AWS Elemental MediaStore 容器：

- 容器策略-设置对容器内所有文件夹和对象的访问权限。MediaStore 设置了默认策略，允许用户在容器上执行所有 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) 策略可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交互。MediaStore 不设置默认 CORS 策略。
- 指标策略 - 允许 MediaStore 向 Amazon CloudWatch 发送指标。MediaStore 不设置默认指标策略。
- 对象生命周期策略 - 控制对象在 MediaStore 容器中保留多长时间。MediaStore 不设置默认对象生命周期策略。

AWS Elemental MediaStore 中的容器策略

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。自动附加到所有新容器的默认策略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 AWS CLI 中，使用 `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：

```
{
  "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 AWS CLI 中，使用 `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
- 示例容器策略：通过 HTTPS 的公共读取访问权限
- 示例容器策略：通过 HTTP 或 HTTPS 的公共读取访问权限
- 示例容器策略：已启用 HTTP 的跨账户读取访问权限
- 示例容器策略：通过 HTTPS 的跨账户读取访问权限
- 示例容器策略：对角色的跨账户读取访问权限
- 示例容器策略：对角色的跨账户完全访问权限
- 示例容器策略：限制对特定 IP 地址的访问

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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MediaStoreFullAccess",
      "Action": [ "mediastore:*" ],
      "Principal":{
          "AWS" : "arn:aws:iam::<aws_account_number>:root"},
      "Effect": "Allow",
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>: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。它通过安全的 SSL/TLS 连接，对任何人都允许读取访问权限，包括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：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadOverHttps",
      "Effect": "Allow",
      "Action": ["mediastore:GetObject", "mediastore:DescribeObject"],
      "Principal": "*",
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>: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）：

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

```
      "Effect": "Allow",
      "Action": ["mediastore:GetObject", "mediastore:DescribeObject"],
      "Principal": "*",
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>:container/<container
 name>/*",
      "Condition": {
        "Bool": { "aws:SecureTransport": ["true", "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。使用 SSL/TLS 证书无需在服务器上托管对象：

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

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：

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CrossAccountReadOverHttps",
      "Effect": "Allow",
      "Action": ["mediastore:GetObject", "mediastore:DescribeObject"],
      "Principal":{
        "AWS": "arn:aws:iam::<other acct number>:root"},
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>: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）。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CrossAccountRolePostOverHttps",
      "Effect": "Allow",
      "Action": "mediastore:PutObject",
      "Principal":{
        "AWS": "arn:aws:iam::<other acct number>:role/<role name>"},
      "Resource": "arn:aws:mediastore:<region>:<owner acct number>:container/<container
 name>/*",
      "Condition": {
        "Bool": {
            "aws:SecureTransport": "true"
        }
      }
    },
    {
```

```
      "Sid": "CrossAccountFullAccessExceptPost",
      "Effect": "Allow",
      "NotAction": "mediastore:PutObject",
      "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：限制对特定 IP 地址的访问

此示例策略允许对在指定容器中的对象访问所有 AWS Elemental MediaStore 操作。但是，请求必须来自条件中指定的 IP 地址范围。

此语句中的条件确定允许的 Internet 协议版本 4 (IPv4) IP 地址范围 198.51.100.*，只有一个例外：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)。aws:sourceIp IPv4 值使用标准 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。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AccessBySpecificIPAddress",
            "Effect": "Allow",
            "Action": [
                "mediastore:GetObject",
                "mediastore:DescribeObject"
            ],
            "Principal": "*",
            "Resource": "arn:aws:mediastore:<region>:<owner acct number>:container/
<container name>/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "198.51.100.0/24"
                    ]
                },
                "NotIpAddress": {
```

```
                    "aws:SourceIp": "198.51.100.188/32"
                }
            }
        }
    ]
}
```

AWS Elemental MediaStore 中的跨源资源共享 (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。AWS Elemental MediaStore 中的 CORS 支持让您可以使用 MediaStore 构建丰富的客户端 Web 应用程序，同时让您可以选择性地允许跨源访问您的 资源。

> **Note**
>
> 如果您使用 Amazon CloudFront 从具有 CORS 策略的容器中分配内容，请务必使用配置 AWS Elemental MediaStore 的分配（包括编辑缓存行为来设置 CORS 的步骤）。

本部分提供 CORS 概述。副主题介绍您如何通过使用 AWS Elemental MediaStore 控制台或以编程方式使用 MediaStore REST API 和 AWS 软件开发工具包来启用 CORS。

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：

- 方案 1：假设您将在名为 LiveVideo 的 AWS Elemental MediaStore 容器中分发实时流式传输视频。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。您希望使用 JavaScript 视频播放器通过未经身份验证的 GET 和 PUT 请求访问源自此容器的视频。浏览器一般会阻止 JavaScript 允许这些请求，但您可以在容器上设置 CORS 策略以显式支持来自 www.example.com 的这些请求。

- 方案 2：假设您要通过 MediaStore 容器托管与方案 1 中相同的实时流式传输流，但想要允许来自任何源的请求。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

本节说明如何将跨源资源共享 (CORS) 配置添加到 AWS Elemental MediaStore 容器。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 AWS CLI 中，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 AWS CLI 中，使用 `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 AWS CLI 中，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 AWS CLI 中，使用 `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。

      如果缺少该标头，则 AWS Elemental MediaStore 不会将请求视为跨源请求，并且不会在响应中发送回 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
- 示例 CORS 策略：特定域的读取访问权限

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检索 AWS Elemental MediaStore 容器中的内容。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

以下策略允许来自 `https://www.example.com`的网页检索 AWS Elemental MediaStore 容器中的内容。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 MediaStore 中的对象生命周期策略

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。当对象到达您指定的最长时限时，AWS Elemental MediaStore 将删除对象。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 MediaStore 应在对象达到一定期限后将其移动到不经常访问 (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。

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

- 对象生命周期策略的组成
- 为容器添加对象生命周期策略
- 查看对象生命周期策略
- 编辑对象生命周期策略
- 删除对象生命周期策略
- 示例对象生命周期策略

对象生命周期策略的组成

对象生命周期策略用于管理对象应在 AWS Elemental MediaStore 容器中保留的时间。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。MediaStore 最长需要 20 分钟才能对容器应用新策略。

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。MediaStore 应用新策略最多需要 20 分钟，但从容器中清除对象可能需要长达 24 小时。

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。MediaStore 应用新策略最多需要 20 分钟，但从容器中清除对象可能需要长达 24 小时。

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：

- 第一个规则指示 AWS Elemental MediaStore 删除存储在 `LiveEvents/Football`和 `LiveEvents/Baseball`文件夹中超过 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。
- 第五个规则指示服务删除 Football 文件夹中与模式 index*.m3u8 匹配的超过 15 秒的对象。在这些文件放入容器后 16 秒，MediaStore 会将其删除。
- 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。您可以设置一个过期日期，在过期日期后，AWS Elemental MediaStore 会删除对象。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 AWS CLI 中，使用 `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 AWS CLI 中，使用 `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 AWS CLI 中，使用 `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 AWS CLI 中，使用 `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
- 对象生命周期策略示例：在几天内过期
- 对象生命周期策略示例：移动到不经常访问存储类
- 示例对象生命周期策略：多个规则
- 示例对象生命周期策略：空容器

示例对象生命周期策略：在几秒内过期

以下策略会指示 MediaStore 删除满足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

以下策略会指示 MediaStore 删除满足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

以下策略会指示 MediaStore 将达到 30 天的对象移动到不经常访问 (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。

`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

以下策略会指示 MediaStore 执行以下操作：

- 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会指示 MediaStore 在对象添加到容器 1 天后删除容器中的所有对象，包括文件夹和子文件夹。如果容器在应用此策略之前包含任何对象，MediaStore 会在策略生效 1 天后删除这些对象。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 MediaStore 中的指标策略

对于每个容器，您可以添加指标策略以允许 AWS Elemental MediaStore 将指标发送到 Amazon 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）。

如果对象匹配多个规则，CloudWatch 将为每个匹配规则显示一个数据点。例如，如果对象匹配名为 `rule1` 和 `rule2` 的两个规则，CloudWatch 将为这些规则显示两个数据点。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

指标策略包含指示 AWS Elemental MediaStore 向 Amazon CloudWatch 发送哪些指标的规则。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

指标策略包含指示 AWS Elemental MediaStore 向 Amazon CloudWatch 发送哪些指标的规则。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
- 示例指标策略：路径级指标
- 示例指标策略：容器级和路径级指标
- 示例指标策略：使用通配符的路径级指标
- 示例指标策略：具有重叠规则的路径级指标

示例指标策略：容器级指标

此示例策略指示 AWS Elemental MediaStore 应在容器级别向 Amazon CloudWatch 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`。

由于此策略没有规则，因此 MediaStore 不会在路径级别发送指标。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

此示例策略指示 AWS Elemental MediaStore 不应在容器级别向 Amazon CloudWatch 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 baseball/saturday文件夹的请求具有 ObjectGroupName=baseballGroup的 CloudWatch 维度。
- 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

此示例策略指示 AWS Elemental MediaStore 应在容器级别向 Amazon CloudWatch 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 baseball/saturday文件夹的请求具有 ObjectGroupName=baseballGroup的 CloudWatch 维度。

- 对 football/saturday 文件夹的请求具有 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

此示例策略指示 AWS Elemental MediaStore 应在容器级别向 Amazon CloudWatch 发送指标。而且，MediaStore 还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

此示例策略指示 AWS Elemental MediaStore 应在容器级别向 Amazon CloudWatch 发送指标。而且，MediaStore 应为两个文件夹发送指标：sports/football/saturday 和 sports/football。

对sports/football/saturday 文件夹的 MediaStore 请求的指标具有 ObjectGroupName=footballGroup1的 CloudWatch 维度。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 MediaStore 中的文件夹

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。如果文件夹已存在，则 AWS Elemental MediaStore 会将对象存储在文件夹中。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 MediaStore 会将对象存储在文件夹 premium 下的子文件夹 canada 中。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。

AWS Elemental MediaStore 会在您删除文件夹中的最后一个对象时自动删除该文件夹。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
- 创建文件夹
- 删除文件夹

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。如果文件夹已存在，则 AWS Elemental MediaStore 会将对象存储在文件夹中。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。

AWS Elemental MediaStore 会在您删除文件夹中的最后一个对象时自动删除该文件夹。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 MediaStore 中的对象

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 MediaStore 中，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 MediaStore 复制到其他位置。这不会将对象从 MediaStore 中删除。
- 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
- 查看对象的列表
- 查看对象的详细信息
- 下载对象
- 删除对象

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。如果文件夹已存在，则 AWS Elemental MediaStore 会将对象存储在文件夹中。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 MediaStore 中的文件夹。

可以使用 MediaStore 控制台或 AWS CLI上传对象。

MediaStore 支持对象分块传输，这让您可以下载仍在上传的对象，并以此减少延迟。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。如果您在请求中不指定此标头，MediaStore 会为对象的上传可用性分配默认值 `standard`。

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 AWS CLI 中，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

可以使用 AWS Elemental MediaStore 控制台查看容器或文件夹最顶层存储的项目（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。可以使用 AWS CLI 查看容器中的对象和文件夹的列表，容器内的文件夹或子文件夹的数量对此没有影响。

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 AWS CLI 中，使用 `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 AWS CLI 中，使用 list-items 命令，在请求末尾包含指定的文件夹名称：

```
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

上传对象之后，AWS Elemental MediaStore 将存储详细信息，例如修改日期、内容长度、ETag（实体标签）和内容类型。要了解如何使用某个对象的元数据，请参阅 MediaStore 与 HTTP 缓存的互动。

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 AWS CLI 中，使用 `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。可以使用 AWS CLI 下载整个对象或仅下载对象的一部分。

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 AWS CLI 中，使用 `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 AWS CLI 中，使用 `get-object` 命令，并指定一个范围。

```
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 提供了从容器中删除对象的不同选项：

- 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**
>
> 删除某个文件夹中仅有的对象时，AWS Elemental MediaStore 将自动删除该文件夹及其上面的任何空文件夹。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。

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 Delete (删除)。

删除对象 (AWS CLI)

- 在 AWS CLI 中，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。

AWSElemental 中的安全 MediaStore

云安全位于 AWS 是最高优先级。作为 AWS 客户，您将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。

安全是双方的共同责任 AWS 还有你。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 负责保护运行的基础架构 AWS 中的服务 AWS Cloud. AWS 还为您提供可以安全使用的服务。作为安全措施的一部分，第三方审计师定期测试和验证我们安全的有效性 AWS 合规计划。要了解适用于 AWS Elemental 的合规计划 MediaStore，请参阅 AWS 按合规计划划分的范围内的服务 。
- 云端安全 — 您的责任由 AWS 您使用的服务。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。你还会学习如何使用其他 AWS 帮助您监控和保护 MediaStore 资源的服务。

主题

- AWSElemental 中的数据保护 MediaStore
- AWSElemental 的 Identity and Access 管理 MediaStore
- 登录和监控 AWS Elemental MediaStore
- AWSElemental 的合规性验证 MediaStore
- AWS元素中的韧性 MediaStore
- AWSElemental 中的基础设施安全 MediaStore
- 防止跨服务混淆座席

AWSElemental 中的数据保护 MediaStore

这些区域有： AWS 分担责任模型适用于 AWS Elemental MediaStore 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设施 AWS Cloud。 您有责任保持对托管在此基础架构上的内容的控制。您还负责以下各项的安全配置和管理任务 AWS 服务 你用的。有关数据隐私的更多信息，请参阅数据隐私FAQ。 有关欧洲数据保护的信息，请参阅 AWS 责任共担模型和GDPR博客文章 AWS 安全博客。

出于数据保护的目的，我们建议您进行保护 AWS 账户 凭据并使用设置个人用户 AWS IAM Identity Center 或者 AWS Identity and Access Management (IAM)。这样，每个用户只获得履行其工作职责所需的权限。我们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)。
- 使用SSL/TLS与之通信 AWS 资源的费用。我们需要 TLS 1.2，建议使用 TLS 1.3。
- 使用API进行设置和用户活动记录 AWS CloudTrail。 有关使用 CloudTrail 轨迹进行捕获的信息 AWS 活动，请参阅使用中的 CloudTrail 轨迹 AWS CloudTrail 用户指南。
- 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。
- 如果您在访问时需要 FIPS 140-3 经过验证的加密模块 AWS 通过命令行界面或API，使用FIPS端点。有关可用FIPS端点的更多信息，请参阅联邦信息处理标准 (FIPS) 140-3。

我们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）。这包括当你使用 MediaStore 或其他 AWS 服务 使用控制台，API， AWS CLI，或 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URL向外部服务器提供，我们强烈建议您不要在中包含凭据信息，URL以验证您对该服务器的请求。

数据加密

MediaStore 使用行业标准 AES -256 算法对容器和静态对象进行加密。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，允许有选择地跨源访问您的资源。 MediaStore 使用CORS，您可以允许加载到一个域中的客户端 Web 应用程序与另一个域中的资源进行交互。有关更多信息，请参阅 the section called “CORS 策略”。

AWSElemental 的 Identity and Access 管理 MediaStore

AWS Identity and Access Management (IAM) 是一个 AWS 服务 可帮助管理员安全地控制对以下内容的访问权限 AWS 资源的费用。IAM管理员控制谁可以通过身份验证（登录）和授权（拥有权限）使用 MediaStore 资源。IAM是一个 AWS 服务 无需支付额外费用即可使用。

主题

- 受众
- 使用身份进行身份验证
- 使用策略管理访问
- AWSElemental 是如何使用 MediaStore 的 IAM
- Elemental 基于身份的策略示例 AWS MediaStore
- AWS元素 MediaStore 身份和访问权限疑难解答

受众

你怎么用 AWS Identity and Access Management (IAM) 会有所不同，具体取决于你所做的工作 MediaStore。

服务用户-如果您使用 MediaStore 服务完成工作，则管理员会为您提供所需的凭证和权限。当你使用更多 MediaStore 功能来完成工作时，你可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问中的功能 MediaStore，请参阅AWS元素 MediaStore 身份和访问权限疑难解答。

服务管理员-如果您负责公司的 MediaStore 资源，则可能拥有完全访问权限 MediaStore。您的工作是确定您的服务用户应访问哪些 MediaStore 功能和资源。然后，您必须向IAM管理员提交更改服务用户权限的请求。查看此页面上的信息以了解的基本概念IAM。要详细了解贵公司如何IAM与配合使用 MediaStore，请参阅AWSElemental 是如何使用 MediaStore 的 IAM。

IAM管理员-如果您是IAM管理员，则可能需要详细了解如何编写策略来管理访问权限 MediaStore。要查看可在中使用的 MediaStore 基于身份的策略示例IAM，请参阅。Elemental 基于身份的策略示例 AWS MediaStore

使用身份进行身份验证

身份验证是您登录的方式 AWS 使用您的身份凭证。您必须经过身份验证（登录到 AWS) 作为 AWS 账户根用户、以IAM用户身份或通过担任IAM角色来完成。

你可以登录 AWS 使用通过身份源提供的凭证作为联合身份。 AWS IAM Identity Center （IAM身份中心）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您

以联合身份登录时，您的管理员之前使用IAM角色设置了联合身份。当你访问时 AWS 通过使用联合，您就是在间接担任角色。

根据您的用户类型，您可以登录 AWS Management Console 或者 AWS 访问门户。有关登录的更多信息 AWS，请参阅如何登录您的 AWS 账户中的 AWS 登录 用户指南。

如果你访问 AWS 以编程方式， AWS 提供了一个软件开发套件 (SDK) 和一个命令行界面 (CLI)，用于使用您的凭证对您的请求进行加密签名。如果你不使用 AWS 工具，你必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅签名 AWS APIIAM用户指南中的请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高帐户的安全性。要了解更多信息，请参阅中的多重身份验证 AWS IAM Identity Center 用户指南和使用多因素身份验证 (MFA) 在 AWS（在 IAM 用户指南中）。

AWS 账户 根用户

当你创建 AWS 账户，您从一个登录身份开始，该身份可以完全访问所有人 AWS 服务 以及账户中的资源。这个身份叫做 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以 root 用户身份登录的任务的完整列表，请参阅《用户指南》中的 “需要根用户凭据的IAM任务”。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份进行访问 AWS 服务 通过使用临时证书。

联合身份是企业用户目录中的用户、Web 身份提供商、 AWS Directory Service、身份中心目录或任何访问的用户 AWS 服务 通过使用通过身份源提供的凭证。当联合身份访问时 AWS 账户，他们扮演角色，角色提供临时证书。

要进行集中访问管理，我们建议您使用 AWS IAM Identity Center。 您可以在 Ident IAM ity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有用户和群组中使用 AWS 账户 和应用程序。有关IAM身份中心的信息，请参阅什么是IAM身份中心？ 在 AWS IAM Identity Center 用户指南。

IAM 用户和组

IAM用户是你内部的身份 AWS 账户 对个人或应用程序具有特定权限。在可能的情况下，我们建议使用临时证书，而不是创建拥有密码和访问密钥等长期凭证的IAM用户。但是，如果您有需要IAM用户长期

凭证的特定用例，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM用户指南》中的定期轮换需要长期凭证的用例的访问密钥。

IAM群组是指定IAM用户集合的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可以拥有一个名为的群组，IAMAdmins并授予该群组管理IAM资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM用户指南》中的何时创建IAM用户（而不是角色）。

IAM角色

IAM角色是你内在的身份 AWS 账户 具有特定权限的。它与IAM用户类似，但与特定人员无关。你可以暂时扮IAM演一个角色 AWS Management Console 通过切换角色。你可以通过调用来扮演角色 AWS CLI 或者 AWS API操作或使用自定义URL。有关使用角色的方法的更多信息，请参阅IAM用户指南中的使用IAM角色。

IAM具有临时证书的角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关用于联合身份验证的角色的信息，请参阅《IAM用户指南》中的为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为了控制您的身份在进行身份验证后可以访问的内容，Ident IAM ity Center 会将权限集关联到中的IAM角色。有关权限集的信息，请参阅中的权限集 AWS IAM Identity Center 用户指南。
- 临时IAM用户权限-IAM 用户或角色可以代入一个IAM角色，为特定任务临时获得不同的权限。
- 跨账户访问-您可以使用IAM角色允许其他账户中的某人（受信任的委托人）访问您账户中的资源。角色是授予跨账户访问权限的主要方式。但是，有些 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解角色和基于资源的跨账户访问策略之间的区别，请参阅IAM用户指南IAM中的跨账户资源访问权限。
- 跨服务访问 — 一些 AWS 服务 使用其他功能 AWS 服务。 例如，当您在服务中拨打电话时，该服务通常会在 Amazon 中运行应用程序EC2或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS)-当您使用IAM用户或角色在中执行操作时 AWS，你被视为校长。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS使用委托人的权限调用 AWS 服务，再加上请求的 AWS 服务 向下游服务发出请求。FAS只有当服务收到需要与其他

服务进行交互的请求时，才会发出请求 AWS 服务 或需要完成的资源。在这种情况下，您必须具有执行这两个操作的权限。有关提出FAS请求时的政策详情，请参阅转发访问会话。

- 服务角色-服务IAM角色是服务代替您执行操作的角色。IAM管理员可以在内部创建、修改和删除服务角色IAM。有关更多信息，请参阅创建角色以向某人委派权限 AWS 服务（在 IAM 用户指南中）。
- 服务相关角色-服务相关角色是一种与服务相关联的服务角色 AWS 服务。 该服务可以代替您执行操作。服务相关角色显示在您的 AWS 账户 并归该服务所有。IAM管理员可以查看但不能编辑服务相关角色的权限。

- 在 Amazon 上运行的应用程序 EC2 — 您可以使用IAM角色管理在EC2实例上运行的应用程序的临时证书 AWS CLI 或者 AWS API请求。这比在EC2实例中存储访问密钥更可取。要分配 AWS 在EC2实例中扮演角色并使其可供其所有应用程序使用，则可以创建附加到该实例的实例配置文件。实例配置文件包含角色并允许在EC2实例上运行的程序获得临时证书。有关更多信息，请参阅IAM用户指南中的使用IAM角色向在 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用IAM角色还是使用IAM用户，请参阅《用户指南》中的何时创建IAM角色（而不是IAM用户）。

使用策略管理访问

您可以控制访问权限 AWS 通过创建策略并将其附加到 AWS 身份或资源。策略是中的一个对象 AWS 当与身份或资源关联时，它定义了他们的权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都存储在 AWS 作为JSON文件。有关JSON策略文档结构和内容的更多信息，请参阅《IAM用户指南》中的JSON策略概述。

管理员可以使用 AWS JSON用于指定谁有权访问什么的策略。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。

默认情况下，用户和角色没有权限。要授予用户对其所需资源执行操作的权限，IAM管理员可以创建IAM策略。然后，管理员可以将IAM策略添加到角色中，用户可以代入角色。

IAM无论您使用何种方法执行操作，策略都会定义该操作的权限。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从中获取角色信息 AWS Management Console，AWS CLI，或者 AWS API.

基于身份的策略

基于身份的策略是可以附加到身份（例如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用户指南中的创建IAM策略。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到您的多个用户、群组和角色 AWS 账户。 托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管策略或内联策略之间进行选择，请参阅《IAM用户指南》中的在托管策略和内联策略之间进行选择。

基于资源的策略

基于资源的JSON策略是您附加到资源的策略文档。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.

基于资源的策略是位于该服务中的内联策略。你不能用 AWS 基于资源的策略IAM中的托管策略。

访问控制列表 (ACLs)

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。

亚马逊 S3， AWS WAF，Amazon VPC 就是支持的服务示例ACLs。要了解更多信息ACLs，请参阅《亚马逊简单存储服务开发者指南》中的访问控制列表 (ACL) 概述。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界-权限边界是一项高级功能，您可以在其中设置基于身份的策略可以向IAM实体（IAM用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，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)-SCPs 是指定组织或组织单位 (OU) 的最大权限的JSON策略 AWS Organizations. AWS Organizations 是一项用于对多个进行分组和集中管理的服务 AWS 账户 你的企业拥有的。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或所有帐

户。SCP限制了成员账户中实体的权限，包括每个 AWS 账户根用户。 有关 Organizations 和的更多信息SCPs，请参阅《》中的服务控制策略 AWS Organizations 用户指南。

- 会话策略 – 会话策略是当您以编程方式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。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。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。要了解如何 AWS 决定在涉及多种策略类型时是否允许请求，请参阅IAM用户指南中的策略评估逻辑。

AWSElemental 是如何使用 MediaStore 的 IAM

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

IAM你可以在 AWS Elemental 中使用的功能 MediaStore

IAM特征	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	不支持
ABAC（策略中的标签）	部分
临时凭证	是
主体权限	是
服务角色	是

IAM特征	MediaStore 支持
服务相关角色	否

要从高层次了解如何 MediaStore 以及其他 AWS 服务适用于大多数IAM功能，请参阅 AWS IAM在《IAM用户指南》中使用的服务。

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以附加到身份（例如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用户指南中的创建IAM策略。

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。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可以在JSON策略中使用的所有元素，请参阅IAM用户指南中的IAMJSON策略元素参考。

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。Elemental 基于身份的策略示例 AWS 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JSON策略是您附加到资源的策略文档。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另一个账户中的IAM实体指定为基于资源的策略中的委托人。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同状态时 AWS 账户，可信账户中的IAM管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM用户指南》IAM中的跨账户资源访问权限。

> **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用于指定谁有权访问什么的策略。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`Action`元素描述了可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的同名 AWS API操作。也有一些例外，例如没有匹配API操作的仅限权限的操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

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。Elemental 基于身份的策略示例 AWS MediaStore

的政策资源 MediaStore

支持策略资源：是

管理员可以使用 AWS JSON用于指定谁有权访问什么的策略。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。

ResourceJSON策略元素指定要应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。最佳做法是，使用资源的 Amazon 资源名称 (ARN) 来指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示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ARN种资源，请参阅 AWSElemental MediaStore 定义的操作。

MediaStore 容器资源具有以下内容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，请使用以下命令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用于指定谁有权访问什么的策略。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。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。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。

如果您在一条语句中指定多个Condition元素，或者在单个Condition元素中指定多个键， AWS 使用逻辑AND运算对其进行评估。如果您为单个条件键指定多个值， AWS 使用逻辑OR运算评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在资源上标有IAM用户的用户名时，您才能向IAM用户授予访问该资源的权限。有关更多信息，请参阅《IAM用户指南》中的IAM策略元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。要查看全部 AWS 全局条件键，请参见 AWS 《IAM用户指南》中的全局条件上下文密钥。

要查看 MediaStore 条件键列表，请参阅《服务授权参考》 MediaStore中的 AWSElemental 的条件密钥。要了解可以使用条件键的操作和资源，请参阅 AWSElemental MediaStore 定义的操作。

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

ACLs在 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与 MediaStore

支持ABAC（策略中的标签）：部分

基于属性的访问控制 (ABAC) 是一种基于属性定义权限的授权策略。In AWS，这些属性称为标签。您可以将标签附加到IAM实体（用户或角色）和许多实体 AWS 资源的费用。为实体和资源添加标签是的第一步。ABAC然后，您可以设计ABAC策略，允许在委托人的标签与他们尝试访问的资源上的标签匹配时进行操作。

ABAC在快速增长的环境中很有用，也有助于解决策略管理变得繁琐的情况。

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。

有关的更多信息ABAC，请参阅什么是ABAC？ 在《IAM用户指南》中。要查看包含设置步骤的教程ABAC，请参阅IAM用户指南中的使用基于属性的访问控制 (ABAC)。

将临时凭证与配合使用 MediaStore

支持临时凭证：是

一段时间 AWS 服务 使用临时凭证登录时不起作用。欲了解更多信息，包括哪个 AWS 服务 使用临时证书，请参阅 AWS 服务 可以IAM在《IAM用户指南》中使用。

如果您登录，则使用的是临时证书 AWS Management Console 使用除用户名和密码之外的任何方法。例如，当你访问时 AWS 使用贵公司的单点登录 (SSO) 链接，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM用户指南》中的切换到角色（控制台）。

您可以使用手动创建临时证书 AWS CLI 或者 AWS API。然后，您可以使用这些临时证书进行访问 AWS. 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，请参阅中的临时安全证书IAM。

的跨服务主体权限 MediaStore

支持转发访问会话 (FAS)：是

当您使用IAM用户或角色在中执行操作时 AWS，你被视为校长。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS使用委托人的权限调用 AWS 服务，再加上请求的 AWS 服务 向下游服务发出请求。FAS只有当服务收到需要与其他服务进行交互的请求时，才会发出请求 AWS 服务 或需要完成的资源。在这种情况下，您必须具有执行这两个操作的权限。有关提出FAS请求时的政策详情，请参阅转发访问会话。

MediaStore 的服务角色

支持服务角色：是

服务IAM角色是服务代替您执行操作的角色。IAM管理员可以在内部创建、修改和删除服务角色IAM。有关更多信息，请参阅创建角色以向某人委派权限 AWS 服务（在 IAM 用户指南中）。

> **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与服务相关联的服务角色 AWS 服务。 该服务可以代替您执行操作。服务相关角色显示在您的 AWS 账户 并归该服务所有。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 AWS 与之配合使用的服务IAM。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。

Elemental 基于身份的策略示例 AWS MediaStore

默认情况下，用户和角色无权创建或修改 MediaStore资源。他们也无法通过使用来执行任务 AWS Management Console, AWS Command Line Interface (AWS CLI)，或 AWS API。要授予用户对其所需资源执行操作的权限，IAM管理员可以创建IAM策略。然后，管理员可以将IAM策略添加到角色中，用户可以代入角色。

要了解如何使用这些示例策略文档创建IAM基于身份的JSON策略，请参阅IAM用户指南中的创建IAM策略。

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ARNs详细信息，请参阅《服务授权参考》 MediaStore中的 AWSElemental 的操作、资源和条件键。

主题

- 策略最佳实践
- 使用控制 MediaStore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。这些操作可能会使您付出代价 AWS 账户。 创建或编辑基于身份的策略时，请遵循以下准则和建议：

- 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 AWS 为许多常见用例授予权限的托管策略。它们在你的 AWS 账户。 我们建议您通过定义来进一步减少权限 AWS 特定于您的用例的客户托管政策。有关更多信息，请参阅 AWS 托管策略或 AWS 《IAM用户指南》 中工作职能的托管策略。
- 应用最低权限权限-使用IAM策略设置权限时，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。有关使用应用权限IAM的更多信息，请参阅IAM用户指南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-您可以在策略中添加条件以限制对操作和资源的访问权限。例如，您可以编写一个策略条件来指定所有请求都必须使用发送SSL。如果通过特定条件使用服务操作，则也可以使用条件来授予对服务操作的访问权限 AWS 服务之外的压缩算法（例如 AWS CloudFormation。 有关更多信息，请参阅《IAM用户指南》中的IAMJSON策略元素：条件。

- 使用 A IAM ccess Analyzer 验证您的IAM策略以确保权限的安全性和功能性 — A IAM ccess Analyzer 会验证新的和现有的策略，以便策略符合IAM策略语言 (JSON) 和IAM最佳实践。IAMAccess Analyzer 提供了 100 多项策略检查和可行的建议，可帮助您制定安全和实用的策略。有关更多信息，请参阅《IAM用户指南》中的 IAMAccess Analyzer 策略验证。
- 需要多因素身份验证 (MFA)-如果您的场景需要IAM用户或 root 用户 AWS 账户，请打开MFA以提高安全性。要要求MFA何时调用API操作，请在策略中添加MFA条件。有关更多信息，请参阅《IAM用户指南》中的配置MFA受保护的API访问权限。

有关中最佳做法的更多信息IAM，请参阅《IAM用户指南》IAM中的安全最佳实践。

使用控制 MediaStore台

要访问 AWS Elemental MediaStore 控制台，你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。 如果您创建的基于身份的策略比所需的最低权限更严格，则控制台将无法按预期运行，适用于使用该策略的实体（用户或角色）。

您无需为仅拨打控制台的用户设置最低控制台权限 AWS CLI 或者 AWS API。相反，只允许访问与他们尝试执行的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，还需要附上 MediaStore *ConsoleAccess*或 *ReadOnly* AWS 针对实体的托管策略。有关更多信息，请参阅《用户指南》中的向IAM用户添加权限。

允许用户查看他们自己的权限

此示例说明如何创建允许IAM用户查看附加到其用户身份的内联和托管策略的策略。此策略包括通过控制台或以编程方式使用控制台完成此操作的权限 AWS CLI 或者 AWS API.

```
{
    "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": "*"
        }
    ]
}
```

AWS元素 MediaStore 身份和访问权限疑难解答

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

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进入 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，表明您无权执行某个操作，则您必须更新策略以允许执行该操作。

当mateojacksonIAM用户尝试使用控制台查看虚构*my-example-widget*资源的详细信息但没有虚构权限时，就会出现以下示例错误。mediastore:*GetWidget*

```
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

如果您收到错误消息，提示您无权执行iam:PassRole操作，则必须更新您的策略以允许您将角色传递给 MediaStore。

一段时间 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。

当名为的IAM用户marymajor尝试使用控制台在中执行操作时，会出现以下示例错误 MediaStore。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进入 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，可以担任角色。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AWSElemental 是如何使用 MediaStore 的 IAM。
- 要了解如何提供对您的资源的访问权限 AWS 账户 您拥有的，请参阅向其他IAM用户提供访问权限 AWS 账户 您在《IAM用户指南》中拥有的内容。
- 了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅提供访问权限 AWS 账户IAM用户指南中归第三方所有。
- 要了解如何通过联合身份验证提供访问权限，请参阅《用户指南》中的向经过外部身份验证的用户提供访问权限（联合身份验证）。IAM
- 要了解使用角色和基于资源的策略进行跨账户访问的区别，请参阅IAM用户指南IAM中的跨账户资源访问权限。

登录和监控 AWS Elemental MediaStore

本节概述了中用于记录和监控的选项 AWS Elemental MediaStore 出于安全考虑。有关登录和监控的更多信息 MediaStore，请参阅在 AWS Elemental MediaStore 中进行监控和标记。

监控是维护可靠性、可用性和性能的重要组成部分 AWS Elemental MediaStore 还有你的 AWS 解决方案。你应该从你的各个部分收集监控数据 AWS 解决方案，以便在出现多点故障时可以更轻松地对其进行调试。 AWS 提供了多种用于监控您的 MediaStore 资源和响应潜在事件的工具。

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阈值，则会向亚马逊SNS主题或 A AWS uto 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 CloudWatch 进行监控。

AWS CloudTrail 日志

CloudTrail 提供用户、角色或用户所执行操作的记录 AWS 服务于 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人提供服务中学到的最佳实践 AWS 客户。 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。全部 AWS 客户可以访问五张 Trusted Advisor 支票。拥有商业或企业支持计划的客户可以查看全部 Trusted Advisor 支票。

有关更多信息，请参阅 AWS Trusted Advisor.

AWSElemental 的合规性验证 MediaStore

要了解是否 AWS 服务 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 划分的范围” 中，选择您感兴趣的合规计划。有关一般信息，请参见 AWS 合规计划 。

您可以使用以下方式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 AWS Artifact.

您在使用时的合规责任 AWS 服务 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。 AWS 提供了以下资源来帮助实现合规性：

- 安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了在上部署基准环境的步骤 AWS 以安全性和合规性为重点。
- 在 Amazon Web Services 上进行HIPAA安全与合规架构 ——本白皮书描述了各公司如何使用 AWS 创建HIPAA符合条件的应用程序。

> **Note**
>
> 不是全部 AWS 服务 符合HIPAA资格。有关更多信息，请参阅《HIPAA合格服务参考》。

- AWS 合规资源 — 此工作簿和指南集可能适用于您所在的行业和所在地。
- AWS 客户合规指南 — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践 AWS 服务并将指南映射到跨多个框架（包括美国国家标准与技术研究所 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- 使用中的规则评估资源 AWS Config 开发者指南 — AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 这个 AWS 服务 提供您的安全状态的全面视图 AWS。 Security Hub 使用安全控制来评估你的 AWS 资源，并检查您是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- 亚马逊 GuardDuty — 这个 AWS 服务 检测您面临的潜在威胁 AWS 账户、工作负载、容器和数据，监控您的环境中是否存在可疑和恶意活动。 GuardDuty 可以帮助您满足各种合规性要求 PCIDSS，例如满足某些合规性框架规定的入侵检测要求。
- AWS Audit Manager— 这个 AWS 服务 帮助您持续审核您的 AWS 用于简化风险管理以及对法规和行业标准的合规性。

AWS元素中的韧性 MediaStore

这些区域有： AWS 全球基础设施是围绕着建立的 AWS 区域 和可用区。 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、容错性和可扩展性。

有关 AWS 区域 和可用区，请参阅 AWS 全球基础设施。

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

AWSElemental 中的基础设施安全 MediaStore

作为一项托管服务，AWSElemental 受以下 MediaStore 条款的保护 AWS 全球网络安全。有关信息 AWS 安全服务及其方式 AWS 保护基础架构，请参阅 AWS 云安全。设计你的 AWS 使用基础设施安全最佳实践的环境，请参阅安全支柱中的基础设施保护 AWS 架构精良的框架。

你用 AWS 已发布 MediaStore 通过网络访问的API呼叫。客户端必须支持以下内容：

- 传输层安全 (TLS)。我们需要 TLS 1.2，建议使用 TLS 1.3。
- 具有完美前向保密性的密码套件 ()，例如（Ephemeral Diffie-HellmanPFS）或（Elliptic C DHE urve Ephemeral Diffie-Hellman）。ECDHE大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与IAM委托人关联的私有访问密钥对请求进行签名。或者你可以使用 AWS Security Token Service (AWS STS) 生成用于签署请求的临时安全证书。

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。In 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。可以操纵调用服务以使用其权限对另一个客户的资源进行操作，否则该服务不应有访问权限。为了防止这种情况， AWS 提供的工具可帮助您保护所有服务的数据，这些服务主体已被授予访问您账户中资源的权限。

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 MediaStore 向该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`。

防止混乱的副手问题的最有效方法是使用具有全部资源的全`aws:SourceArn`ARN局条件上下文密钥。如果您不知道资源的全部ARN内容，或者要指定多个资源，请使用带有通配符 (*) 的`aws:SourceArn`全局上下文条件键来表示未知部分。ARN例如，`arn:aws:`*`servicename`*`:*:`*`123456789012`*`:*`。

如果该`aws:SourceArn`值不包含账户 ID，例如 Amazon S3 存储桶ARN，则必须同时使用两个全局条件上下文密钥来限制权限。

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。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Sid": "ConfusedDeputyPreventionExamplePolicy",
    "Effect": "Allow",
    "Principal": {
      "Service": "servicename.amazonaws.com"
    },
    "Action": "servicename:ActionName",
    "Resource": [
      "arn:aws:servicename:::ResourceName/*"
    ],
    "Condition": {
      "ArnLike": {
        "aws:SourceArn": "arn:aws:servicename:*:123456789012:*"
      },
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
      }
    }
  }
}
```

在 AWS Elemental MediaStore 中进行监控和标记

监控是保持 AWS Elemental MediaStore 和您其他AWS解决方案AWS的可靠性、可用性和性能的重要方面。提供了以下一些监控工具来监控 MediaStore，在出现错误时进行报告并适时自动采取措施。

- AWS CloudTrail 捕获由您的AWS账户或代表该账户发出的 API 调用和相关事件，并将日志文件传输到您指定的 Simple Storage Service (Amazon S3) 存储桶。您可以标识哪些用户和账户调用了AWS、从中发出调用的源 IP 地址以及调用的发生时间。有关更多信息，请参阅 AWS CloudTrail《用户指南》。
- Amazon CloudWatch 实时监控您的 AWS 资源以及在 AWS 上运行的应用程序。您可以收集和跟踪指标，创建自定义的控制面板，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，您可以具有 Amazon EC2 实例的 CloudWatch 跟踪 CPU 使用率或其他指标并且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 提供系统事件流，这些事件描述 AWS 资源的更改。通常，AWS 服务会在几秒钟内将事件通知传送到 CloudWatch Events，不过有时可能需要一分钟或更长时间。CloudWatch Events 支持自动事件驱动型计算，因为您可以编写规则，以监控某些事件和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 CloudWatch Events《用户指南》。
- Amazon CloudWatch Logs 使您能够监控、存储和访问来自 Amazon EC2 实例、CloudTrail 和其他来源的日志文件。CloudWatch Logs 可以监控日志文件中的信息，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 Logs《用户指南》。

您也可以以标签的形式将元数据分配给您的 MediaStore 容器。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。您可以使用标签来整理 AWS 管理控制台中的 AWS 资源，围绕您的所有 AWS 资源创建使用情况和账单报告，以及在基础设施自动化活动期间筛选资源。

主题

- 使用 AWS CloudTrail记录 AWS Elemental MediaStore API 调用
- 使用 Amazon CloudWatch 监控 AWS Elemental MediaStore
- 标记 AWS Elemental MediaStore 资源

使用 AWS CloudTrail记录 AWS Elemental MediaStore API 调用

AWS Elemental MediaStore 与 AWS CloudTrail 集成，后者是一项服务，提供 Amazon ECR 中由用户、角色或 AWS MediaStore 服务所采取操作的记录。CloudTrail 将对 MediaStore 的 API 调用子集作为事件捕获，包括来自 MediaStore 控制台的调用和对 MediaStore API 的代码调用。如果您创建跟踪记录，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 MediaStore 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的 Event history（事件历史记录）中查看最新事件。使用 CloudTrail 收集的信息，您可以确定向 MediaStore 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间等。

要了解有关 CloudTrail 的更多信息（包括如何对其进行配置和启用），请参阅 AWS CloudTrail《用户指南》。

主题

- CloudTrail 中的 AWS Elemental MediaStore 信息
- 示例：AWS Elemental MediaStore 日志文件条目

CloudTrail 中的 AWS Elemental MediaStore 信息

在您创建 AWS 账户时，将在该账户上启用 CloudTrail。当 AWS Elemental MediaStore 中发生受支持的事件活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同保存在 Event history（事件历史记录）中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录 AWS 账户中的事件（包括 MediaStore 的事件），请创建跟踪。通过跟踪，CloudTrail 可将日志文件传送至 Amazon S3 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。此跟踪记录在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service (Amazon S3) 桶。此外，您可以配置其它AWS服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅以下主题：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

AWS Elemental MediaStore 支持在 CloudTrail 日志文件中将以下操作记录为事件：

- 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。身份信息可帮助您确定以下内容：

- 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身份用户的临时安全凭证发出的
- 请求是否由其它 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，可用于将事件作为日志文件传送到您指定的 Amazon S3 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，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 CreateContainer 操作。

```
{
    "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 Amazon CloudWatch 监控 AWS Elemental MediaStore

您可以使用 CloudWatch 监控 AWS Elemental MediaStore。CloudWatch 会收集原始数据并将其处理为易读指标。CloudWatch 会保存这些统计数据 15 个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。此外，可以设置用于监测特定阈值的警报，并在达到相应阈值时发送通知或执行操作。有关更多信息，请参阅 Amazon CloudWatch《用户指南》。

AWS 提供以下监控工具来监控 MediaStore、在出现错误时进行报告并在适当的时候自动采取措施：

- Amazon CloudWatch Logs 使您能够监控、存储和访问来AWS自 AWS Elemental MediaStore 等服务的日志文件。您可以使用 CloudWatch Logs 通过日志数据来监控应用程序和系统。例如，CloudWatch Logs 能够跟踪应用程序日志中发生的错误数，并在错误率超过指定阈值时向您发送通知。CloudWatch Logs 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以查找特定的文本字词（如“ValidationException”）或为在特定时间段内发出的 `PutObject` 请求计数。找到您所搜索的字词时，CloudWatch Logs 会向您指定的 CloudWatch 指标报告该数据。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 提供系统事件，这些系统事件描述了AWS资源（例如 MediaStore 对象）中的变化。通常，AWS服务会在几秒钟内将事件通知传送到 CloudWatch Events，不过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），并将这些事件路由到一个或多个目标函数或流。CloudWatch Events 随着运营变化的发生而发现。此外，CloudWatch Events 将响应这些操作更改并在必要时采取纠正措施，方式是发送消息以响应环境、激活函数、进行更改并捕获状态信息。

CloudWatch Logs

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。它们还可以帮助您了解您的客户群并了解您的 MediaStore 账单。CloudWatch Logs 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。当您在容器上启用访问日志记录时，MediaStore 会创建一个日志组，名称如 `/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 CloudWatch 设置权限

使用 AWS Identity and Access Management (IAM) 创建一个角色来授予 AWS Elemental MediaStore 访问Amazon CloudWatch 的权限。您必须执行以下步骤才能为您的账户发布 CloudWatch Logs。CloudWatch 会自动为您的账户发布指标。

允许 MediaStore 访问 CloudWatch

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。

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：

```
{
    "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/*"
        }
    ]
}
```

   此策略允许 MediaStore 为您AWS账户中任何区域的任何容器创建日志组和日志流。

4. 选择Review policy（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 Roles，然后选择 Create role。
7. 选择 其他 Amazon Web Services 账户 角色类型。
8. 对于 Account ID (账户 ID)，输入您的 AWS 账户 ID。
9. 选择Next: Permissions（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 Role name (角色名称)，输入 **MediaStoreAccessLogs**，然后选择 Create role (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 Edit trust relationship (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：

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

18. 选择 Update Trust Policy（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 不会收集访问日志。当您在容器中启用访问日志记录时，MediaStore 会将该容器中存储的对象的访问日志记录传送给 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。这些信息可能包括请求类型、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 AWS CLI 中，使用 `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记录时，AWS Elemental MediaStore 停止发送访问日志到 Amazon 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 AWS CLI 中，使用 `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 MediaStore 中的访问日志进行故障排除

当 AWS Elemental MediaStore 访问日志记录未显示在 Amazon CloudWatch 中时，请参阅下表来获取潜在原因和解决方法。

> **Note**
>
> 请务必启用 AWS CloudTrail Logs 来协助问题排查过程。

症状	该问题可能是...	尝试这项操作...
即使您启用了 CloudTrail 日志，也没有看到任何 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 “设置 CloudWatch 的权限”。
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 “设置 CloudWatch 的权限”。
	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 控制台中，您会看到一个事件，此事件具有与 `MediaStoreAccessLogs` 角色相关的拒绝访问错误。CloudTrail 事件可能包含以下数行内容：
 	
`"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-2:1111`
	IAM 角色没有针对 AWS Elemental MediaStore 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 “设置 CloudWatch 的权限”。

症状	该问题可能是...	尝试这项操作...
22223333:log-group::log-stream:",		
您没有看到整个容器或多个容器的任何日志。	您的账户可能已超出每个区域每个账户的 CloudWatch 日志组配额。请参阅 Amazon CloudWatch Logs《用户指南》中的日志组配额。	在 CloudWatch 控制台中，确定您的账户是否已满足日志组的 CloudWatch 配额。如有必要，您可以请求提高配额。
您看到 CloudWatch 中的一些日志，但没有看到所有想要查看的日志。	您的账户可能已超出每个区域每个账户每秒事务数的 CloudWatch 配额。请参阅 Amazon CloudWatch Logs《用户指南》PutLogEvents 中的配额。	为 每个区域设置每个区域的每个账户每秒 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":
 "dddDDD444eeeEEE555fffFFF666gggGGG777hhhHHH888iiiIII999jjjJJJ000cccCCC333bbbBBB222aaaAAA",
  "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：

AWSAccountId

用于发出请求的账户的 AWS 账户 ID 。

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 MediaStore 生成的字符串，可用于唯一地标识每个请求。

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

MediaStore 处理您的请求所花费的毫秒数。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 Management Console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 Events

Amazon CloudWatch Events 使您能够实现AWS 服务自动化，以自动响应系统事件，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服务会在几秒钟内将事件通知传送到 CloudWatch Events，不过有时可能需要一分钟或更长时间。

当文件上传到容器或从容器中删除后，CloudWatch 服务中将连续触发两个事件：

1. the section called “对象状态更改事件”
2. the section called “容器状态更改事件”

有关订阅这些事件的信息，请参阅Amazon CloudWatch。

可自动触发的操作包括：

- 调用 AWS Lambda 函数
- 调用 Amazon EC2 Run Command
- 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 AWS SMS队列

一些将 CloudWatch Events 与 AWS Elemental MediaStore 结合使用的示例包括：

- 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 CloudWatch Events《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。

有关订阅此事件的信息，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。有关订阅此事件的信息，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 Amazon CloudWatch 指标监控 AWS Elemental MediaStore

您可以使用 CloudWatch 监控 AWS Elemental MediaStore。CloudWatch 会收集原始数据并将其处理为易读指标。CloudWatch 会保存这些统计数据 15 个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。此外，可以设置用于监测特定阈值的警报，并在达到相应阈值时发送通知或执行操作。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，您可能需要监测`BytesDownloaded`并向自己发送电子邮件。

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 Management Console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 Metrics（指标）。
3. 在 全部指标 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。

使用 AWS CLI 查看指标

- 在命令提示符处，输入以下命令：

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

AWS Elemental MediaStore 指标

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

> **Note**
>
> 要查看指标，您必须向容器添加指标策略，以允许 MediaStore 向 Amazon CloudWatch 发送指标。

指标	描述
RequestCount	对 Put MediaStore 容器发出的 HTTP 请求总数，按操作类型（、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	导致 4xx 错误的对 MediaStore 发出的 HTTP 请求数。 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

指标	描述
5xxErrorCount	导致 5xx 错误的对 MediaStore 发出的 HTTP 请求数。 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
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

指标	描述
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
TotalTime	从服务器的角度传输请求的毫秒数。此值计算从 MediaStore 收到您的请求到它发送响应最后一个字节之间的时间。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

指标	描述
TurnaroundTime	MediaStore 处理您的请求所花费的毫秒数。此值计算从 MediaStore 收到您的请求的最后一个字段到它发送响应最后一个字节之间的时间。 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
ThrottleCount	向 MediaStore 发出的 HTTP 请求的数量受到限制。 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 Elemental MediaStore 资源

标签是您分配的自定义属性标签，或者是 AWS 分配给 AWS 资源的标签。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：

- 标识和整理您的 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，您可以将同一标签分配给您分配给AWS Elemental MediaLive输入的 AWS Elemental MediaStore##。
- 跟踪您的 AWS 成本。您在 AWS Billing and Cost Management 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。

以下各部分提供有关 AWS Elemental MediaStore 的标签的更多信息。

AWS Elemental MediaStore 中支持的资源

AWS Elemental MediaStore 中的以下资源支持标记：

- ##

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) 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约定适用于将标签与 AWS Elemental MediaStore 资源一起使用的情况：

- 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。
- 对标签禁止使用 `aws:` 前缀；它是为使用 AWS 而保留的。您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。

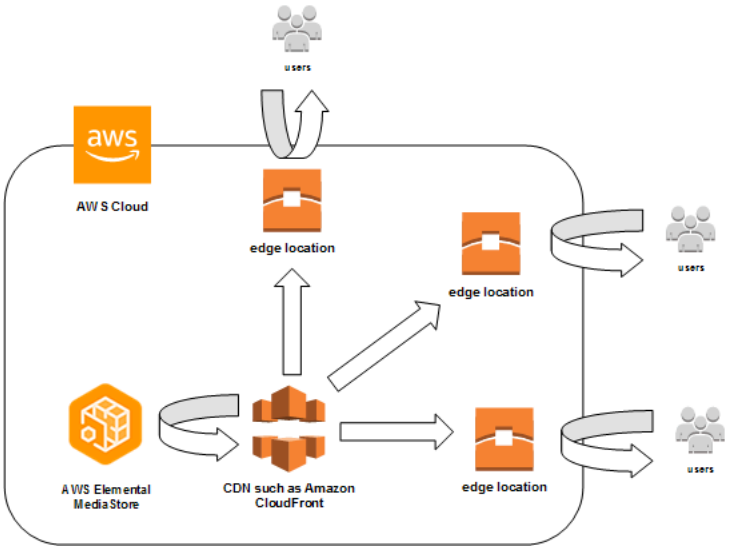
管理标签

标签由资源上的 `Key` 和 `Value` 属性构成。可以使用 AWS CLI 或 MediaStore API 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

使用内容分发网络 (CDN)

您可以使用内容分发网络 (CDN)（如 Amazon CloudFront）来提供您存储在 AWS Elemental MediaStore 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。如果您的内容目前不在该边缘站点中，CDN 将从您的原始服务器（如您的 MediaStore 容器）检索内容并将其分发到用户。



主题

- 允许 Amazon CloudFront 访问您的 AWS Elemental MediaStore 容器
- AWS Elemental MediaStore 与 HTTP 缓存的互动

允许 Amazon CloudFront 访问您的 AWS Elemental MediaStore 容器

您可以使用 Amazon CloudFront 来提供您存储在 AWS Elemental MediaStore 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 CloudFront 的 OAC 功能，请使用此选项。
- 使用共享密钥如果您AWS 区域不支持 CloudFront 的 OAC 功能，请使用此选项。

使用来源访问控制 (OAC)

您可以使用 Amazon CloudFront 的源站访问控制 (OAC) 功能，通过更高的安全性来保护 AWS Elemental MediaStore 的来源。您可以对 MediaStore 来源的 CloudFront 请求启用AWS签名版本 4 (SigV4)，并设置 CloudFront 应何时以及是否应签署请求。您可以通过控制台、API、软件开发工具包或 CLI 访问 CloudFront 的 OAC 功能，使用该功能无需支付额外费用。

有关在 MediaStore 中使用 OAC 功能的更多信息，请参阅 Amazon CloudFront 开发者指南中的限制对 MediaStore 来源的访问。

使用共享密钥

如果您AWS 区域不支持 Amazon CloudFront 的 OAC 功能，则可以将策略附加到您的 AWS Elemental MediaStore 容器，授予对 CloudFront 的读取权限或更高的读取权限。

> **Note**
>
> 如果您AWS 区域支持 OAC 功能，我们建议您使用该功能。以下过程要求您使用共享密钥配置 MediaStore 和 CloudFront，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。通过 MediaStore 来源上的 OAC，您可以指示 CloudFront 使用 SigV4 签署请求并将其转发到 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，从而使通过 MediaStore 和 CloudFront 交付媒体内容变得更加简单和安全。

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 容器策略 部分中，将附加一个授予对 Amazon CloudFront 的读取访问权限或更高权限的策略。

   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。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudFrontRead",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "mediastore:GetObject",
        "mediastore:DescribeObject"
      ],
      "Resource": "arn:aws:mediastore:<region>:<owner acct
 number>:container/<container name>/*",
      "Condition": {
        "StringEquals": {
          "aws:Referer": "<secretValue>"
        },
        "Bool": {
          "aws:SecureTransport": "true"
        }
      }
    }
 ]}
```

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 CloudFront 中，数据终端节点也称为源域名。
   - 存储对象的容器中的文件夹结构。在 CloudFront 中，这也称为源路径。请注意，此设置为可选。有关源路径的更多信息，请参阅Amazon CloudFront 开发人员指南。

6. 在 CloudFront 中，创建配置为从 AWS Elemental MediaStore 提供内容的分配。您将需要在上一步中收集的信息。

将策略附加到您的 MediaStore 容器后，您必须将 CloudFront 配置为仅对源请求使用 HTTPS 连接，并添加具有正确密钥值的自定义标头。

将 CloudFront 配置为通过 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 CloudFront 控制台
2. 在 Origins 页面上，选择您的 MediaStore 来源。
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 MediaStore 与 HTTP 缓存的互动

AWS Elemental MediaStore 存储对象的方式使得像 Amazon CloudFront 这样的内容分发网络 (CDN) 可以正确高效地缓存对象。当最终用户或 CDN 从 MediaStore 中检索对象时，服务将返回影响对象缓存行为的 HTTP 标头。（HTTP 1.1 缓存标准行为可在 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。符合标准的 CDN 和 Web 浏览器使用此标签作为缓存对象的密钥。上传对象时，MediaStore 会自动为其生成一个ETag。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。上传对象时，MediaStore 会自动生成该数值。
- **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 API 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 MediaStore 在检索对象时不会返回此标头。

  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

向 MediaStore 发出的有条件请求

MediaStore 以相同的方式响应有条件请求（使用 `If-Modified-Since`和 `If-None-Match` 等请求标头，如 RFC7232 中所述）和无条件请求。这意味着当 MediaStore 收到有效的 `GetObject` 请求时，即使客户端已经拥有对象，服务也始终返回该对象。

对 CDN 的有条件请求

代表 MediaStore 提供内容的 CDN 可以通过返回 `304 Not Modified` 处理有条件请求，如 RFC7232 第 4.1 节中所述。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。

CDN（以及符合 HTTP/1.1 的其他缓存）基于源服务器转发的 `ETag` 和 `Cache-Control` 标头做出这些决策。要控制 CDN 查询 MediaStore 源服务器以获取对重复检索对象的更新的频率，请在将这些对象上传到 MediaStore 时设置其 `Cache-Control` 标头。

将此服务与 AWS SDK

AWS 软件开发套件 (SDKs) 可用于许多流行的编程语言。每个版本都SDK提供了API代码示例和文档，使开发人员可以更轻松地使用自己的首选语言构建应用程序。

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

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

以下代码示例演示如何 MediaStore 与 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。

有关完整列表 AWS SDK开发者指南和代码示例，请参阅将此服务与 AWS SDK。本主题还包括有关入门的信息以及有关先前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搭配使用 AWS SDK或 CLI
    - DeleteContainer搭配使用 AWS SDK或 CLI
    - DeleteObject搭配使用 AWS SDK或 CLI
    - DescribeContainer搭配使用 AWS SDK或 CLI
    - GetObject搭配使用 AWS SDK或 CLI
    - ListContainers搭配使用 AWS SDK或 CLI
    - PutObject搭配使用 AWS SDK或 CLI

使用的基本示 MediaStore 例 AWS SDKs

以下代码示例展示了如何使用以下基础知识 AWS Elemental MediaStore 替换为 AWS SDKs.

示例

- MediaStore 使用的操作 AWS SDKs
  - CreateContainer搭配使用 AWS SDK或 CLI
  - DeleteContainer搭配使用 AWS SDK或 CLI
  - DeleteObject搭配使用 AWS SDK或 CLI
  - DescribeContainer搭配使用 AWS SDK或 CLI
  - GetObject搭配使用 AWS SDK或 CLI
  - ListContainers搭配使用 AWS SDK或 CLI
  - PutObject搭配使用 AWS SDK或 CLI

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搭配使用 AWS SDK或 CLI
- DeleteContainer搭配使用 AWS SDK或 CLI
- DeleteObject搭配使用 AWS SDK或 CLI
- DescribeContainer搭配使用 AWS SDK或 CLI
- GetObject搭配使用 AWS SDK或 CLI
- ListContainers搭配使用 AWS SDK或 CLI
- PutObject搭配使用 AWS SDK或 CLI

`CreateContainer`搭配使用 AWS SDK或 CLI

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，请参阅中的创建容器 AWS 元素 MediaStore 用户指南。

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

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在中查找完整的示例，学习如何设置和运行 AWS 代码示例存储库。

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

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

DeleteContainer搭配使用 AWS SDK或 CLI

以下代码示例演示如何使用 DeleteContainer。

CLI

AWS CLI

删除容器

以下delete-container示例删除了指定的容器。您只能在容器没有对象时将其删除。

```
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《中删除容器》AWS 元素 MediaStore 用户指南。

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

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在中查找完整的示例，学习如何设置和运行 AWS 代码示例存储库。

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

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

`DeleteObject`搭配使用 AWS SDK或 CLI

以下代码示例显示了如何使用`DeleteObject`。

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在中查找完整的示例，学习如何设置和运行 AWS 代码示例存储库。

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

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

`DescribeContainer`搭配使用 AWS SDK或 CLI

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`示例显示了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，请参阅中查看容器的详细信息 AWS 元素 MediaStore 用户指南。

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

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在中查找完整的示例，学习如何设置和运行 AWS 代码示例存储库。

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

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

`GetObject`搭配使用 AWS SDK或 CLI

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的终端节点。

```
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的终端节点。

```
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，请参阅中的下载对象 AWS 元素 MediaStore 用户指南。

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

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在中查找完整的示例，学习如何设置和运行 AWS 代码示例存储库。

```
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详细信息，请参阅GetObject中的 AWS SDK for Java 2.x API参考。

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

`ListContainers`搭配使用 AWS SDK或 CLI

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`示例显示了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《查看容器列表》AWS 元素 MediaStore 用户指南。

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

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在中查找完整的示例，学习如何设置和运行 AWS 代码示例存储库。

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

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

`PutObject`搭配使用 AWS SDK或 CLI

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的容器。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 将对象 MediaStore 存储在文件夹中。如果该文件夹不存在，则服务会创建该文件夹，然后将该对象存储在文件夹中。

```
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，请参阅中的上传对象 AWS 元素 MediaStore 用户指南。

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

Java

SDK适用于 Java 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在中查找完整的示例，学习如何设置和运行 AWS 代码示例存储库。

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

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

AWS Elemental MediaStore 的配额

“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 Elemental MediaStore 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	注释
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	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	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	注释
标准上传可用性的 GetObject 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。
流式上传可用性的 GetObject API 请求的速率	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。
分块传输编码的 PutObject API 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 PutObject 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。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 MediaStore 相关信息

下表列出了在您使用 AWS Elemental MediaStore 时可提供帮助的相关资源。

- 课程和研讨会 – 指向基于角色的专业课程和自主进度动手实验室的链接，这些课程和实验室旨在帮助您增强 AWS 技能并获得实践经验。
- AWS 开发人员中心 – 浏览教程、下载工具并了解 AWS 开发人员活动。
- AWS 开发人员工具 – 指向开发人员工具、开发工具包、IDE 工具包和命令行工具的链接，这些资源用于开发和管理 AWS 应用程序。
- 入门资源中心 – 了解如何设置 AWS 账户、加入 AWS 社区和启动您的第一个应用程序。
- 动手实践教程 – 按照分步教程在 AWS 上启动您的第一个应用程序。
- AWS 白皮书 – 指向 AWS 技术白皮书的完整列表的链接，这些资料涵盖了架构、安全性、经济性等主题，由 AWS 解决方案架构师或其他技术专家编写。
- AWS Support 中心 – 用于创建和管理 AWS Support 案例的中心。还提供指向其他有用资源的链接，如论坛、技术常见问题、服务运行状况以及AWS Trusted Advisor。
- AWS Support – 提供有关 AWS Support 的信息的主要网页，这是一个一对一的快速响应支持渠道，可以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、您的账户、许可、网站访问和其他主题的详细信息。

用户指南文档历史记录

下表介绍此 AWS Elemental MediaStore 版本的文档。如需对此文档更新的通知，您可以订阅 RSS 源。

变更	说明	日期
源站访问控制 (OAC) 改进	增加了有关如何将 OAC 与 AWS Elemental MediaStore 结合使用的信息。	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 字段	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
支持 Amazon CloudWatch 指标	您可以设置指标策略来指示 MediaStore 向 CloudWatch 发送哪些指标。	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	2019 年 12 月 20 日

	您希望服务在一定天数后删除的文件。	
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
AWS CloudFormation 支持	现在，您可以使用 AWS CloudFormation 模板自动创建容器。该 AWS 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 现在支持访问日志记录，它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 集成	已更新 CloudTrail 集成内容，以便符合对 CloudTrail 服务的最新更改。	2018 年 7 月 12 日
CDN 协作	增加了有关如何结合使用 AWS Elemental MediaStore 和内容分发网络 (CDN)（如 Amazon CloudFront）的信息。	2018 年 4 月 14 日
CORS 配置	AWS Elemental MediaStore 现在支持跨源资源共享 (CORS)，跨源资源共享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。	2018 年 2 月 7 日
新服务和指南	这是视频创作和存储服务 AWS Elemental MediaStore 的初始版本和 AWS Elemental MediaStore 用户指南。	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 术语表

有关最新的 AWS 术语，请参阅《AWS 词汇表参考》中的 AWS 词汇表。

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。