aws

User Guide

Amazon S3 on Outposts



API Version 2006-03-01

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

Amazon S3 on Outposts: User Guide

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

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

Table of Contents

What is S3 on Outposts? ... 1
How S3 on Outposts works ... 1
Regions ... 2
Buckets ... 2
Objects ... 3
Keys ... 3
S3 Versioning ... 4
Version ID ... 4
Storage class and encryption ... 4
Bucket policy ... 4
S3 on Outposts access points ... 5
Features of S3 on Outposts ... 5
Access management ... 5
Storage logging and monitoring ... 6
Strong consistency ... 6
Related services ... 7
Accessing S3 on Outposts ... 7
AWS Management Console ... 7
AWS Command Line Interface ... 7
AWS SDKs ... 8
Paying for S3 on Outposts ... 8
Next steps ... 8
Setting up your Outpost ... 10
Order a new Outpost ... 10
How S3 on Outposts is different ... 11
Specifications ... 11
Supported API operations ... 12
Amazon S3 AWS CLI commands supported by S3 on Outposts ... 12
Unsupported Amazon S3 features ... 12
Network restrictions ... 13
Getting started with S3 on Outposts ... 15
Using the S3 console ... 15
Create a bucket, an access point, and an endpoint ... 16
Next steps ... 18

Using the AWS CLI and SDK for Java ... 18
Step 1: Create a bucket ... 19
Step 2: Create an access point ... 20
Step 3: Create an endpoint ... 20
Step 4: Upload an object to an S3 on Outposts bucket ... 21
Networking for S3 on Outposts ... 22
Choosing your networking access type ... 22
Accessing your S3 on Outposts buckets and objects ... 22
Managing connections using cross-account elastic network interfaces ... 23
Working with S3 on Outposts buckets ... 24
Buckets ... 24
Access points ... 24
Endpoints ... 25
API operations on S3 on Outposts ... 25
Creating and managing S3 on Outposts buckets ... 27
Creating a bucket ... 27
Adding tags ... 30
Using bucket policies ... 32
Adding a bucket policy ... 32
Viewing a bucket policy ... 34
Deleting a bucket policy ... 36
Bucket policy examples ... 37
Listing buckets ... 41
Getting a bucket ... 42
Deleting your bucket ... 44
Working with access points ... 45
Creating an access point ... 46
Using a bucket-style alias for your access point ... 47
Viewing access point configuration ... 51
Listing access points ... 52
Deleting an access point ... 53
Adding an access point policy ... 54
Viewing an access point policy ... 56
Working with endpoints ... 57
Creating an endpoint ... 59
Listing endpoints ... 60

Deleting an endpoint ... 61
Working with S3 on Outposts objects ... 64
Upload an object ... 65
Copying an object ... 66
Using the AWS SDK for Java ... 67
Getting an object ... 68
Listing objects ... 71
Deleting objects ... 74
Using HeadBucket ... 78
Performing a multipart upload ... 80
Perform a multipart upload of an object in an S3 on Outposts bucket ... 81
Copy a large object in an S3 on Outposts bucket by using multipart upload ... 83
List parts of an object in an S3 on Outposts bucket ... 85
Retrieve a list of in-progress multipart uploads in an S3 on Outposts bucket ... 86
Using presigned URLs ... 88
Limiting presigned URL capabilities ... 88
Who can create a presigned URL ... 90
When does S3 on Outposts check the expiration date and time of a presigned URL? ... 91
Sharing objects ... 91
Uploading an object ... 96
Amazon S3 on Outposts with local Amazon EMR ... 101
Creating an Amazon S3 on Outposts bucket ... 102
Getting started using Amazon EMR with Amazon S3 on Outposts ... 103
Authorization and authentication caching ... 108
Configuring the authorization and authentication cache ... 109
Validating SigV4A signing ... 109
Security ... 110
Setting up IAM ... 110
Principals for S3 on Outposts policies ... 113
ARNs for S3 on Outposts ... 113
Example policies for S3 on Outposts ... 115
Permissions for endpoints ... 115
Service-linked roles for S3 on Outposts ... 117
Data encryption ... 118
AWS PrivateLink for S3 on Outposts ... 118
Restrictions and limitations ... 119

Accessing S3 on Outposts interface endpoints ... 120
Updating an on-premises DNS configuration ... 122
Creating a VPC endpoint ... 122
Creating VPC endpoint policies and bucket policies ... 122
Signature Version 4 (SigV4) policy keys ... 124
Bucket policy examples that use Signature Version 4-related condition keys ... 126
AWS managed policies ... 128
AWSS3OnOutpostsServiceRolePolicy ... 129
Policy updates ... 129
Using service-linked roles ... 130
Service-linked role permissions for S3 on Outposts ... 130
Creating a service-linked role for S3 on Outposts ... 133
Editing a service-linked role for S3 on Outposts ... 133
Deleting a service-linked role for S3 on Outposts ... 134
Supported Regions for S3 on Outposts service-linked roles ... 134
Managing S3 on Outposts storage ... 135
Managing S3 Versioning ... 135
Creating and managing a lifecycle configuration ... 137
Using the console ... 138
Using the AWS CLI and SDK for Java ... 141
Replicating objects for S3 on Outposts ... 145
Replication configuration ... 146
Requirements for S3 Replication on Outposts ... 147
What is replicated? ... 148
What isn't replicated? ... 148
What isn't supported by S3 Replication on Outposts? ... 149
Setting up replication ... 149
Managing your replication ... 168
Sharing S3 on Outposts ... 176
Prerequisites ... 176
Procedure ... 177
Usage examples ... 178
Other services ... 180
Monitoring S3 on Outposts ... 182
CloudWatch metrics ... 182
CloudWatch metrics ... 183

Amazon CloudWatch Events ... 184
CloudTrail logs ... 186
Enabling CloudTrail logging for S3 on Outposts objects ... 186
Amazon S3 on Outposts AWS CloudTrail log file entries ... 189
Developing with S3 on Outposts ... 192
Supported regions ... 192
S3 on Outposts APIs ... 193
Amazon S3 API operations for managing objects ... 193
Amazon S3 Control API operations for managing buckets ... 194
S3 on Outposts API operations for managing Outposts ... 195
Configuring S3 control client ... 196
Making requests over IPv6 ... 196
Getting started with IPv6 ... 197
Making requests using dual-stack endpoints ... 198
Using IPv6 addresses in IAM policies ... 198
Testing IP address compatibility ... 200
Using IPv6 with AWS PrivateLink ... 200
Using dual-stack endpoints ... 203

What is Amazon S3 on Outposts?

AWS Outposts is a fully managed service that offers the same AWS infrastructure, AWS services, APIs, and tools to virtually any data center, co-location space, or on-premises facility for a truly consistent hybrid experience. AWS Outposts is ideal for workloads that require low-latency access to on-premises systems, local data processing, data residency, and migration of applications with local system interdependencies. For more information, see What is AWS Outposts? in the *AWS Outposts User Guide*.

With Amazon S3 on Outposts, you can create S3 buckets on your Outposts and easily store and retrieve objects on premises. S3 on Outposts provides a new storage class, OUTPOSTS, which uses the Amazon S3 APIs and is designed to store data durably and redundantly across multiple devices and servers on your Outposts. You communicate with your Outposts bucket by using an access point and endpoint connection over a virtual private cloud (VPC).

You can use the same APIs and features on Outposts buckets as you do on Amazon S3, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API.

- How S3 on Outposts works
- Features of S3 on Outposts
- Related services
- Accessing S3 on Outposts
- Paying for S3 on Outposts
- Next steps

How S3 on Outposts works

S3 on Outposts is an object storage service that stores data as objects within buckets on your Outpost. An *object* is a data file and any metadata that describes the file. A *bucket* is a container for objects.

To store your data in S3 on Outposts, you first create a bucket. When you create the bucket, you specify a bucket name and the Outpost that will hold the bucket. To access your S3 on Outposts bucket and perform object operations, you next create and configure an access point. You must also create an endpoint to route requests to your access point.

Access points simplify data access for any AWS service or customer application that stores data in S3. Access points are named network endpoints that are attached to buckets and can be used to perform object operations, such as `GetObject` and `PutObject`. Each access point has distinct permissions and network controls.

You can create and manage your S3 on Outposts buckets, access points, and endpoints by using the AWS Management Console, AWS CLI, AWS SDKs, or REST API. To upload and manage objects in your S3 on Outposts bucket, you can use the AWS CLI, AWS SDKs, or REST API.

Regions

During AWS Outposts provisioning, you or AWS creates a service link connection that connects your Outpost back to your chosen AWS Region or Outposts home Region for bucket operations and telemetry. An Outpost relies on connectivity to the parent AWS Region. The Outposts rack is not designed for disconnected operations or environments with limited to no connectivity. For more information, see Outpost connectivity to AWS Regions in the *AWS Outposts User Guide*.

Buckets

A bucket is a container for objects stored in S3 on Outposts. You can store any number of objects in a bucket and can have up to 100 buckets per account per Outpost.

When you create a bucket, you enter a bucket name and choose the Outpost where the bucket will reside. After you create a bucket, you cannot change the bucket name or move the bucket to a different Outpost. Bucket names must follow Amazon S3 bucket naming rules. In S3 on Outposts, bucket names are unique to an Outpost and AWS account. S3 on Outposts buckets require the `outpost-id`, `account-id`, and bucket name to identify them.

The following example shows the Amazon Resource Name (ARN) format for S3 on Outposts buckets. The ARN is comprised of the Region your Outpost is homed to, your Outpost account, the Outpost ID, and the bucket name.

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/bucket/bucket-name
```

Every object is contained in a bucket. You must use access points to access any object in an Outposts bucket. When you specify the bucket for object operations, you use the access point ARN or access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the access point ARN format for S3 on Outposts, which includes the `outpost-id`, `account-id`, and access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about buckets, see Working with S3 on Outposts buckets.

Objects

Objects are the fundamental entities stored in S3 on Outposts. Objects consist of object data and metadata. The metadata is a set of name-value pairs that describe the object. These pairs include some default metadata, such as the date last modified, and standard HTTP metadata, such as `Content-Type`. You can also specify custom metadata at the time that the object is stored. An object is uniquely identified within a bucket by a key (or name).

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

Keys

An *object key* (or *key name*) is the unique identifier for an object within a bucket. Every object in a bucket has exactly one key. The combination of a bucket and object key uniquely identifies each object.

The following example shows the ARN format for S3 on Outposts objects, which includes the AWS Region code for the Region that the Outpost is homed to, AWS account ID, Outpost ID, bucket name, and object key:

```
arn:aws:s3-outposts:us-west-2:123456789012:outpost/ op-01ac5d28a6a232904/bucket/amzn-s3-demo-bucket1/object/myobject
```

For more information about object keys, see Working with S3 on Outposts objects.

S3 Versioning

You can use S3 Versioning on Outposts buckets to keep multiple variants of an object in the same bucket. With S3 Versioning, you can preserve, retrieve, and restore every version of every object stored in your buckets. S3 Versioning helps you recover from unintended user actions and application failures.

For more information, see Managing S3 Versioning for your S3 on Outposts bucket.

Version ID

When you enable S3 Versioning in a bucket, S3 on Outposts generates a unique version ID for each object added to the bucket. Objects that already existed in the bucket at the time that you enable versioning have a version ID of `null`. If you modify these (or any other) objects with other operations, such as PutObject, the new objects get a unique version ID.

For more information, see Managing S3 Versioning for your S3 on Outposts bucket.

Storage class and encryption

S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS). The S3 Outposts storage class is available only for objects stored in buckets on AWS Outposts. If you try to use other S3 storage classes with S3 on Outposts, S3 on Outposts returns the `InvalidStorageClass` error.

By default, objects stored in the S3 Outposts (OUTPOSTS) storage class are encrypted using server-side encryption with Amazon S3 managed encryption keys (SSE-S3). For more information, see Data encryption in S3 on Outposts.

Bucket policy

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size.

Bucket policies use JSON-based IAM policy language that is standard across AWS. You can use bucket policies to add or deny permissions for the objects in a bucket. Bucket policies allow or deny requests based on the elements in the policy. These elements can include the requester, S3 on Outposts actions, resources, and aspects or conditions of the request (for example, the IP address

used to make the request). For example, you can create a bucket policy that grants cross-account permissions to upload objects to an S3 on Outposts bucket while ensuring that the bucket owner has full control of the uploaded objects.

In your bucket policy, you can use wildcard characters (*) in ARNs and other values to grant permissions to a subset of objects. For example, you can control access to groups of objects that begin with a common prefix or end with a given extension, such as `.html`.

S3 on Outposts access points

S3 on Outposts access points are named network endpoints with dedicated access policies that describe how data can be accessed using that endpoint. Access points simplify managing data access at scale for shared datasets in S3 on Outposts. Access points are attached to buckets that you can use to perform S3 object operations, such as `GetObject` and `PutObject`.

When you specify the bucket for object operations, you use the access point ARN or access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

Access points have distinct permissions and network controls that S3 on Outposts applies for any request that is made through that access point. Each access point enforces a customized access point policy that works in conjunction with the bucket policy that is attached to the underlying bucket.

For more information, see Accessing your S3 on Outposts buckets and objects.

Features of S3 on Outposts

Access management

S3 on Outposts provides features for auditing and managing access to your buckets and objects. By default, S3 on Outposts buckets and the objects in them are private. You have access only to the S3 on Outposts resources that you create.

To grant granular resource permissions that support your specific use case or to audit the permissions of your S3 on Outposts resources, you can use the following features.

- S3 Block Public Access – Block public access to buckets and objects. For buckets on Outposts, Block Public Access is always enabled by default.

- AWS Identity and Access Management (IAM) – IAM is a web service that helps you securely control access to AWS resources, including your S3 on Outposts resources. With IAM, you can centrally manage permissions that control which AWS resources users can access. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
- S3 on Outposts access points – Manage data access for shared datasets in S3 on Outposts. Access points are named network endpoints with dedicated access policies. Access points are attached to buckets and can be used to perform object operations, such as `GetObject` and `PutObject`.
- Bucket policies – Use IAM-based policy language to configure resource-based permissions for your S3 buckets and the objects in them.
- AWS Resource Access Manager (AWS RAM) – Securely share your S3 on Outposts capacity across AWS accounts, within your organization or organizational units (OUs) in AWS Organizations.

Storage logging and monitoring

S3 on Outposts provides logging and monitoring tools that you can use to monitor and control how your S3 on Outposts resources are being used. For more information, see Monitoring tools.

- Amazon CloudWatch metrics for S3 on Outposts – Track the operational health of your resources and understand your capacity availability.
- Amazon CloudWatch Events events for S3 on Outposts – Create a rule for any S3 on Outposts API event to receive notifications through all supported CloudWatch Events targets, including Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), and AWS Lambda.
- AWS CloudTrail logs for S3 on Outposts – Record actions taken by a user, a role, or an AWS service in S3 on Outposts. CloudTrail logs provide you with detailed API tracking for S3 bucket-level and object-level operations.

Strong consistency

S3 on Outposts provides strong read-after-write consistency for PUT and DELETE requests of objects in your S3 on Outposts bucket in all AWS Regions. This behavior applies to both writes of new objects and to PUT requests that overwrite existing objects and to DELETE requests. In addition, S3 on Outposts object tags and object metadata (for example, the HEAD object) are strongly consistent. For more information, see Amazon S3 data consistency model in the *Amazon S3 User Guide*.

Related services

After you load your data into S3 on Outposts, you can use it with other AWS services. The following are the services that you might use most frequently:

- Amazon Elastic Compute Cloud (Amazon EC2) – Provides secure and scalable computing capacity in the AWS Cloud. Using Amazon EC2 lessens your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.
- Amazon Elastic Block Store (Amazon EBS) on Outposts – Use Amazon EBS local snapshots on Outposts to store snapshots of volumes on an Outpost locally in S3 on Outposts.
- Amazon Relational Database Service (Amazon RDS) on Outposts – Use Amazon RDS local backups to store your Amazon RDS backups locally on your Outpost.
- AWS DataSync – Automate transferring data between your Outposts and AWS Regions, choosing what to transfer, when to transfer, and how much network bandwidth to use. S3 on Outposts is integrated with AWS DataSync. For on-premises applications that require high-throughput local processing, S3 on Outposts provides on-premises object storage to minimize data transfers and buffer from network variations, while providing you the ability to easily transfer data between Outposts and AWS Regions.

Accessing S3 on Outposts

You can work with S3 on Outposts in any of the following ways:

AWS Management Console

The console is a web-based user interface for managing S3 on Outposts and AWS resources. If you've signed up for an AWS account, you can access S3 on Outposts by signing into the AWS Management Console and choosing **S3** from the AWS Management Console home page. Then, choose **Outposts buckets** from the left navigation pane.

AWS Command Line Interface

You can use the AWS command line tools to issue commands or build scripts at your system's command line to perform AWS (including S3) tasks.

The AWS Command Line Interface (AWS CLI) provides commands for a broad set of AWS services. The AWS CLI is supported on Windows, macOS, and Linux. To get started, see the AWS Command Line Interface User Guide. For more information about the commands that you can use with S3 on Outposts, see s3api, s3control, and s3outposts in the *AWS CLI Command Reference*.

AWS SDKs

AWS provides SDKs (software development kits) that consist of libraries and sample code for various programming languages and platforms (Java, Python, Ruby, .NET, iOS, Android, and so on). The AWS SDKs provide a convenient way to create programmatic access to S3 on Outposts and AWS. Because S3 on Outposts uses the same SDKs as Amazon S3, S3 on Outposts provides a consistent experience using the same S3 APIs, automation, and tools.

S3 on Outposts is a REST service. You can send requests to S3 on Outposts by using the AWS SDK libraries, which wrap the underlying REST API and simplify your programming tasks. For example, the SDKs take care of tasks such as calculating signatures, cryptographically signing requests, managing errors, and retrying requests automatically. For information about the AWS SDKs, including how to download and install them, see Tools to Build on AWS.

Paying for S3 on Outposts

You can purchase a variety of AWS Outposts rack configurations featuring a combination of Amazon EC2 instance types, Amazon EBS General Purpose solid state drive (SSD) volumes (gp2), and S3 on Outposts. Pricing includes delivery, installation, infrastructure service maintenance, and software patches and upgrades.

For more information, see AWS Outposts rack pricing.

Next steps

For more information about working with S3 on Outposts, see the following topics:

- Setting up your Outpost
- How is Amazon S3 on Outposts different from Amazon S3?
- Getting started with Amazon S3 on Outposts
- Networking for S3 on Outposts
- Working with S3 on Outposts buckets

- Working with S3 on Outposts objects
- Security in S3 on Outposts
- Managing S3 on Outposts storage
- Developing with Amazon S3 on Outposts

Setting up your Outpost

To get started with Amazon S3 on Outposts, you will need an Outpost with Amazon S3 capacity deployed at your facility. For information about options for ordering an Outpost and S3 capacity, see AWS Outposts. To check if your Outposts has S3 capacity on it, you can use the ListOutpostsWithS3 API call. For specifications and to see how S3 on Outposts is different than Amazon S3, see How is Amazon S3 on Outposts different from Amazon S3?

For more information, see the following topics.

Topics

- Order a new Outpost

Order a new Outpost

If you need to order a new Outpost with S3 capacity, see AWS Outposts rack pricing to understand the capacity options for Amazon Elastic Compute Cloud (Amazon EC2), Amazon Elastic Block Store (Amazon EBS), and Amazon S3.

After you select your configuration, follow the steps in Create an Outpost and order Outpost capacity in the *AWS Outposts User Guide.*

How is Amazon S3 on Outposts different from Amazon S3?

Amazon S3 on Outposts delivers object storage to your on-premises AWS Outposts environment. Using S3 on Outposts helps you to meet local processing, data residency, and demanding performance needs by keeping data close to on-premises applications. Because it uses Amazon S3 APIs and features, S3 on Outposts makes it easy to store, secure, tag, report on, and control access to the data on your Outposts and extend AWS infrastructure to your on-premises facility for a consistent hybrid experience.

For more information about how S3 on Outposts is unique, see the following topics.

Topics

- S3 on Outposts specifications
- API operations supported by S3 on Outposts
- Amazon S3 AWS CLI commands supported by S3 on Outposts
- Amazon S3 features not supported by S3 on Outposts
- S3 on Outposts network requirements

S3 on Outposts specifications

- The maximum Outposts bucket size is 50 TB.
- The maximum object size is 5 TB in Outposts buckets.
- The maximum number of Outposts buckets is 100 per AWS account.
- Outposts buckets can be accessed only by using access points and endpoints.
- The maximum number of access points per Outposts bucket is 10.
- Access point policies are limited to 20 KB in size.
- The Outpost owner can manage access within your organization in AWS Organizations by using AWS Resource Access Manager. All accounts that need access to the Outpost must be within the same organization as the owner account in AWS Organizations.
- The S3 on Outposts bucket owner account is always the owner of all objects in the bucket.
- Only the S3 on Outposts bucket owner account can perform operations on the bucket.

- Object size limitations are consistent with Amazon S3.
- All objects stored on S3 on Outposts are stored in the OUTPOSTS storage class.
- By default, all objects stored in the OUTPOSTS storage class are stored by using server-side encryption with Amazon S3 managed encryption keys (SSE-S3). You can also explicitly choose to store objects by using server-side encryption with customer-provided encryption keys (SSE-C).
- If there is not enough space to store an object on your Outpost, the API returns an insufficient capacity exception (ICE).

API operations supported by S3 on Outposts

For a list of API operations supported by S3 on Outposts, see Amazon S3 on Outposts API operations.

Amazon S3 AWS CLI commands supported by S3 on Outposts

The following Amazon S3 AWS CLI commands are currently supported by Amazon S3 on Outposts. For more information, see Available Commands in the AWS CLI Command Reference.

- `cp`, `mv`, and `sync` within the same Outposts bucket, or between a local environment and an Outposts bucket.
- `ls`
- `presign`
- `rm`

Amazon S3 features not supported by S3 on Outposts

The following Amazon S3 features are currently not supported by Amazon S3 on Outposts. Any attempts to use them are rejected.

- Conditional requests
- Access control lists (ACLs)
- Cross-origin resource sharing (CORS)
- S3 Batch Operations
- S3 Inventory reports

- Changing the default bucket encryption
- Public buckets
- Multi-factor authentication (MFA) delete
- S3 Lifecycle transitions (aside from object deletion and stopping incomplete multipart uploads)
- S3 Object Lock legal hold
- Object Lock retention
- Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)
- S3 Replication Time Control (S3 RTC)
- Amazon CloudWatch request metrics
- Metrics configuration
- Transfer Acceleration
- S3 Event Notifications
- Requester Pays buckets
- S3 Select
- AWS Lambda events
- Server access logging
- HTTP POST requests
- SOAP
- Website access

S3 on Outposts network requirements

- To route requests to an S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. The following limits apply to endpoints for S3 on Outposts:
  - Each virtual private cloud (VPC) on an Outpost can have one associated endpoint, and you can have up to 100 endpoints per Outpost.
  - You can map multiple access points to the same endpoint.
  - You can add endpoints only to VPCs with CIDR blocks in the subspaces of the following CIDR ranges:
    - 10.0.0.0/8
    - 172.16.0.0/12

  - 192.168.0.0/16
- You can create endpoints to an Outpost only from VPCs that have non-overlapping CIDR blocks.
- You can create an endpoint only from within its Outposts subnet.
- The subnet that you use to create an endpoint must contain four IP addresses for S3 on Outposts to use.
- If you specify the customer-owned IP address pool (CoIP pool), it must contain four IP addresses for S3 on Outposts to use.
- You can create only one endpoint per Outpost per VPC.

Getting started with Amazon S3 on Outposts

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API.

With Amazon S3 on Outposts, you can use the Amazon S3 APIs and features, such as object storage, access policies, encryption, and tagging, on AWS Outposts as you do on Amazon S3. For information about S3 on Outposts, see What is Amazon S3 on Outposts?

Topics

- Getting started by using the AWS Management Console
- Getting started by using the AWS CLI and SDK for Java

Getting started by using the AWS Management Console

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

To get started with S3 on Outposts by using the console, see the following topics. To get started by using the AWS CLI or AWS SDK for Java, see Getting started by using the AWS CLI and SDK for Java.

Topics

- Create a bucket, an access point, and an endpoint
- Next steps

Create a bucket, an access point, and an endpoint

The following procedure shows you how to create your first bucket in S3 on Outposts. When you create a bucket using the console, you also create an access point and an endpoint associated with the bucket so that you can immediately begin storing objects in your bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose **Create Outposts bucket**.
4. For **Bucket name**, enter a Domain Name System (DNS)-compliant name for your bucket.

   The bucket name must:

   - Be unique within the AWS account, the Outpost, and the AWS Region that the Outpost is homed to.
   - Be 3–63 characters long.
   - Not contain uppercase characters.
   - Start with a lowercase letter or number.

   After you create the bucket, you can't change its name. For information about naming buckets, see General purpose bucket naming rules in the *Amazon S3 User Guide*.

   > **Important**
   >
   > Avoid including sensitive information such as account numbers in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

5. For **Outpost**, choose the Outpost where you want the bucket to reside.
6. Under **Bucket Versioning**, set the S3 Versioning state for your S3 on Outposts bucket to one of the following options:

   - **Disable** (default) – The bucket remains unversioned.

- **Enable** – Enables S3 Versioning for the objects in the bucket. All objects added to the bucket receive a unique version ID.

For more information about S3 Versioning, see Managing S3 Versioning for your S3 on Outposts bucket.

7. (Optional) Add any **optional tags** that you would like to associate with the Outposts bucket. You can use tags to track criteria for individual projects or groups of projects, or to label your buckets by using cost-allocation tags.

   By default, all objects stored in your Outposts bucket are stored by using server-side encryption with Amazon S3 managed encryption keys (SSE-S3). You can also explicitly choose to store objects by using server-side encryption with customer-provided encryption keys (SSE-C). To change the encryption type, you must use the REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs.

8. In the **Outposts access point settings** section, enter the access point name.

   S3 on Outposts access points simplify managing data access at scale for shared datasets in S3 on Outposts. Access points are named network endpoints that are attached to Outposts buckets that you can use to perform S3 object operations. For more information, see Access points.

   Access point names must be unique within the account for this Region and Outpost, and comply with the access point restrictions and limitations.

9. Choose the **VPC** for this Amazon S3 on Outposts access point.

   If you don't have a VPC, choose **Create VPC**. For more information, see Creating access points restricted to a virtual private cloud (VPC) in the *Amazon S3 User Guide*.

   A virtual private cloud (VPC) enables you to launch AWS resources into a virtual network that you define. This virtual network closely resembles a traditional network that you would operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

10. (Optional for an existing VPC) Choose an **Endpoint subnet** for your endpoint.

    A subnet is a range of IP addresses in your VPC. If you don't have the subnet that you want, choose **Create subnet**. For more information, see Networking for S3 on Outposts.

11. (Optional for an existing VPC) Choose an **Endpoint security group** for your endpoint.

A security group acts as a virtual firewall to control inbound and outbound traffic.

12. (Optional for an existing VPC) Choose the **Endpoint access type**:

    - **Private** – To be used with the VPC.
    - **Customer owned IP** – To be used with a customer-owned IP address pool (CoIP pool) from within your on-premises network.

13. (Optional) Specify the **Outpost access point policy**. The console automatically displays the **Amazon Resource Name (ARN)** for the access point, which you can use in the policy.
14. Choose **Create Outposts bucket**.

> **Note**
>
> It can take up to 5 minutes for your Outpost endpoint to be created and your bucket to be ready to use. To configure additional bucket settings, choose **View details**.

Next steps

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

After you create an S3 on Outposts bucket, access point, and endpoint, you can use the AWS CLI or SDK for Java to upload an object to your bucket. For more information, see Upload an object to an S3 on Outposts bucket.

Getting started by using the AWS CLI and SDK for Java

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with

your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

To get started with S3 on Outposts, you must create a bucket, an access point, and an endpoint. Then, you can upload objects to your bucket. The following examples show you how to get started with S3 on Outposts by using the AWS CLI and SDK for Java. To get started by using the console, see Getting started by using the AWS Management Console.

Topics

- Step 1: Create a bucket
- Step 2: Create an access point
- Step 3: Create an endpoint
- Step 4: Upload an object to an S3 on Outposts bucket

Step 1: Create a bucket

The following AWS CLI and SDK for Java examples show you how to create an S3 on Outposts bucket.

AWS CLI

Example

The following example creates an S3 on Outposts bucket (`s3-outposts:CreateBucket`) by using the AWS CLI. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-bucket --bucket example-outposts-bucket --outpost-
id op-01ac5d28a6a232904
```

SDK for Java

Example

For examples of how to create an S3 Outposts bucket with the AWS SDK for Java, see CreateOutpostsBucket.java in the *AWS SDK for Java 2.x Code Examples*.

Step 2: Create an access point

To access your Amazon S3 on Outposts bucket, you must create and configure an access point. These examples how you how to create an access point by using the AWS CLI and the SDK for Java.

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

AWS CLI

Example

The following AWS CLI example creates an access point for an Outposts bucket. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-access-point --account-id 123456789012
 --name example-outposts-access-point --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket" --vpc-configuration VpcId=example-vpc-12345
```

SDK for Java

Example

For examples of how to create an access point for an S3 Outposts bucket with the AWS SDK for Java, see CreateOutpostsAccessPoint.java in the *AWS SDK for Java 2.x Code Examples*.

Step 3: Create an endpoint

To route requests to an Amazon S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. In order to create an endpoint, you will need an active connection with your service link to your Outposts home region. Each virtual private cloud (VPC) on your Outpost can have one associated endpoint. For more information about endpoint quotas, see S3 on Outposts network requirements. You must create an endpoint to be able to access your Outposts buckets and perform object operations. For more information, see Endpoints.

These examples show you how to create an endpoint by using the AWS CLI and the SDK for Java. For more information about the permissions required to create and manage endpoints, see Permissions for S3 on Outposts endpoints.

AWS CLI

Example

The following AWS CLI example creates an endpoint for an Outpost by using the VPC resource access type. The VPC is derived from the subnet. To run this command, replace the *user input placeholders* with your own information.

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1
```

The following AWS CLI example creates an endpoint for an Outpost by using the customer-owned IP address pool (CoIP pool) access type. To run this command, replace the *user input placeholders* with your own information.

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1 --access-type CustomerOwnedIp --
customer-owned-ipv4-pool ipv4pool-coip-12345678901234567
```

SDK for Java

Example

For examples of how to create an endpoint for an S3 Outpost with the AWS SDK for Java, see CreateOutpostsEndPoint.java in the *AWS SDK for Java 2.x Code Examples*.

Step 4: Upload an object to an S3 on Outposts bucket

To upload an object, see Upload an object to an S3 on Outposts bucket.

Networking for S3 on Outposts

You can use Amazon S3 on Outposts to store and retrieve objects on-premises for applications that require local data access, data processing, and data residency. This section describes the networking requirements for accessing S3 on Outposts.

Topics

- Choosing your networking access type
- Accessing your S3 on Outposts buckets and objects
- Cross-account elastic network interfaces

Choosing your networking access type

You can access S3 on Outposts from within a VPC or from your on-premises network. You communicate with your Outpost bucket by using an access point and endpoint connection. This connection keeps traffic between your VPC and your S3 on Outposts buckets within the AWS network. When you create an endpoint, you must specify the endpoint access type as either `Private` (for VPC routing) or `CustomerOwnedIp` (for a customer-owned IP address pool [CoIP pool]).

- `Private` (for VPC routing) – If you don't specify the access type, S3 on Outposts uses `Private` by default. With the `Private` access type, instances in your VPC don't require public IP addresses to communicate with resources in your Outpost. You can work with S3 on Outposts from within a VPC. This type of endpoint is accessible from your on-premises network through direct VPC routing. For more information, see Local gateway route tables in the *AWS Outposts User Guide*.
- `CustomerOwnedIp` (for CoIP pool) – If you don't default to the `Private` access type and choose `CustomerOwnedIp`, you must specify an IP address range. You can use this access type to work with S3 on Outposts from both your on-premises network and within a VPC. When accessing S3 on Outposts within a VPC, your traffic is limited to the bandwidth of the local gateway.

Accessing your S3 on Outposts buckets and objects

To access your S3 on Outposts buckets and objects, you must have the following:

- An access point for the VPC.

- An endpoint for the same VPC.
- An active connection between your Outpost and your AWS Region. For more information about how to connect your Outpost to a Region, see Outpost connectivity to AWS Regions in the *AWS Outposts User Guide*.

For more information about accessing buckets and objects in S3 on Outposts, see Working with S3 on Outposts buckets and Working with S3 on Outposts objects.

Cross-account elastic network interfaces

S3 on Outposts endpoints are named resources with Amazon Resource Names (ARNs). When these endpoints are created, AWS Outposts sets up multiple cross-account elastic network interfaces. S3 on Outposts cross-account elastic network interfaces are like other network interfaces with one exception: S3 on Outposts associates the cross-account elastic network interfaces to Amazon EC2 instances.

The S3 on Outposts Domain Name System (DNS) load balances your requests over the cross-account elastic network interface. S3 on Outposts creates the cross-account elastic network interface in your AWS account that is visible from the **Network interfaces** pane of the Amazon EC2 console.

For endpoints that use the CoIP pool access type, S3 on Outposts allocates and associates IP addresses with the cross-account elastic network interface from the configured CoIP pool.

Working with S3 on Outposts buckets

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You can use the same APIs and features on Outpost buckets as you do on Amazon S3, including access policies, encryption, and tagging. For more information, see What is Amazon S3 on Outposts?

You communicate with your Outpost buckets by using an access point and endpoint connection over a virtual private cloud (VPC). To access your S3 on Outposts buckets and objects, you must have an access point for the VPC and an endpoint for the same VPC. For more information, see Networking for S3 on Outposts.

Buckets

In S3 on Outposts, bucket names are unique to an Outpost and require the AWS Region code for the Region the Outpost is homed to, AWS account ID, Outpost ID, and the bucket name to identify them.

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/bucket/bucket-name
```

For more information, see Resource ARNs for S3 on Outposts.

Access points

Amazon S3 on Outposts supports virtual private cloud (VPC)-only access points as the only means to access your Outposts buckets.

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

The following example shows the ARN format that you use for S3 on Outposts access points. The access point ARN includes the AWS Region code for the Region the Outpost is homed to, AWS account ID, Outpost ID, and access point name.

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

Endpoints

To route requests to an S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. With S3 on Outposts endpoints, you can privately connect your VPC to your Outpost bucket. S3 on Outposts endpoints are virtual uniform resource identifiers (URIs) of the entry point to your S3 on Outposts bucket. They are horizontally scaled, redundant, and highly available VPC components.

Each virtual private cloud (VPC) on your Outpost can have one associated endpoint, and you can have up to 100 endpoints per Outpost. You must create these endpoints to be able to access your Outpost buckets and perform object operations. Creating these endpoints also enables the API model and behaviors to be the same by allowing the same operations to work in S3 and S3 on Outposts.

API operations on S3 on Outposts

To manage Outpost bucket API operations, S3 on Outposts hosts a separate endpoint that is distinct from the Amazon S3 endpoint. This endpoint is `s3-outposts.region.amazonaws.com`.

To use Amazon S3 API operations, you must sign the bucket and objects using the correct ARN format. You must pass ARNs to API operations so that Amazon S3 can determine whether the request is for Amazon S3 (`s3-control.region.amazonaws.com`) or for S3 on Outposts (`s3-outposts.region.amazonaws.com`). Based on the ARN format, S3 can then sign and route the request appropriately.

Whenever a request is sent to the Amazon S3 control plane, the SDK extracts the components from the ARN and includes the additional header `x-amz-outpost-id`, with the *outpost-id* value extracted from the ARN. The service name from the ARN is used to sign the request before it is routed to the S3 on Outposts endpoint. This behavior applies to all API operations handled by the `s3control` client.

The following table lists the extended API operations for Amazon S3 on Outposts and their changes relative to Amazon S3.

API	S3 on Outposts parameter value	
`CreateBucket`	Bucket name as ARN, Outpost ID	
`ListRegionalBuckets`	Outpost ID	
`DeleteBucket`	Bucket name as ARN	
`DeleteBucketLifecycleConfiguration`	Bucket name as ARN	
`GetBucketLifecycleConfiguration`	Bucket name as ARN	
`PutBucketLifecycleConfiguration`	Bucket name as ARN	
`GetBucketPolicy`	Bucket name as ARN	
`PutBucketPolicy`	Bucket name as ARN	
`DeleteBucketPolicy`	Bucket name as ARN	
`GetBucketTagging`	Bucket name as ARN	
`PutBucketTagging`	Bucket name as ARN	
`DeleteBucketTagging`	Bucket name as ARN	
`CreateAccessPoint`	Access point name as ARN	
`DeleteAccessPoint`	Access point name as ARN	
`GetAccessPoint`	Access point name as ARN	
`GetAccessPoint`	Access point name as ARN	
`ListAccessPoints`	Access point name as ARN	

API	S3 on Outposts parameter value
PutAccessPointPolicy	Access point name as ARN
GetAccessPointPolicy	Access point name as ARN
DeleteAccessPointPolicy	Access point name as ARN

Creating and managing S3 on Outposts buckets

For more information about creating and managing S3 on Outposts buckets, see the following topics.

Creating an S3 on Outposts bucket

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

Note

The AWS account that creates the bucket owns it and is the only one that can commit actions to it. Buckets have configuration properties, such as Outpost, tag, default encryption, and access point settings. The access point settings include the virtual private cloud (VPC), the access point policy for accessing the objects in the bucket, and other metadata. For more information, see S3 on Outposts specifications.

If you want to create a bucket that uses AWS PrivateLink to provide bucket and endpoint management access through *interface VPC endpoints* in your virtual private cloud (VPC), see AWS PrivateLink for S3 on Outposts.

The following examples show you how to create an S3 on Outposts bucket by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose **Create Outposts bucket**.
4. For **Bucket name**, enter a Domain Name System (DNS)-compliant name for your bucket.

   The bucket name must:

   - Be unique within the AWS account, the Outpost, and the AWS Region that the Outpost is homed to.
   - Be 3–63 characters long.
   - Not contain uppercase characters.
   - Start with a lowercase letter or number.

   After you create the bucket, you can't change its name. For information about naming buckets, see General purpose bucket naming rules in the *Amazon S3 User Guide*.

   > **Important**
   >
   > Avoid including sensitive information such as account numbers in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

5. For **Outpost**, choose the Outpost where you want the bucket to reside.
6. Under **Bucket Versioning**, set the S3 Versioning state for your S3 on Outposts bucket to one of the following options:

   - **Disable** (default) – The bucket remains unversioned.
   - **Enable** – Enables S3 Versioning for the objects in the bucket. All objects added to the bucket receive a unique version ID.

   For more information about S3 Versioning, see Managing S3 Versioning for your S3 on Outposts bucket.

7. (Optional) Add any **optional tags** that you would like to associate with the Outposts bucket. You can use tags to track criteria for individual projects or groups of projects, or to label your buckets by using cost-allocation tags.

   By default, all objects stored in your Outposts bucket are stored by using server-side encryption with Amazon S3 managed encryption keys (SSE-S3). You can also explicitly choose to store objects by using server-side encryption with customer-provided encryption keys (SSE-C). To change the encryption type, you must use the REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs.

8. In the **Outposts access point settings** section, enter the access point name.

   S3 on Outposts access points simplify managing data access at scale for shared datasets in S3 on Outposts. Access points are named network endpoints that are attached to Outposts buckets that you can use to perform S3 object operations. For more information, see Access points.

   Access point names must be unique within the account for this Region and Outpost, and comply with the access point restrictions and limitations.

9. Choose the **VPC** for this Amazon S3 on Outposts access point.

   If you don't have a VPC, choose **Create VPC**. For more information, see Creating access points restricted to a virtual private cloud (VPC) in the *Amazon S3 User Guide*.

   A virtual private cloud (VPC) enables you to launch AWS resources into a virtual network that you define. This virtual network closely resembles a traditional network that you would operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

10. (Optional for an existing VPC) Choose an **Endpoint subnet** for your endpoint.

    A subnet is a range of IP addresses in your VPC. If you don't have the subnet that you want, choose **Create subnet**. For more information, see Networking for S3 on Outposts.

11. (Optional for an existing VPC) Choose an **Endpoint security group** for your endpoint.

    A security group acts as a virtual firewall to control inbound and outbound traffic.

12. (Optional for an existing VPC) Choose the **Endpoint access type**:

    - **Private** – To be used with the VPC.
    - **Customer owned IP** – To be used with a customer-owned IP address pool (CoIP pool) from within your on-premises network.

13. (Optional) Specify the **Outpost access point policy**. The console automatically displays the **Amazon Resource Name (ARN)** for the access point, which you can use in the policy.
14. Choose **Create Outposts bucket**.

> **Note**
>
> It can take up to 5 minutes for your Outpost endpoint to be created and your bucket to be ready to use. To configure additional bucket settings, choose **View details**.

Using the AWS CLI

Example

The following example creates an S3 on Outposts bucket (`s3-outposts:CreateBucket`) by using the AWS CLI. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-bucket --bucket example-outposts-bucket --outpost-
id op-01ac5d28a6a232904
```

Using the AWS SDK for Java

Example

For examples of how to create an S3 Outposts bucket with the AWS SDK for Java, see CreateOutpostsBucket.java in the *AWS SDK for Java 2.x Code Examples*.

Adding tags for S3 on Outposts buckets

You can add tags for your Amazon S3 on Outposts buckets to track storage costs and other criteria for individual projects or groups of projects.

> **Note**
>
> The AWS account that creates the bucket owns it and is the only one that can change its tags.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket whose tags you want to edit.
4. Choose the **Properties** tab.
5. Under **Tags**, choose **Edit**.
6. Choose **Add new tag**, and enter the **Key** and optional **Value**.

   Add any tags that you would like to associate with an Outposts bucket to track other criteria for individual projects or groups of projects.
7. Choose **Save changes**.

Using the AWS CLI

The following AWS CLI example applies a tagging configuration to an S3 on Outposts bucket by using a JSON document in the current folder that specifies tags (*tagging.json*). To use this example, replace each *user input placeholder* with your own information.

```
aws s3control put-bucket-tagging --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --tagging file://tagging.json
```

tagging.json

```
{
   "TagSet": [
     {
       "Key": "organization",
       "Value": "marketing"
     }
   ]
}
```

The following AWS CLI example applies a tagging configuration to an S3 on Outposts bucket directly from the command line.

```
aws s3control put-bucket-tagging --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --tagging 'TagSet=[{Key=organization,Value=marketing}]'
```

For more information about this command, see put-bucket-tagging in the *AWS CLI Reference*.

Managing access to an Amazon S3 on Outposts bucket using a bucket policy

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size. For more information, see Bucket policy.

You can update your bucket policy to manage access to your Amazon S3 on Outposts bucket. For more information, see the following topics.

Topics

- Adding or editing a bucket policy for an Amazon S3 on Outposts bucket
- Viewing the bucket policy for your Amazon S3 on Outposts bucket
- Deleting the bucket policy for your Amazon S3 on Outposts bucket
- Bucket policy examples

Adding or editing a bucket policy for an Amazon S3 on Outposts bucket

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size. For more information, see Bucket policy.

The following topics show you how to update your Amazon S3 on Outposts bucket policy by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS SDK for Java.

Using the S3 console

To create or edit a bucket policy

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket whose bucket policy you want to edit.
4. Choose the **Permissions** tab.
5. In the **Outposts bucket policy** section, to create or edit new policy, choose **Edit**.

   You can now add or edit the S3 on Outposts bucket policy. For more information, see Setting up IAM with S3 on Outposts.

Using the AWS CLI

The following AWS CLI example puts a policy on an Outposts bucket.

1. Save the following bucket policy to a JSON file. In this example, the file is named `policy1.json`. Replace the *`user input placeholders`* with your own information.

   JSON

```
{
   "Version":"2012-10-17",
   "Id":"testBucketPolicy",
   "Statement":[
      {
         "Sid":"st1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"arn:aws:iam::123456789012:root"
         },
         "Action":[
            "s3-outposts:GetObject",
            "s3-outposts:PutObject",
            "s3-outposts:DeleteObject",
            "s3-outposts:ListBucket"
         ],
```

```
            "Resource":"arn:aws:s3-outposts:us-
east-1:123456789012:outpost/op-01ac5d28a6a232904/bucket/amzn-s3-demo-bucket"
         }
      ]
}
```

2. Submit the JSON file as part of the `put-bucket-policy` CLI command. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control put-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --policy file://policy1.json
```

Using the AWS SDK for Java

The following SDK for Java example puts a policy on an Outposts bucket.

```
import com.amazonaws.services.s3control.model.*;

public void putBucketPolicy(String bucketArn) {

    String policy = "{\"Version\":\"2012-10-17\",\"Id\":\"testBucketPolicy\",
\"Statement\":[{\"Sid\":\"st1\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"" +
 AccountId+ "\"},\"Action\":\"s3-outposts:*\",\"Resource\":\"" + bucketArn + "\"}]}";

    PutBucketPolicyRequest reqPutBucketPolicy = new PutBucketPolicyRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withPolicy(policy);

    PutBucketPolicyResult respPutBucketPolicy =
 s3ControlClient.putBucketPolicy(reqPutBucketPolicy);
    System.out.printf("PutBucketPolicy Response: %s%n",
 respPutBucketPolicy.toString());

}
```

Viewing the bucket policy for your Amazon S3 on Outposts bucket

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can

associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size. For more information, see Bucket policy.

The following topics show you how to view your Amazon S3 on Outposts bucket policy by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS SDK for Java.

Using the S3 console

To create or edit a bucket policy

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket whose permission you want to edit.
4. Choose the **Permissions** tab.
5. In the **Outposts bucket policy** section, you can review your existing bucket policy. For more information, see Setting up IAM with S3 on Outposts.

Using the AWS CLI

The following AWS CLI example gets a policy for an Outposts bucket. To run this command, replace the *user input placeholders* with your own information.

```
aws s3control get-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket
```

Using the AWS SDK for Java

The following SDK for Java example gets a policy for an Outposts bucket.

```
import com.amazonaws.services.s3control.model.*;

public void getBucketPolicy(String bucketArn) {

    GetBucketPolicyRequest reqGetBucketPolicy = new GetBucketPolicyRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);
```

```
    GetBucketPolicyResult respGetBucketPolicy =
 s3ControlClient.getBucketPolicy(reqGetBucketPolicy);
    System.out.printf("GetBucketPolicy Response: %s%n",
 respGetBucketPolicy.toString());

}
```

Deleting the bucket policy for your Amazon S3 on Outposts bucket

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size. For more information, see Bucket policy.

The following topics show you how to view your Amazon S3 on Outposts bucket policy by using the AWS Management Console or AWS Command Line Interface (AWS CLI).

Using the S3 console

To delete a bucket policy

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket whose permission you want to edit.
4. Choose the **Permissions** tab.
5. In the **Outposts bucket policy** section, choose **Delete**.
6. Confirm the deletion.

Using the AWS CLI

The following example deletes the bucket policy for an S3 on Outposts bucket (`s3-outposts:DeleteBucket`) by using the AWS CLI. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control delete-bucket-policy --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

Bucket policy examples

With S3 on Outposts bucket policies, you can secure access to objects in your S3 on Outposts buckets, so that only users with the appropriate permissions can access them. You can even prevent authenticated users without the appropriate permissions from accessing your S3 on Outposts resources.

This section presents examples of typical use cases for S3 on Outposts bucket policies. To test these policies, replace the *`user input placeholders`* with your own information (such as your bucket name).

To grant or deny permissions to a set of objects, you can use wildcard characters (`*`) in Amazon Resource Names (ARNs) and other values. For example, you can control access to groups of objects that begin with a common prefix or end with a given extension, such as `.html`.

For more information about AWS Identity and Access Management (IAM) policy language, see Setting up IAM with S3 on Outposts.

> **Note**
>
> When testing s3outposts permissions by using the Amazon S3 console, you must grant additional permissions that the console requires, such as `s3outposts:createendpoint`, `s3outposts:listendpoints`, and so on.

Additional resources for creating bucket policies

- For a list of the IAM policy actions, resources, and condition keys you can use when creating an S3 on Outposts bucket policy, see Actions, resources, and condition keys for Amazon S3 on Outposts.
- For guidance on creating your S3 on Outposts policy, see Adding or editing a bucket policy for an Amazon S3 on Outposts bucket.

Topics

- Managing access to an Amazon S3 on Outposts bucket based on specific IP addresses

Managing access to an Amazon S3 on Outposts bucket based on specific IP addresses

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size. For more information, see Bucket policy.

Restrict access to specific IP addresses

The following example denies all users from performing any S3 on Outposts operations on objects in the specified buckets unless the request originates from the specified range of IP addresses.

> **Note**
>
> When restricting access to a specific IP address, make sure that you also specify which VPC endpoints, VPC source IP addresses, or external IP addresses can access the S3 on Outposts bucket. Otherwise, you might lose access to the bucket if your policy denies all users from performing any s3outposts operations on objects in your S3 on Outposts bucket without the proper permissions already in place.

This policy's `Condition` statement identifies *`192.0.2.0/24`* as the range of allowed IP version 4 (IPv4) IP addresses.

The `Condition` block uses the `NotIpAddress` condition and the `aws:SourceIp` condition key, which is an AWS wide condition key. The `aws:SourceIp` condition key can only be used for public IP address ranges. For more information about these condition keys, see Actions, resources, and condition keys for S3 on Outposts. The `aws:SourceIp` IPv4 values use standard CIDR notation. For more information, see IAM JSON policy elements reference in the *IAM User Guide*.

> **Warning**
>
> Before using this S3 on Outposts policy, replace the *`192.0.2.0/24`* IP address range in this example with an appropriate value for your use case. Otherwise, you'll lose the ability to access your bucket.

```
{
    "Version": "2012-10-17",
    "Id": "S3OutpostsPolicyId1",
    "Statement": [
        {
            "Sid": "IPAllow",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3-outposts:*",
            "Resource": [
                "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
accesspoint/EXAMPLE-ACCESS-POINT-NAME",
                "arn:aws:aws:s3-outposts:region:111122223333:outpost/OUTPOSTS-ID/
bucket/amzn-s3-demo-bucket"
            ],
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": "192.0.2.0/24"
                }
            }
        }
    ]
}
```

Allow both IPv4 and IPv6 addresses

When you start using IPv6 addresses, we recommend that you update all of your organization's policies with your IPv6 address ranges in addition to your existing IPv4 ranges. Doing this will help to make sure that the policies continue to work as you make the transition to IPv6.

The following S3 on Outposts example bucket policy shows how to mix IPv4 and IPv6 address ranges to cover all of your organization's valid IP addresses. The example policy allows access to the example IP addresses *192.0.2.1* and *2001:DB8:1234:5678::1* and denies access to the addresses *203.0.113.1* and *2001:DB8:1234:5678:ABCD::1*.

The `aws:SourceIp` condition key can only be used for public IP address ranges. The IPv6 values for `aws:SourceIp` must be in standard CIDR format. For IPv6, we support using `::` to represent a range of 0s (for example, `2001:DB8:1234:5678::/64`). For more information, see IP address condition operators in the *IAM User Guide*.

Warning

Replace the IP address ranges in this example with appropriate values for your use case before using this S3 on Outposts policy. Otherwise, you might lose the ability to access your bucket.

JSON

```
{
    "Id": "S3OutpostsPolicyId2",
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowIPmix",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "s3-outposts:GetObject",
                "s3-outposts:PutObject",
                "s3-outposts:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/amzn-s3-demo-bucket",
                "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/amzn-s3-demo-bucket/*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
                        "2001:DB8:1234:5678::/64"
                    ]
                },
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "203.0.113.0/24",
                        "2001:DB8:1234:5678:ABCD::/80"
                    ]
```

```
                }
            }
        }
    ]
}
```

Listing Amazon S3 on Outposts buckets

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

For more information about working with buckets in S3 on Outposts, see Working with S3 on Outposts buckets.

The following examples show you how to return a list of your S3 on Outposts buckets by using the AWS Management Console, AWS CLI, and AWS SDK for Java.

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Under **Outposts buckets**, review your list of S3 on Outposts buckets.

Using the AWS CLI

The following AWS CLI example gets a list of buckets in an Outpost. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see list-regional-buckets in the *AWS CLI Reference*.

```
aws s3control list-regional-buckets --account-id 123456789012 --outpost-
id op-01ac5d28a6a232904
```

Using the AWS SDK for Java

The following SDK for Java example gets a list of buckets in an Outpost. For more information, see ListRegionalBuckets in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.services.s3control.model.*;

public void listRegionalBuckets() {

    ListRegionalBucketsRequest reqListBuckets = new ListRegionalBucketsRequest()
            .withAccountId(AccountId)
            .withOutpostId(OutpostId);

    ListRegionalBucketsResult respListBuckets =
 s3ControlClient.listRegionalBuckets(reqListBuckets);
    System.out.printf("ListRegionalBuckets Response: %s%n",
 respListBuckets.toString());

}
```

Getting an S3 on Outposts bucket by using the AWS CLI and the SDK for Java

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

The following examples show you how to get an S3 on Outposts bucket by using the AWS CLI and AWS SDK for Java.

Note

When you're working with Amazon S3 on Outposts through the AWS CLI or AWS SDKs, you provide the access point ARN for the Outpost in place of the bucket name. The access point ARN takes the following form, where *region* is the AWS Region code for the Region that the Outpost is homed to:
`arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-access-point`
For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

Using the AWS CLI

The following S3 on Outposts example gets a bucket by using the AWS CLI. To use this command, replace each *user input placeholder* with your own information. For more information about this command, see get-bucket in the *AWS CLI Reference*.

```
aws s3control get-bucket --account-id 123456789012 --bucket "arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket"
```

Using the AWS SDK for Java

The following S3 on Outposts example gets a bucket by using the SDK for Java. For more information, see GetBucket in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.services.s3control.model.*;

public void getBucket(String bucketArn) {

    GetBucketRequest reqGetBucket = new GetBucketRequest()
            .withBucket(bucketArn)
            .withAccountId(AccountId);

    GetBucketResult respGetBucket = s3ControlClient.getBucket(reqGetBucket);
    System.out.printf("GetBucket Response: %s%n", respGetBucket.toString());

}
```

Deleting your Amazon S3 on Outposts bucket

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

For more information about working with buckets in S3 on Outposts, see Working with S3 on Outposts buckets.

The AWS account that creates the bucket owns it and is the only one that can delete it.

Note

- Outposts buckets must be empty before they can be deleted.

  The Amazon S3 console doesn't support S3 on Outposts object actions. To delete objects in an S3 on Outposts bucket, you must use the REST API, AWS CLI, or AWS SDKs.
- Before you can delete an Outposts bucket, you must delete any Outposts access points for the bucket. For more information, see Deleting an access point.
- You cannot recover a bucket after it has been deleted.

The following examples show you how to delete an S3 on Outposts bucket by using the AWS Management Console and AWS Command Line Interface (AWS CLI).

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.

3. Choose the bucket that you want to delete, and choose **Delete**.
4. Confirm the deletion.

Using the AWS CLI

The following example deletes an S3 on Outposts bucket (`s3-outposts:DeleteBucket`) by using the AWS CLI. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control delete-bucket --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

Working with Amazon S3 on Outposts access points

To access your Amazon S3 on Outposts bucket, you must create and configure an access point.

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

> **Note**
>
> The AWS account that creates the Outposts bucket owns it and is the only one that can assign access points to it.

The following sections describe how to create and manage access points for S3 on Outposts buckets.

Topics

- Creating an S3 on Outposts access point
- Using a bucket-style alias for your S3 on Outposts bucket access point
- Viewing information about an access point configuration

- View a list of your Amazon S3 on Outposts access points
- Deleting an access point
- Adding or editing an access point policy
- Viewing an access point policy for an S3 on Outposts access point

Creating an S3 on Outposts access point

To access your Amazon S3 on Outposts bucket, you must create and configure an access point.

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

The following examples show you how to create an S3 on Outposts access point by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

> **Note**
>
> The AWS account that creates the Outposts bucket owns it and is the only one that can assign access points to it.

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to create an Outposts access point for.
4. Choose the **Outposts access points** tab.
5. In the **Outposts access points** section, choose **Create Outposts access point**.
6. In **Outposts access point settings**, enter a name for the access point, and then choose the virtual private cloud (VPC) for the access point.
7. If you want to add a policy for your access point, enter it in the **Outposts access point policy** section.

For more information, see Setting up IAM with S3 on Outposts.

Using the AWS CLI

Example

The following AWS CLI example creates an access point for an Outposts bucket. To run this command, replace the *user input placeholders* with your own information.

```
aws s3control create-access-point --account-id 123456789012
 --name example-outposts-access-point --bucket "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket" --vpc-configuration VpcId=example-vpc-12345
```

Using the AWS SDK for Java

Example

For examples of how to create an access point for an S3 Outposts bucket with the AWS SDK for Java, see CreateOutpostsAccessPoint.java in the *AWS SDK for Java 2.x Code Examples*.

Using a bucket-style alias for your S3 on Outposts bucket access point

With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Every time you create an access point for a bucket, S3 on Outposts automatically generates an access point alias. You can use this access point alias instead of an access point ARN for any data plane operation. For example, you can use an access point alias to perform object-level operations such as PUT, GET, LIST, and more. For a list of these operations, see Amazon S3 API operations for managing objects.

The following examples show an ARN and access point alias for an access point named *my-access-point*.

- **Access point ARN** – `arn:aws:s3-outposts:`*region*`:`*123456789012*`:outpost/`*op-01ac5d28a6a232904*`/accesspoint/`*my-access-point*
- **Access point alias** – *my-access-po-o01ac5d28a6a232904e8xz5w8ijx1qzlbp3i3kuse10*`--op-s3`

For more information about ARNs, see Amazon Resource Names (ARNs) in the *AWS General Reference*.

For more information about access point aliases, see the following topics.

Topics

- Access point aliases
- Using an access point alias in an S3 on Outposts object operation
- Limitations

Access point aliases

An access point alias is created within the same namespace as an S3 on Outposts bucket. When you create an access point, S3 on Outposts automatically generates an access point alias that cannot be changed. An access point alias meets all the requirements of a valid S3 on Outposts bucket name and consists of the following parts:

```
access point name prefix-metadata--op-s3
```

> **Note**
>
> The `--op-s3` suffix is reserved for access point aliases, we recommend that you don't use it for bucket or access point names. For more information about S3 on Outposts bucket-naming rules, see Working with S3 on Outposts buckets.

Finding the access point alias

The following examples show you how to find an access point alias by using the Amazon S3 console and the AWS CLI.

Example: Find and copy an access point alias in the Amazon S3 console

After you create an access point in the console, you can get the access point alias from the **Access Point alias** column in the **Access Points** list.

To copy an access point alias

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.

3. To copy the access point alias, do one of the following:

   - In the **Access Points** list, select the option button next to the access point name, and then choose **Copy Access Point alias**.
   - Choose the access point name. Then, under **Outposts access point overview**, copy the access point alias.

Example: Create an access point by using the AWS CLI and find the access point alias in the response

The following AWS CLI example for the `create-access-point` command creates the access point and returns the automatically generated access point alias. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-access-point --bucket example-outposts-bucket --name example-
outposts-access-point --account-id 123456789012

{
    "AccessPointArn":
    "arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/
accesspoint/example-outposts-access-point",
    "Alias": "example-outp-o01ac5d28a6a232904e8xz5w8ijx1qzlbp3i3kuse10--op-s3"
}
```

Example: Get an access point alias by using the AWS CLI

The following AWS CLI example for the `get-access-point` command returns information about the specified access point. This information includes the access point alias. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control get-access-point --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket --name example-outposts-access-point --account-id 123456789012

{
    "Name": "example-outposts-access-point",
    "Bucket": "example-outposts-bucket",
    "NetworkOrigin": "Vpc",
    "VpcConfiguration": {
        "VpcId": "vpc-01234567890abcdef"
```

```
    },
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2022-09-18T17:49:15.584000+00:00",
    "Alias": "example-outp-o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3"
}
```

Example: List access points to find an access point alias by using the AWS CLI

The following AWS CLI example for the `list-access-points` command lists information about the specified access point. This information includes the access point alias. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control list-access-points --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket

{
    "AccessPointList": [
        {
            "Name": "example-outposts-access-point",
            "NetworkOrigin": "Vpc",
            "VpcConfiguration": {
                "VpcId": "vpc-01234567890abcdef"
            },
            "Bucket": "example-outposts-bucket",
            "AccessPointArn": "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point",
            "Alias": "example-outp-o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3"
        }
    ]
}
```

Using an access point alias in an S3 on Outposts object operation

When adopting access points, you can use access point alias without requiring extensive code changes.

This AWS CLI example shows a `get-object` operation for an S3 on Outposts bucket. This example uses the access point alias as the value for `--bucket` instead of the full access point ARN.

```
aws s3api get-object --bucket my-access-po-
o0b1d075431d83bebde8xz5w8ijx1qzlbp3i3kuse10--op-s3 --key  testkey  sample-object.rtf

{
    "AcceptRanges": "bytes",
    "LastModified": "2020-01-08T22:16:28+00:00",
    "ContentLength": 910,
    "ETag": "\"00751974dc146b76404bb7290f8f51bb\"",
    "VersionId": "null",
    "ContentType": "text/rtf",
    "Metadata": {}
}
```

Limitations

- Aliases cannot be configured by customers.
- Aliases cannot be deleted or modified or disabled on an access point.
- You can't use an access point alias for S3 on Outposts control plane operations. For a list of S3 on Outposts control plane operations, see Amazon S3 Control API operations for managing buckets.
- Aliases cannot be used in AWS Identity and Access Management (IAM) policies.

Viewing information about an access point configuration

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

The following topics show you how to return configuration information for an S3 on Outposts access point by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Outposts access points**.
3. Choose the Outposts access point that you want to view configuration details for.
4. Under **Outposts access point overview**, review the access point configuration details.

Using the AWS CLI

The following AWS CLI example gets an access point for an Outposts bucket. Replace the *`user input placeholders`* with your own information.

```
aws s3control get-access-point --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

Using the AWS SDK for Java

The following SDK for Java example gets an access point for an Outposts bucket.

```
import com.amazonaws.services.s3control.model.*;

public void getAccessPoint(String accessPointArn) {

    GetAccessPointRequest reqGetAP = new GetAccessPointRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn);

    GetAccessPointResult respGetAP = s3ControlClient.getAccessPoint(reqGetAP);
    System.out.printf("GetAccessPoint Response: %s%n", respGetAP.toString());

}
```

View a list of your Amazon S3 on Outposts access points

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

The following topics show you how to return a list of your S3 on Outposts access points by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.
3. Under **Outposts access points**, review your list of S3 on Outposts access points.

Using the AWS CLI

The following AWS CLI example lists the access points for an Outposts bucket. To run this command, replace the *user input placeholders* with your own information.

```
aws s3control list-access-points --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-
bucket
```

Using the AWS SDK for Java

The following SDK for Java example lists the access points for an Outposts bucket.

```
import com.amazonaws.services.s3control.model.*;

public void listAccessPoints(String bucketArn) {

    ListAccessPointsRequest reqListAPs = new ListAccessPointsRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    ListAccessPointsResult respListAPs = s3ControlClient.listAccessPoints(reqListAPs);
    System.out.printf("ListAccessPoints Response: %s%n", respListAPs.toString());

}
```

Deleting an access point

Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 object operations, such as `GetObject` and `PutObject`. With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Access points support only virtual-host-style addressing.

The following examples show you how to delete an access point by using the AWS Management Console and the AWS Command Line Interface (AWS CLI).

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.
3. In the **Outposts access points** section, choose the Outposts access point that you want to delete.
4. Choose **Delete**.
5. Confirm the deletion.

Using the AWS CLI

The following AWS CLI example deletes an Outposts access point. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control delete-access-point --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

Adding or editing an access point policy

Access points have distinct permissions and network controls that Amazon S3 on Outposts applies for any request that is made through that access point. Each access point enforces a customized access point policy that works in conjunction with the bucket policy that is attached to the underlying bucket. For more information, see Access points.

The following topics show you how to add or edit the access point policy for your S3 on Outposts access point by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to edit the access point policy for.
4. Choose the **Outposts access points** tab.

5. In the **Outposts access points** section, choose the access point whose policy you want to edit, and choose **Edit policy**.
6. Add or edit the policy in the **Outposts access point policy** section. For more information, see Setting up IAM with S3 on Outposts.

Using the AWS CLI

The following AWS CLI example puts a policy on an Outposts access point.

1. Save the following access point policy to a JSON file. In this example, the file is named `appolicy1.json`. Replace the *`user input placeholders`* with your own information.

```
{
   "Version":"2012-10-17",
   "Id":"exampleAccessPointPolicy",
   "Statement":[
      {
         "Sid":"st1",
         "Effect":"Allow",
         "Principal":{
            "AWS":"123456789012"
         },
         "Action":"s3-outposts:*",
         "Resource":"arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point"
      }
   ]
}
```

2. Submit the JSON file as part of the `put-access-point-policy` CLI command. Replace the *`user input placeholders`* with your own information.

```
aws s3control put-access-point-policy --account-id 123456789012 --name arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --policy file://appolicy1.json
```

Using the AWS SDK for Java

The following SDK for Java example puts a policy on an Outposts access point.

```
import com.amazonaws.services.s3control.model.*;

public void putAccessPointPolicy(String accessPointArn) {

    String policy = "{\"Version\":\"2012-10-17\",\"Id\":\"testAccessPointPolicy\",
\"Statement\":[{\"Sid\":\"st1\",\"Effect\":\"Allow\",\"Principal\":{\"AWS\":\"" +
 AccountId + "\"},\"Action\":\"s3-outposts:*\",\"Resource\":\"" + accessPointArn +
 "\"}]}";

    PutAccessPointPolicyRequest reqPutAccessPointPolicy = new
 PutAccessPointPolicyRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn)
            .withPolicy(policy);

    PutAccessPointPolicyResult respPutAccessPointPolicy =
 s3ControlClient.putAccessPointPolicy(reqPutAccessPointPolicy);
    System.out.printf("PutAccessPointPolicy Response: %s%n",
 respPutAccessPointPolicy.toString());
    printWriter.printf("PutAccessPointPolicy Response: %s%n",
 respPutAccessPointPolicy.toString());

}
```

Viewing an access point policy for an S3 on Outposts access point

Access points have distinct permissions and network controls that Amazon S3 on Outposts applies for any request that is made through that access point. Each access point enforces a customized access point policy that works in conjunction with the bucket policy that is attached to the underlying bucket. For more information, see Access points.

For more information about working with access points in S3 on Outposts, see Working with S3 on Outposts buckets.

The following topics show you how to view your S3 on Outposts access point policy by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.

3. Choose the Outposts access point that you want to view the policy for.
4. On the **Permissions** tab, review the S3 on Outposts access point policy.
5. To edit the access point policy, see Adding or editing an access point policy.

Using the AWS CLI

The following AWS CLI example gets a policy for an Outposts access point. To run this command, replace the *user input placeholders* with your own information.

```
aws s3control get-access-point-policy --account-id 123456789012 --name arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-access-point
```

Using the AWS SDK for Java

The following SDK for Java example gets a policy for an Outposts access point.

```
import com.amazonaws.services.s3control.model.*;

public void getAccessPointPolicy(String accessPointArn) {

    GetAccessPointPolicyRequest reqGetAccessPointPolicy = new
 GetAccessPointPolicyRequest()
            .withAccountId(AccountId)
            .withName(accessPointArn);

    GetAccessPointPolicyResult respGetAccessPointPolicy =
 s3ControlClient.getAccessPointPolicy(reqGetAccessPointPolicy);
    System.out.printf("GetAccessPointPolicy Response: %s%n",
 respGetAccessPointPolicy.toString());
    printWriter.printf("GetAccessPointPolicy Response: %s%n",
 respGetAccessPointPolicy.toString());

}
```

Working with Amazon S3 on Outposts endpoints

To route requests to an Amazon S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. In order to create an endpoint, you will need an active connection with your service link to your Outposts home region. Each virtual private cloud (VPC) on your Outpost can

have one associated endpoint. For more information about endpoint quotas, see S3 on Outposts network requirements. You must create an endpoint to be able to access your Outposts buckets and perform object operations. For more information, see Endpoints.

After you create an endpoint, you can use the 'Status' field, to understand the state of the endpoint. If your Outposts is offline, it will return a CREATE_FAILED. You can check your service link connection, delete the endpoint, and retry the create operation after your connection has resumed. For a list of additional error codes, see below. For more information, see Endpoints.

API	Status	Failed Reason Error Code	Message - Failed Reason
`CreateEndpoint`	Create_Failed	OutpostNotReachable	Endpoint could not be created as the service link connection to your Outposts home Region is down. Check your connection, delete the endpoint, and try again.
`CreateEndpoint`	Create_Failed	InternalError	Endpoint could not be created due to Internal Error. Please delete the endpoint and create again.
`DeleteEndpoint`	Delete_Failed	OutpostNotReachable	Endpoint could not be deleted as the service link connection to your Outposts home Region is down. Check your connection and please try again.
`DeleteEndpoint`	Delete_Failed	InternalError	Endpoint could not be deleted due to Internal Error. Please try again.

For more information about working with buckets on S3 on Outposts, see Working with S3 on Outposts buckets.

The following sections describe how to create and manage endpoints for S3 on Outposts.

Topics

- Creating an endpoint on an Outpost
- Viewing a list of your Amazon S3 on Outposts endpoints

- Deleting an Amazon S3 on Outposts endpoint

Creating an endpoint on an Outpost

To route requests to an Amazon S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. In order to create an endpoint, you will need an active connection with your service link to your Outposts home region. Each virtual private cloud (VPC) on your Outpost can have one associated endpoint. For more information about endpoint quotas, see S3 on Outposts network requirements. You must create an endpoint to be able to access your Outposts buckets and perform object operations. For more information, see Endpoints.

Permissions

For more information about the permissions that are required to create an endpoint, see Permissions for S3 on Outposts endpoints.

When you create an endpoint, S3 on Outposts also creates a service-linked role in your AWS account. For more information, see Using service-linked roles for Amazon S3 on Outposts.

The following examples show you how to create an S3 on Outposts endpoint by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.
3. Choose the **Outposts endpoints** tab.
4. Choose **Create Outposts endpoint**.
5. Under **Outpost**, choose the Outpost to create this endpoint on.
6. Under **VPC**, choose a VPC that does not yet have an endpoint and that also complies with the rules for Outposts endpoints.

   A virtual private cloud (VPC) enables you to launch AWS resources into a virtual network that you define. This virtual network closely resembles a traditional network that you would operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

   If you don't have a VPC, choose **Create VPC**. For more information, see Creating access points restricted to a virtual private cloud (VPC) in the *Amazon S3 User Guide*.

7. Choose **Create Outposts endpoint**.

Using the AWS CLI

Example

The following AWS CLI example creates an endpoint for an Outpost by using the VPC resource access type. The VPC is derived from the subnet. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1
```

The following AWS CLI example creates an endpoint for an Outpost by using the customer-owned IP address pool (CoIP pool) access type. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3outposts create-endpoint --outpost-id op-01ac5d28a6a232904 --subnet-id
 subnet-8c7a57c5 --security-group-id sg-ab19e0d1 --access-type CustomerOwnedIp --
customer-owned-ipv4-pool ipv4pool-coip-12345678901234567
```

Using the AWS SDK for Java

Example

For examples of how to create an endpoint for an S3 Outpost with the AWS SDK for Java, see CreateOutpostsEndPoint.java in the *AWS SDK for Java 2.x Code Examples*.

Viewing a list of your Amazon S3 on Outposts endpoints

To route requests to an Amazon S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. In order to create an endpoint, you will need an active connection with your service link to your Outposts home region. Each virtual private cloud (VPC) on your Outpost can have one associated endpoint. For more information about endpoint quotas, see S3 on Outposts network requirements. You must create an endpoint to be able to access your Outposts buckets and perform object operations. For more information, see Endpoints.

The following examples show you how to return a list of your S3 on Outposts endpoints by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.
3. On the **Outposts access points** page, choose the **Outposts endpoints** tab.
4. Under **Outposts endpoints**, you can view a list of your S3 on Outposts endpoints.

Using the AWS CLI

The following AWS CLI example lists the endpoints for the AWS Outposts resources that are associated with your account. For more information about this command, see list-endpoints in the *AWS CLI Reference*.

```
aws s3outposts list-endpoints
```

Using the AWS SDK for Java

The following SDK for Java example lists the endpoints for an Outpost. For more information, see ListEndpoints in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.ListEndpointsRequest;
import com.amazonaws.services.s3outposts.model.ListEndpointsResult;

public void listEndpoints() {
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    ListEndpointsRequest listEndpointsRequest = new ListEndpointsRequest();
    ListEndpointsResult listEndpointsResult =
 s3OutpostsClient.listEndpoints(listEndpointsRequest);
    System.out.println("List endpoints result is " + listEndpointsResult);
}
```

Deleting an Amazon S3 on Outposts endpoint

To route requests to an Amazon S3 on Outposts access point, you must create and configure an S3 on Outposts endpoint. In order to create an endpoint, you will need an active connection with your

service link to your Outposts home region. Each virtual private cloud (VPC) on your Outpost can have one associated endpoint. For more information about endpoint quotas, see S3 on Outposts network requirements. You must create an endpoint to be able to access your Outposts buckets and perform object operations. For more information, see Endpoints.

The following examples show you how to delete your S3 on Outposts endpoints by using the AWS Management Console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts access points**.
3. On the **Outposts access points** page, choose the **Outposts endpoints** tab.
4. Under **Outposts endpoints**, choose the endpoint that you want to delete, and choose **Delete**.

Using the AWS CLI

The following AWS CLI example deletes an endpoint for an Outpost. To run this command, replace the *user input placeholders* with your own information.

```
aws s3outposts delete-endpoint --endpoint-id example-endpoint-id --outpost-
id op-01ac5d28a6a232904
```

Using the AWS SDK for Java

The following SDK for Java example deletes an endpoint for an Outpost. To use this example, replace the *user input placeholders* with your own information.

```
import com.amazonaws.arn.Arn;
import com.amazonaws.services.s3outposts.AmazonS3Outposts;
import com.amazonaws.services.s3outposts.AmazonS3OutpostsClientBuilder;
import com.amazonaws.services.s3outposts.model.DeleteEndpointRequest;

public void deleteEndpoint(String endpointArnInput) {
    String outpostId = "op-01ac5d28a6a232904";
    AmazonS3Outposts s3OutpostsClient = AmazonS3OutpostsClientBuilder
                .standard().build();

    Arn endpointArn = Arn.fromString(endpointArnInput);
    String[] resourceParts = endpointArn.getResource().getResource().split("/");
```

```
    String endpointId = resourceParts[resourceParts.length - 1];
    DeleteEndpointRequest deleteEndpointRequest = new DeleteEndpointRequest()
                .withEndpointId(endpointId)
                .withOutpostId(outpostId);
    s3OutpostsClient.deleteEndpoint(deleteEndpointRequest);
    System.out.println("Endpoint with id " + endpointId + " is deleted.");
}
```

Working with S3 on Outposts objects

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (0UTP0STS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API.

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

Object ARNs use the following format, which includes the AWS Region that the Outpost is homed to, AWS account ID, Outpost ID, bucket name, and object key:

```
arn:aws:s3-outposts:us-west-2:123456789012:outpost/ op-01ac5d28a6a232904/bucket/amzn-s3-demo-bucket1/object/myobject
```

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

Topics

- Upload an object to an S3 on Outposts bucket
- Copying an object in an Amazon S3 on Outposts bucket using the AWS SDK for Java
- Getting an object from an Amazon S3 on Outposts bucket
- Listing the objects in an Amazon S3 on Outposts bucket
- Deleting objects in Amazon S3 on Outposts buckets
- Using HeadBucket to determine if an S3 on Outposts bucket exists and you have access permissions
- Performing and managing a multipart upload with the SDK for Java
- Using presigned URLs for S3 on Outposts
- Amazon S3 on Outposts with local Amazon EMR on Outposts
- Authorization and authentication caching

Upload an object to an S3 on Outposts bucket

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

The following AWS CLI and AWS SDK for Java examples show you how to upload an object to an S3 on Outposts bucket by using an access point.

AWS CLI

Example

The following example puts an object named `sample-object.xml` into an S3 on Outposts bucket (`s3-outposts:PutObject`) by using the AWS CLI. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see put-object in the *AWS CLI Reference*.

```
aws s3api put-object --bucket arn:aws:s3-
outposts:Region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --key sample-object.xml --body sample-object.xml
```

SDK for Java

Example

For examples of how to upload an object to an S3 Outposts bucket with the AWS SDK for Java, see PutObjectOnOutpost.java in the *AWS SDK for Java 2.x Code Examples*.

Copying an object in an Amazon S3 on Outposts bucket using the AWS SDK for Java

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

The following example shows you how to copy an object in an S3 on Outposts bucket by using the AWS SDK for Java.

Using the AWS SDK for Java

The following S3 on Outposts example copies an object into a new object in the same bucket by using the SDK for Java. To use this example, replace the *`user input placeholders`* with your own information.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;

public class CopyObject {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String sourceKey = "*** Source object key ***";
        String destinationKey = "*** Destination object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Copy the object into a new object in the same bucket.
            CopyObjectRequest copyObjectRequest = new CopyObjectRequest(accessPointArn,
 sourceKey, accessPointArn, destinationKey);
            s3Client.copyObject(copyObjectRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
```

```
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Getting an object from an Amazon S3 on Outposts bucket

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

The following examples show you how to download (get) an object by using the AWS Command Line Interface (AWS CLI) and AWS SDK for Java.

Using the AWS CLI

The following example gets an object named `sample-object.xml` from an S3 on Outposts bucket (`s3-outposts:GetObject`) by using the AWS CLI. To use this command, replace each

user input placeholder with your own information. For more information about this command, see get-object in the *AWS CLI Reference*.

```
aws s3api get-object --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point --key testkey sample-object.xml
```

Using the AWS SDK for Java

The following S3 on Outposts example gets an object by using the SDK for Java. To use this example, replace each *user input placeholder* with your own information. For more information, see GetObject in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.ResponseHeaderOverrides;
import com.amazonaws.services.s3.model.S3Object;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class GetObject {
    public static void main(String[] args) throws IOException {
        String accessPointArn = "*** access point ARN ***";
        String key = "*** Object key ***";

        S3Object fullObject = null, objectPortion = null, headerOverrideObject = null;
        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Get an object and print its contents.
            System.out.println("Downloading an object");
```

```
            fullObject = s3Client.getObject(new GetObjectRequest(accessPointArn, key));
            System.out.println("Content-Type: " +
fullObject.getObjectMetadata().getContentType());
            System.out.println("Content: ");
            displayTextInputStream(fullObject.getObjectContent());

            // Get a range of bytes from an object and print the bytes.
            GetObjectRequest rangeObjectRequest = new GetObjectRequest(accessPointArn,
key)
                    .withRange(0, 9);
            objectPortion = s3Client.getObject(rangeObjectRequest);
            System.out.println("Printing bytes retrieved.");
            displayTextInputStream(objectPortion.getObjectContent());

            // Get an entire object, overriding the specified response headers, and
print the object's content.
            ResponseHeaderOverrides headerOverrides = new ResponseHeaderOverrides()
                    .withCacheControl("No-cache")
                    .withContentDisposition("attachment; filename=example.txt");
            GetObjectRequest getObjectRequestHeaderOverride = new
GetObjectRequest(accessPointArn, key)
                    .withResponseHeaders(headerOverrides);
            headerOverrideObject = s3Client.getObject(getObjectRequestHeaderOverride);
            displayTextInputStream(headerOverrideObject.getObjectContent());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        } finally {
            // To ensure that the network connection doesn't remain open, close any
open input streams.
            if (fullObject != null) {
                fullObject.close();
            }
            if (objectPortion != null) {
                objectPortion.close();
            }
            if (headerOverrideObject != null) {
                headerOverrideObject.close();
            }
```

```
        }
    }

    private static void displayTextInputStream(InputStream input) throws IOException {
        // Read the text input stream one line at a time and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        String line = null;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
}
```

Listing the objects in an Amazon S3 on Outposts bucket

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

> **Note**
>
> With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

The following examples show you how to list the objects in an S3 on Outposts bucket using the AWS CLI and AWS SDK for Java.

Using the AWS CLI

The following example lists the objects in an S3 on Outposts bucket (`s3-outposts:ListObjectsV2`) by using the AWS CLI. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see list-objects-v2 in the *AWS CLI Reference*.

```
aws s3api list-objects-v2 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-access-point
```

Note

When using this action with Amazon S3 on Outposts through the AWS SDKs, you provide the Outposts access point ARN in place of the bucket name, in the following form: `arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-Outposts-Access-Point`. For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

Using the AWS SDK for Java

The following S3 on Outposts example lists objects in a bucket by using the SDK for Java. To use this example, replace each *`user input placeholder`* with your own information.

Important

This example uses ListObjectsV2, which is the latest revision of the `ListObjects` API operation. We recommend that you use this revised API operation for application development. For backward compatibility, Amazon S3 continues to support the prior version of this API operation.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
```

```
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListObjectsV2Request;
import com.amazonaws.services.s3.model.ListObjectsV2Result;
import com.amazonaws.services.s3.model.S3ObjectSummary;

public class ListObjectsV2 {

    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            System.out.println("Listing objects");

            // maxKeys is set to 2 to demonstrate the use of
            // ListObjectsV2Result.getNextContinuationToken()
            ListObjectsV2Request req = new
 ListObjectsV2Request().withBucketName(accessPointArn).withMaxKeys(2);
            ListObjectsV2Result result;

            do {
                result = s3Client.listObjectsV2(req);

                for (S3ObjectSummary objectSummary : result.getObjectSummaries()) {
                    System.out.printf(" - %s (size: %d)\n", objectSummary.getKey(),
 objectSummary.getSize());
                }
                // If there are more than maxKeys keys in the bucket, get a
 continuation token
                // and list the next objects.
                String token = result.getNextContinuationToken();
                System.out.println("Next Continuation Token: " + token);
                req.setContinuationToken(token);
            } while (result.isTruncated());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
```

```
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Deleting objects in Amazon S3 on Outposts buckets

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

The following examples show you how to delete a single object or multiple objects in an S3 on Outposts bucket by using the AWS Command Line Interface (AWS CLI) and AWS SDK for Java.

Using the AWS CLI

The following examples show you how to delete a single object or multiple objects from an S3 on Outposts bucket.

delete-object

The following example deletes an object named `sample-object.xml` from an S3 on Outposts bucket (`s3-outposts:DeleteObject`) by using the AWS CLI. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see delete-object in the *AWS CLI Reference*.

```
aws s3api delete-object --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --key sample-object.xml
```

delete-objects

The following example deletes two objects named `sample-object.xml` and `test1.text` from an S3 on Outposts bucket (`s3-outposts:DeleteObject`) by using the AWS CLI. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see delete-objects in the *AWS CLI Reference*.

```
aws s3api delete-objects --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point --delete file://delete.json

delete.json
{
  "Objects": [
    {
      "Key": "test1.txt"
    },
    {
      "Key": "sample-object.xml"
    }
  ],
  "Quiet": false
}
```

Using the AWS SDK for Java

The following examples show you how to delete a single object or multiple objects from an S3 on Outposts bucket.

DeleteObject

The following S3 on Outposts example deletes an object in a bucket by using the SDK for Java. To use this example, specify the access point ARN for the Outpost and the key name for the object that you want to delete. For more information, see DeleteObject in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectRequest;

public class DeleteObject {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
        String keyName = "*** key name ****";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            s3Client.deleteObject(new DeleteObjectRequest(accessPointArn, keyName));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

DeleteObjects

The following S3 on Outposts example uploads and then deletes objects in a bucket by using the SDK for Java. To use this example, specify the access point ARN for the Outpost. For more information, see DeleteObjects in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.DeleteObjectsRequest;
import com.amazonaws.services.s3.model.DeleteObjectsRequest.KeyVersion;
import com.amazonaws.services.s3.model.DeleteObjectsResult;

import java.util.ArrayList;

public class DeleteObjects {

    public static void main(String[] args) {
       String accessPointArn = "arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-
outposts-access-point";


        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Upload three sample objects.
            ArrayList<KeyVersion> keys = new ArrayList<KeyVersion>();
            for (int i = 0; i < 3; i++) {
                String keyName = "delete object example " + i;
                s3Client.putObject(accessPointArn, keyName, "Object number " + i + "
 to be deleted.");
                keys.add(new KeyVersion(keyName));
            }
            System.out.println(keys.size() + " objects successfully created.");

            // Delete the sample objects.
```

```
            DeleteObjectsRequest multiObjectDeleteRequest = new
 DeleteObjectsRequest(accessPointArn)
                    .withKeys(keys)
                    .withQuiet(false);

            // Verify that the objects were deleted successfully.
            DeleteObjectsResult delObjRes =
 s3Client.deleteObjects(multiObjectDeleteRequest);
            int successfulDeletes = delObjRes.getDeletedObjects().size();
            System.out.println(successfulDeletes + " objects successfully
 deleted.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Using HeadBucket to determine if an S3 on Outposts bucket exists and you have access permissions

Objects are the fundamental entities stored in Amazon S3 on Outposts. Every object is contained in a bucket. You must use access points to access any object in an Outpost bucket. When you specify the bucket for object operations, you use the access point Amazon Resource Name (ARN) or the access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

The following example shows the ARN format for S3 on Outposts access points, which includes the AWS Region code for the Region that the Outpost is homed to, the AWS account ID, the Outpost ID, and the access point name:

```
arn:aws:s3-outposts:region:account-id:outpost/outpost-id/accesspoint/accesspoint-name
```

For more information about S3 on Outposts ARNs, see Resource ARNs for S3 on Outposts.

> **Note**
>
> With Amazon S3 on Outposts, object data is always stored on the Outpost. When AWS installs an Outpost rack, your data stays local to your Outpost to meet data-residency requirements. Your objects never leave your Outpost and are not in an AWS Region. Because the AWS Management Console is hosted in-Region, you can't use the console to upload or manage objects in your Outpost. However, you can use the REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs to upload and manage your objects through your access points.

The following AWS Command Line Interface (AWS CLI) and AWS SDK for Java examples show you how to use the HeadBucket API operation to determine if an Amazon S3 on Outposts bucket exists and whether you have permission to access it. For more information, see HeadBucket in the *Amazon Simple Storage Service API Reference*.

Using the AWS CLI

The following S3 on Outposts AWS CLI example uses the `head-bucket` command to determine if a bucket exists and you have permissions to access it. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see head-bucket in the *AWS CLI Reference*.

```
aws s3api head-bucket --bucket arn:aws:s3-
outposts:region:123456789012:outpost/op-01ac5d28a6a232904/accesspoint/example-outposts-
access-point
```

Using the AWS SDK for Java

The following S3 on Outposts example shows how to determine if a bucket exists and if you have permission to access it. To use this example, specify the access point ARN for the Outpost. For more information, see HeadBucket in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.HeadBucketRequest;
```

```
public class HeadBucket {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            s3Client.headBucket(new HeadBucketRequest(accessPointArn));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Performing and managing a multipart upload with the SDK for Java

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts resources and store and retrieve objects on-premises for applications that require local data access, local data processing, and data residency. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

The following examples show how you can use S3 on Outposts with the AWS SDK for Java to perform and manage a multipart upload.

Topics

- Perform a multipart upload of an object in an S3 on Outposts bucket

- Copy a large object in an S3 on Outposts bucket by using multipart upload
- List parts of an object in an S3 on Outposts bucket
- Retrieve a list of in-progress multipart uploads in an S3 on Outposts bucket

Perform a multipart upload of an object in an S3 on Outposts bucket

The following S3 on Outposts example initiates, uploads, and finishes a multipart upload of an object to a bucket by using the SDK for Java. To use this example, replace each *`user input placeholder`* with your own information. For more information, see Uploading an object using multipart upload in the *Amazon Simple Storage Service User Guide*.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.ArrayList;
import java.util.List;

public class MultipartUploadCopy {
    public static void main(String[] args) {
        String accessPointArn = "*** Source access point ARN ***";
        String sourceObjectKey = "*** Source object key ***";
        String destObjectKey = "*** Target object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
 InitiateMultipartUploadRequest(accessPointArn, destObjectKey);
            InitiateMultipartUploadResult initResult =
 s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
```

```
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest(accessPointArn, sourceObjectKey);
            ObjectMetadata metadataResult =
s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();

            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make sure
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(accessPointArn)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(accessPointArn)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and make
the copied object available.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
                    accessPointArn,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
```

```
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
        return etags;
    }
```

Copy a large object in an S3 on Outposts bucket by using multipart upload

The following S3 on Outposts example uses the SDK for Java to copy an object in a bucket. To use this example, replace each *user input placeholder* with your own information.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.ArrayList;
import java.util.List;

public class MultipartUploadCopy {
    public static void main(String[] args) {
        String accessPointArn = "*** Source access point ARN ***";
        String sourceObjectKey = "*** Source object key ***";
        String destObjectKey = "*** Target object key ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
```

```
                    .enableUseArnRegion()
                    .build();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
InitiateMultipartUploadRequest(accessPointArn, destObjectKey);
            InitiateMultipartUploadResult initResult =
s3Client.initiateMultipartUpload(initRequest);

            // Get the object size to track the end of the copy operation.
            GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest(accessPointArn, sourceObjectKey);
            ObjectMetadata metadataResult =
s3Client.getObjectMetadata(metadataRequest);
            long objectSize = metadataResult.getContentLength();

            // Copy the object using 5 MB parts.
            long partSize = 5 * 1024 * 1024;
            long bytePosition = 0;
            int partNum = 1;
            List<CopyPartResult> copyResponses = new ArrayList<CopyPartResult>();
            while (bytePosition < objectSize) {
                // The last part might be smaller than partSize, so check to make sure
                // that lastByte isn't beyond the end of the object.
                long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

                // Copy this part.
                CopyPartRequest copyRequest = new CopyPartRequest()
                        .withSourceBucketName(accessPointArn)
                        .withSourceKey(sourceObjectKey)
                        .withDestinationBucketName(accessPointArn)
                        .withDestinationKey(destObjectKey)
                        .withUploadId(initResult.getUploadId())
                        .withFirstByte(bytePosition)
                        .withLastByte(lastByte)
                        .withPartNumber(partNum++);
                copyResponses.add(s3Client.copyPart(copyRequest));
                bytePosition += partSize;
            }

            // Complete the upload request to concatenate all uploaded parts and make
the copied object available.
            CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest(
```

```
                    accessPointArn,
                    destObjectKey,
                    initResult.getUploadId(),
                    getETags(copyResponses));
            s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    // This is a helper function to construct a list of ETags.
    private static List<PartETag> getETags(List<CopyPartResult> responses) {
        List<PartETag> etags = new ArrayList<PartETag>();
        for (CopyPartResult response : responses) {
            etags.add(new PartETag(response.getPartNumber(), response.getETag()));
        }
        return etags;
    }
}
```

List parts of an object in an S3 on Outposts bucket

The following S3 on Outposts example lists the parts of an object in a bucket by using the SDK for Java. To use this example, replace each *`user input placeholder`* with your own information.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.util.List;

public class ListParts {
    public static void main(String[] args) {
        String accessPointArn = "*** access point ARN ***";
```

```
        String keyName = "*** Key name ***";
        String uploadId = "*** Upload ID ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            ListPartsRequest listPartsRequest = new ListPartsRequest(accessPointArn,
 keyName, uploadId);
            PartListing partListing = s3Client.listParts(listPartsRequest);
            List<PartSummary> partSummaries = partListing.getParts();

            System.out.println(partSummaries.size() + " multipart upload parts");
            for (PartSummary p : partSummaries) {
                System.out.println("Upload part: Part number = \"" + p.getPartNumber()
 + "\", ETag = " + p.getETag());
            }

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Retrieve a list of in-progress multipart uploads in an S3 on Outposts bucket

The following S3 on Outposts example shows how to retrieve a list of the in-progress multipart uploads from an Outposts bucket by using the SDK for Java. To use this example, replace each *`user input placeholder`* with your own information.

```
import com.amazonaws.AmazonServiceException;
```

```
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListMultipartUploadsRequest;
import com.amazonaws.services.s3.model.MultipartUpload;
import com.amazonaws.services.s3.model.MultipartUploadListing;

import java.util.List;

public class ListMultipartUploads {
    public static void main(String[] args) {
                String accessPointArn = "*** access point ARN ***";

        try {
            // This code expects that you have AWS credentials set up per:
            // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-
credentials.html
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .enableUseArnRegion()
                    .build();

            // Retrieve a list of all in-progress multipart uploads.
            ListMultipartUploadsRequest allMultipartUploadsRequest = new
 ListMultipartUploadsRequest(accessPointArn);
            MultipartUploadListing multipartUploadListing =
 s3Client.listMultipartUploads(allMultipartUploadsRequest);
            List<MultipartUpload> uploads =
 multipartUploadListing.getMultipartUploads();

            // Display information about all in-progress multipart uploads.
            System.out.println(uploads.size() + " multipart upload(s) in progress.");
            for (MultipartUpload u : uploads) {
                System.out.println("Upload in progress: Key = \"" + u.getKey() + "\",
 id = " + u.getUploadId());
            }
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
```

```
    }
}
```

Using presigned URLs for S3 on Outposts

To grant time-limited access to objects that are stored locally on an Outpost without updating your bucket policy, you can use a presigned URL. With presigned URLs, you as the bucket owner can share objects with individuals in your virtual private cloud (VPC) or grant them the ability to upload or delete objects.

When you create a presigned URL by using the AWS SDKs or the AWS Command Line Interface (AWS CLI), you associate the URL with a specific action. You also grant time-limited access to the presigned URL by choosing a custom expiration time that can be as low as 1 second and as high as 7 days. When you share the presigned URL, the individual in the VPC can perform the action embedded in the URL as if they were the original signing user. When the URL reaches its expiration time, the URL expires and no longer works.

Limiting presigned URL capabilities

The capabilities of a presigned URL are limited by the permissions of the user who created it. In essence, presigned URLs are bearer tokens that grant access to those who possess them. As such, we recommend that you protect them appropriately.

AWS Signature Version 4 (SigV4)

To enforce specific behavior when presigned URL requests are authenticated by using AWS Signature Version 4 (SigV4), you can use condition keys in bucket policies and access point policies. For example, you can create a bucket policy that uses the `s3-outposts:signatureAge` condition to deny any Amazon S3 on Outposts presigned URL request on objects in the `example-outpost-bucket` bucket if the signature is more than 10 minutes old. To use this example, replace the *`user input placeholders`* with your own information.

JSON

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

```
            "Sid": "Deny a presigned URL request if the signature is more than 10
 minutes old",
            "Effect": "Deny",
            "Principal": {"AWS":"444455556666"},
            "Action": "s3-outposts:*",
            "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/
object/*",
            "Condition": {
                "NumericGreaterThan": {"s3-outposts:signatureAge": 600000},
                "StringEquals": {"s3-outposts:authType": "REST-QUERY-STRING"}
            }
        }
    ]
}
```

For a list of condition keys and additional example policies that you can use to enforce specific behavior when presigned URL requests are authenticated by using Signature Version 4, see AWS Signature Version 4 (SigV4) authentication-specific policy keys.

Network path restriction

If you want to restrict the use of presigned URLs and all S3 on Outposts access to particular network paths, you can write policies that require a particular network path. To set the restriction on the IAM principal that makes the call, you can use identity-based AWS Identity and Access Management (IAM) policies (for example, user, group, or role policies). To set the restriction on the S3 on Outposts resource, you can use resource-based policies (for example, bucket and access point policies).

A network-path restriction on the IAM principal requires the user of those credentials to make requests from the specified network. A restriction on the bucket or access point requires that all requests to that resource originate from the specified network. These restrictions also apply outside of the presigned URL scenario.

The IAM global condition that you use depends on the type of endpoint. If you are using the public endpoint for S3 on Outposts, use `aws:SourceIp`. If you are using a VPC endpoint for S3 on Outposts, use `aws:SourceVpc` or `aws:SourceVpce`.

The following IAM policy statement requires the principal to access AWS only from the specified network range. With this policy statement, all access must originate from that range. This includes

the case of someone who's using a presigned URL for S3 on Outposts. To use this example, replace the *user input placeholders* with your own information.

```
{
    "Sid": "NetworkRestrictionForIAMPrincipal",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "NotIpAddressIfExists": {"aws:SourceIp": "IP-address-range"},
        "BoolIfExists": {"aws:ViaAWSService": "false"}
    }
}
```

For an example bucket policy that uses the `aws:SourceIP` AWS global condition key to restrict access to an S3 on Outposts bucket to a specific network range, see Setting up IAM with S3 on Outposts.

Who can create a presigned URL

Anyone with valid security credentials can create a presigned URL. But for a user in the VPC to successfully access an object, the presigned URL must be created by someone who has permission to perform the operation that the presigned URL is based upon.

You can use the following credentials to create a presigned URL:

- **IAM instance profile** – Valid up to 6 hours.
- **AWS Security Token Service** – Valid up to 36 hours when signed with permanent credentials, such as the credentials of the AWS account root user or an IAM user.
- **IAM user** – Valid up to 7 days when you're using AWS Signature Version 4.

  To create a presigned URL that's valid for up to 7 days, first delegate IAM user credentials (the access key and secret key) to the SDK that you're using. Then, generate a presigned URL by using AWS Signature Version 4.

Note

- If you created a presigned URL by using a temporary token, the URL expires when the token expires, even if you created the URL with a later expiration time.

- Because presigned URLs grant access to your S3 on Outposts buckets to whoever has the URL, we recommend that you protect them appropriately. For more information about protecting presigned URLs, see Limiting presigned URL capabilities.

When does S3 on Outposts check the expiration date and time of a presigned URL?

At the time of the HTTP request, S3 on Outposts checks the expiration date and time of a signed URL. For example, if a client begins to download a large file immediately before the expiration time, the download continues even if the expiration time passes during the download. However, if the connection drops and the client tries to restart the download after the expiration time passes, the download fails.

For more information about using a presigned URL to share or upload objects, see the following topics.

Topics

- Sharing objects by using presigned URLs
- Generating a presigned URL to upload an object to an S3 on Outposts bucket

Sharing objects by using presigned URLs

To grant time-limited access to objects that are stored locally on an Outpost without updating your bucket policy, you can use a presigned URL. With presigned URLs, you as the bucket owner can share objects with individuals in your virtual private cloud (VPC) or grant them the ability to upload or delete objects.

When you create a presigned URL by using the AWS SDKs or the AWS Command Line Interface (AWS CLI), you associate the URL with a specific action. You also grant time-limited access to the presigned URL by choosing a custom expiration time that can be as low as 1 second and as high as 7 days. When you share the presigned URL, the individual in the VPC can perform the action embedded in the URL as if they were the original signing user. When the URL reaches its expiration time, the URL expires and no longer works.

When you create a presigned URL, you must provide your security credentials, and then specify the following:

- An access point Amazon Resource Name (ARN) for the Amazon S3 on Outposts bucket
- An object key
- An HTTP method (GET for downloading objects)
- An expiration date and time

A presigned URL is valid only for the specified duration. That is, you must start the action that's allowed by the URL before the expiration date and time. You can use a presigned URL multiple times, up to the expiration date and time. If you created a presigned URL by using a temporary token, then the URL expires when the token expires, even if you created the URL with a later expiration time.

Users in the virtual private cloud (VPC) who have access to the presigned URL can access the object. For example, if you have a video in your bucket and both the bucket and the object are private, you can share the video with others by generating a presigned URL. Because presigned URLs grant access to your S3 on Outposts buckets to whoever has the URL, we recommend that you protect these URLs appropriately. For more details about protecting presigned URLs, see Limiting presigned URL capabilities.

Anyone with valid security credentials can create a presigned URL. However, the presigned URL must be created by someone who has permission to perform the operation that the presigned URL is based upon. For more information, see Who can create a presigned URL.

You can generate a presigned URL to share an object in an S3 on Outposts bucket by using the AWS SDKs and the AWS CLI. For more information, see the following examples.

Using the AWS SDKs

You can use the AWS SDKs to generate a presigned URL that you can give to others so that they can retrieve an object.

> **Note**
>
> When you use the AWS SDKs to generate a presigned URL, the maximum expiration time for a presigned URL is 7 days from the time of creation.

Java

Example

The following example generates a presigned URL that you can give to others so that they can retrieve an object from an S3 on Outposts bucket. For more information, see Using presigned URLs for S3 on Outposts. To use this example, replace the *user input placeholders* with your own information.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.HttpMethod;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GeneratePresignedUrlRequest;

import java.io.IOException;
import java.net.URL;
import java.time.Instant;

public class GeneratePresignedURL {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String accessPointArn = "*** access point ARN ***";
        String objectKey = "*** object key ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Set the presigned URL to expire after one hour.
            java.util.Date expiration = new java.util.Date();
            long expTimeMillis = Instant.now().toEpochMilli();
            expTimeMillis += 1000 * 60 * 60;
            expiration.setTime(expTimeMillis);

            // Generate the presigned URL.
            System.out.println("Generating pre-signed URL.");
```

```
            GeneratePresignedUrlRequest generatePresignedUrlRequest =
                    new GeneratePresignedUrlRequest(accessPointArn, objectKey)
                            .withMethod(HttpMethod.GET)
                            .withExpiration(expiration);
            URL url = s3Client.generatePresignedUrl(generatePresignedUrlRequest);

            System.out.println("Pre-Signed URL: " + url.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't
 process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

Example

The following example generates a presigned URL that you can give to others so that they can retrieve an object from an S3 on Outposts bucket. For more information, see Using presigned URLs for S3 on Outposts. To use this example, replace the *`user input placeholders`* with your own information.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;

namespace Amazon.DocSamples.S3
{
    class GenPresignedURLTest
    {
        private const string accessPointArn = "*** access point ARN ***";
        private const string objectKey = "*** object key ***";
        // Specify how long the presigned URL lasts, in hours.
        private const double timeoutDuration = 12;
```

```
        // Specify your bucket Region (an example Region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            string urlString = GeneratePreSignedURL(timeoutDuration);
        }
        static string GeneratePreSignedURL(double duration)
        {
            string urlString = "";
            try
            {
                GetPreSignedUrlRequest request1 = new GetPreSignedUrlRequest
                {
                    BucketName = accessPointArn,
                    Key = objectKey,
                    Expires = DateTime.UtcNow.AddHours(duration)
                };
                urlString = s3Client.GetPreSignedURL(request1);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            return urlString;
        }
    }
}
```

Python

The following example generates a presigned URL to share an object by using the SDK for Python (Boto3). For example, use a Boto3 client and the `generate_presigned_url` function to generate a presigned URL that allows you to `GET` an object.

```
import boto3
    url = boto3.client('s3').generate_presigned_url(
    ClientMethod='get_object',
    Params={'Bucket': 'ACCESS_POINT_ARN', 'Key': 'OBJECT_KEY'},
    ExpiresIn=3600)
```

For more information about using the SDK for Python (Boto3) to generate a presigned URL, see Python in the *AWS SDK for Python (Boto) API Reference*.

Using the AWS CLI

The following example AWS CLI command generates a presigned URL for an S3 on Outposts bucket. To use this example, replace the *`user input placeholders`* with your own information.

> **Note**
>
> When you use the AWS CLI to generate a presigned URL, the maximum expiration time for a presigned URL is 7 days from the time of creation.

```
aws s3 presign s3://arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/example-outpost-access-
point/mydoc.txt --expires-in 604800
```

For more information, see presign in the *AWS CLI Command Reference*.

Generating a presigned URL to upload an object to an S3 on Outposts bucket

To grant time-limited access to objects that are stored locally on an Outpost without updating your bucket policy, you can use a presigned URL. With presigned URLs, you as the bucket owner can share objects with individuals in your virtual private cloud (VPC) or grant them the ability to upload or delete objects.

When you create a presigned URL by using the AWS SDKs or the AWS Command Line Interface (AWS CLI), you associate the URL with a specific action. You also grant time-limited access to the presigned URL by choosing a custom expiration time that can be as low as 1 second and as high

as 7 days. When you share the presigned URL, the individual in the VPC can perform the action embedded in the URL as if they were the original signing user. When the URL reaches its expiration time, the URL expires and no longer works.

When you create a presigned URL, you must provide your security credentials, and then specify the following:

- An access point Amazon Resource Name (ARN) for the Amazon S3 on Outposts bucket
- An object key
- An HTTP method (PUT for uploading objects)
- An expiration date and time

A presigned URL is valid only for the specified duration. That is, you must start the action that's allowed by the URL before the expiration date and time. You can use a presigned URL multiple times, up to the expiration date and time. If you created a presigned URL by using a temporary token, then the URL expires when the token expires, even if you created the URL with a later expiration time.

If the action allowed by a presigned URL consists of multiple steps, such as a multipart upload, you must start all steps before the expiration time. If S3 on Outposts tries to start a step with an expired URL, you receive an error.

Users in the virtual private cloud (VPC) who have access to the presigned URL can upload objects. For example, a user in the VPC who has access to the presigned URL can upload an object to your bucket. Because presigned URLs grant access to your S3 on Outposts bucket to any user in the VPC who has access to the presigned URL, we recommend that you protect these URLs appropriately. For more details about protecting presigned URLs, see Limiting presigned URL capabilities.

Anyone with valid security credentials can create a presigned URL. However, the presigned URL must be created by someone who has permission to perform the operation that the presigned URL is based upon. For more information, see Who can create a presigned URL.

Using the AWS SDKs to generate a presigned URL for an S3 on Outposts object operation

Java

SDK for Java 2.x

This example shows how to generate a presigned URL that you can use to upload an object to an S3 on Outposts bucket for a limited time. For more information, see Using presigned URLs for S3 on Outposts.

```
   public static void signBucket(S3Presigner presigner, String
outpostAccessPointArn, String keyName) {

       try {
           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(accessPointArn)
                   .key(keyName)
                   .contentType("text/plain")
                   .build();

           PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))
                   .putObjectRequest(objectRequest)
                   .build();

           PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);


           String myURL = presignedRequest.url().toString();
           System.out.println("Presigned URL to upload a file to: " +myURL);
           System.out.println("Which HTTP method must be used when uploading a
file: " +
                   presignedRequest.httpRequest().method());

           // Upload content to the S3 on Outposts bucket by using this URL.
           URL url = presignedRequest.url();

           // Create the connection and use it to upload the new object by using
the presigned URL.
```

```
            HttpURLConnection connection = (HttpURLConnection)
 url.openConnection();
            connection.setDoOutput(true);
            connection.setRequestProperty("Content-Type","text/plain");
            connection.setRequestMethod("PUT");
            OutputStreamWriter out = new
 OutputStreamWriter(connection.getOutputStream());
            out.write("This text was uploaded as an object by using a presigned
 URL.");
            out.close();

            connection.getResponseCode();
            System.out.println("HTTP response code is " +
 connection.getResponseCode());

        } catch (S3Exception e) {
            e.getStackTrace();
        } catch (IOException e) {
            e.getStackTrace();
        }
    }
```

Python

SDK for Python (Boto3)

This example shows how to generate a presigned URL that can perform an S3 on Outposts action for a limited time. For more information, see Using presigned URLs for S3 on Outposts. To make a request with the URL, use the `Requests` package.

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)


def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
```

```
    Generate a presigned S3 on Outposts URL that can be used to perform an
 action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds that the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method,
            Params=method_parameters,
            ExpiresIn=expires_in
        )
        logger.info("Got presigned URL: %s", url)
    except ClientError:
        logger.exception(
            "Couldn't get a presigned URL for client method '%s'.",
 client_method)
        raise
    return url


def usage_demo():
    logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')

    print('-'*88)
    print("Welcome to the Amazon S3 on Outposts presigned URL demo.")
    print('-'*88)

    parser = argparse.ArgumentParser()
    parser.add_argument('accessPointArn', help="The name of the S3 on Outposts
 access point ARN.")
    parser.add_argument(
        'key', help="For a GET operation, the key of the object in S3 on
 Outposts. For a "
                    "PUT operation, the name of a file to upload.")
    parser.add_argument(
        'action', choices=('get', 'put'), help="The action to perform.")
    args = parser.parse_args()

    s3_client = boto3.client('s3')
    client_action = 'get_object' if args.action == 'get' else 'put_object'
```

```
    url = generate_presigned_url(
        s3_client, client_action, {'Bucket': args.accessPointArn, 'Key':
 args.key}, 1000)

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == 'get':
        response = requests.get(url)
    elif args.action == 'put':
        print("Putting data to the URL.")
        try:
            with open(args.key, 'r') as object_file:
                object_text = object_file.read()
            response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(f"Couldn't find {args.key}. For a PUT operation, the key must
 be the "
                  f"name of a file that exists on your computer.")

    if response is not None:
        print("Got response:")
        print(f"Status: {response.status_code}")
        print(response.text)

    print('-'*88)


if __name__ == '__main__':
    usage_demo()
```

Amazon S3 on Outposts with local Amazon EMR on Outposts

Amazon EMR is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark, on AWS to process and analyze vast amounts of data. By using these frameworks and related open-source projects, you can process data for analytics purposes and business intelligence workloads. Amazon EMR also helps you transform and move large amounts of data into and out of other AWS data stores and databases, and supports Amazon S3 on Outposts. For more information about Amazon EMR, see Amazon EMR on Outposts in the *Amazon EMR Management Guide*.

For Amazon S3 on Outposts, Amazon EMR started to support the Apache Hadoop S3A connector in version 7.0.0. Earlier versions of Amazon EMR don't support local S3 on Outposts, and the EMR File System (EMRFS) is not supported.

Supported applications

Amazon EMR with Amazon S3 on Outposts supports the following applications:

- Hadoop
- Spark
- Hue
- Hive
- Sqoop
- Pig
- Hudi
- Flink

For more information, see the Amazon EMR Release Guide.

Create and configure an Amazon S3 on Outposts bucket

Amazon EMR uses the AWS SDK for Java with Amazon S3 on Outposts to store input data and output data. Your Amazon EMR log files are stored in a Regional Amazon S3 location that you select and aren't stored locally on the Outpost. For more information, see Amazon EMR logs in the *Amazon EMR Management Guide*.

To conform with Amazon S3 and DNS requirements, S3 on Outposts buckets have naming restrictions and limitations. For more information, see Creating an S3 on Outposts bucket.

With Amazon EMR version 7.0.0 and later, you can use Amazon EMR with S3 on Outposts and the S3A file system.

Prerequisites

S3 on Outposts permissions – When you create your Amazon EMR instance profile, your role must contain the AWS Identity and Access Management (IAM) namespace for S3 on Outposts. S3 on Outposts has its own namespace, `s3-outposts*`. For an example policy that uses this namespace, see Setting up IAM with S3 on Outposts.

S3A connector – To configure your EMR cluster to access data from an Amazon S3 on Outposts bucket, you must use the Apache Hadoop S3A connector. To use the connector, ensure that all of your S3 URIs use the `s3a` scheme. If they don't, you can configure the file system implementation that you use for your EMR cluster so that your S3 URIs work with the S3A connector.

To configure the file system implementation to work with the S3A connector, you use the `fs.`*`file_scheme`*`.impl` and `fs.AbstractFileSystem.`*`file_scheme`*`.impl` configuration properties for your EMR cluster, where *`file_scheme`* corresponds to the type of S3 URIs that you have. To use the following example, replace the *`user input placeholders`* with your own information. For example, to change the file system implementation for S3 URIs that use the `s3` scheme, specify the following cluster configuration properties:

```
[
  {
"Classification": "core-site",
    "Properties": {
    "fs.s3.impl": "org.apache.hadoop.fs.s3a.S3AFileSystem",
    "fs.AbstractFileSystem.s3.impl": "org.apache.hadoop.fs.s3a.S3A"
    }
  }
]
```

To use S3A, set the `fs.`*`file_scheme`*`.impl` configuration property to `org.apache.hadoop.fs.s3a.S3AFileSystem`, and set the `fs.AbstractFileSystem.`*`file_scheme`*`.impl` property to `org.apache.hadoop.fs.s3a.S3A`.

For example, if you are accessing the path `s3a://`*`bucket`*`/...`, set the `fs.s3a.impl` property to `org.apache.hadoop.fs.s3a.S3AFileSystem`, and set the `fs.AbstractFileSystem.s3a.impl` property to `org.apache.hadoop.fs.s3a.S3A`.

Getting started using Amazon EMR with Amazon S3 on Outposts

The following topics explain how to get started using Amazon EMR with Amazon S3 on Outposts.

Topics

- Create a permissions policy
- Create and configure your cluster

- Configurations overview
- Considerations

Create a permissions policy

Before you can create an EMR cluster that uses Amazon S3 on Outposts, you must create an IAM policy to attach to the Amazon EC2 instance profile for the cluster. The policy must have permissions to access the S3 on Outposts access point Amazon Resource Name (ARN). For more information about creating IAM policies for S3 on Outposts, see Setting up IAM with S3 on Outposts.

The following example policy shows how to grant the required permissions. After you create the policy, attach the policy to the instance profile role that you use to create your EMR cluster, as described in the the section called "Create and configure your cluster" section. To use this example, replace the *user input placeholders* with your own information.

```
{
"Version":"2012-10-17",
  "Statement": [
        {
  "Effect": "Allow",
            "Resource": "arn:aws:s3-outposts:us-
west-2:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/access-point-name",
            "Action": [
                "s3-outposts:*"
            ]
        }
    ]

 }
```

Create and configure your cluster

To create a cluster that runs Spark with S3 on Outposts, complete the following steps in the console.

To create a cluster that runs Spark with S3 on Outposts

1. Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/.
2. In the left navigation pane, choose **Clusters**.

3. Choose **Create cluster**.

4. For **Amazon EMR release**, choose **emr-7.0.0** or later.
5. For Application bundle, choose **Spark interactive**. Then select any other supported applications that you want to be included in your cluster.
6. To enable Amazon S3 on Outposts, enter your configuration settings.

   Sample configuration settings

   To use the following sample configuration settings, replace the *user input placeholders* with your own information.

```
[
 {
   "Classification": "core-site",
   "Properties": {
     "fs.s3a.bucket.DOC-EXAMPLE-BUCKET.accesspoint.arn": "arn:aws:s3-outposts:us-
west-2:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/access-point-name"
     "fs.s3a.committer.name": "magic",
     "fs.s3a.select.enabled": "false"
    }
  },
  {
    "Classification": "hadoop-env",
    "Configurations": [
      {
        "Classification": "export",
        "Properties": {
          "JAVA_HOME": "/usr/lib/jvm/java-11-amazon-corretto.x86_64"
          }
       }
     ],
     "Properties": {}
   },
   {
     "Classification": "spark-env",
     "Configurations": [
       {
         "Classification": "export",
         "Properties": {
           "JAVA_HOME": "/usr/lib/jvm/java-11-amazon-corretto.x86_64"
         }
```

```
       }
      ],
      "Properties": {}
     },
     {
      "Classification": "spark-defaults",
      "Properties": {
        "spark.executorEnv.JAVA_HOME": "/usr/lib/jvm/java-11-amazon-
corretto.x86_64",
        "spark.sql.sources.fastS3PartitionDiscovery.enabled": "false"
      }
     }
  ]
```

7. In the **Networking** section, choose a virtual private cloud (VPC) and subnet that are on your AWS Outposts rack. For more information about Amazon EMR on Outposts, see EMR clusters on AWS Outposts in the *Amazon EMR Management Guide*.
8. In the **EC2 instance profile for Amazon EMR** section, choose the IAM role that has the permissions policy that you created earlier attached.
9. Configure your remaining cluster settings, and then choose **Create cluster**.

Configurations overview

The following table describes S3A configurations and the values to specify for their parameters when you set up a cluster that uses S3 on Outposts with Amazon EMR.

Parameter	Default value	Required value for S3 on Outposts	Explanation
`fs.s3a.aws.credentials.provider`	If not specified, S3A will look for S3 in Region bucket with the Outposts bucket name.	The access point ARN of the S3 on Outposts bucket	Amazon S3 on Outposts supports virtual private cloud (VPC)-only access points as the only means to access your Outposts buckets.

Parameter	Default value	Required value for S3 on Outposts	Explanation
fs.s3a.committer.name	file	magic	Magic committer is the only supported committer for S3 on Outposts.
fs.s3a.select.enabled	TRUE	FALSE	S3 Select is not supported on Outposts.
JAVA_HOME	/usr/lib/jvm/java-8	/usr/lib/jvm/java-11-amazon-corretto.x86_64	S3 on Outposts on S3A requires Java version 11.

The following table describes Spark configurations and the values to specify for their parameters when you set up a cluster that uses S3 on Outposts with Amazon EMR.

Parameter	Default value	Required value for S3 on Outposts	Explanation
spark.sql.sources.fastS3PartitionDiscovery.enabled	TRUE	FALSE	S3 on Outposts doesn't support fast partition.
spark.executorEnv.JAVA_HOME	/usr/lib/jvm/java-8	/usr/lib/jvm/java-11-amazon-corretto.x86_64	S3 on Outposts on S3A requires Java version 11.

Considerations

Consider the following when you integrate Amazon EMR with S3 on Outposts buckets:

- Amazon S3 on Outposts is supported with Amazon EMR version 7.0.0 and later.
- The S3A connector is required to use S3 on Outposts with Amazon EMR. Only S3A has the features required to interact with S3 on Outposts buckets. For S3A connector setup information, see Prerequisites.
- Amazon S3 on Outposts supports only server-side encryption with Amazon S3 managed keys (SSE-S3) with Amazon EMR. For more information, see the section called “Data encryption”.
- Amazon S3 on Outposts doesn't support writes with the S3A FileOutputCommitter. Writes with the S3A FileOutputCommitter on S3 on Outposts buckets result in the following error: InvalidStorageClass: The storage class you specified is not valid.
- Amazon S3 on Outposts isn't supported with Amazon EMR Serverless or Amazon EMR on EKS.
- Amazon EMR logs are stored in a Regional Amazon S3 location that you select, and are not stored locally in the S3 on Outposts bucket.

Authorization and authentication caching

S3 on Outposts securely caches authentication and authorization data locally on Outposts racks. The cache removes round trips to the parent AWS Region for every request. This eliminates the variability that is introduced by network round trips. With the authentication and authorization cache in S3 on Outposts, you get consistent latencies that are independent from the latency of the connection between the Outposts and the AWS Region.

When you make an S3 on Outposts API request, the authentication and authorization data is securely cached. The cached data is then used to authenticate subsequent S3 object API requests. S3 on Outposts only caches authentication and authorization data when the request is signed using Signature Version 4A (SigV4A). The cache is stored locally on the Outposts within the S3 on Outposts service. It asynchronously refreshes when you make an S3 API request. The cache is encrypted, and no plaintext cryptographic keys are stored on Outposts.

The cache is valid for up to 10 minutes when the Outpost is connected to the AWS Region. It is refreshed asynchronously when you make an S3 on Outposts API request, to ensure that the latest policies are used. If the Outpost is disconnected from the AWS Region, the cache will be valid for up to 12 hours.

Configuring the authorization and authentication cache

S3 on Outposts automatically caches authentication and authorization data for requests signed with the SigV4A algorithm. For more information, see Signing AWS API requests in the *AWS Identity and Access Management User Guide*. The SigV4A algorithm is available in the latest versions of the AWS SDKs. You can obtain it through a dependency on the AWS Common Runtime (CRT) libraries.

You need to use the latest version of the AWS SDK and install the latest version of the CRT. For example, you can run `pip install awscrt` to obtain the latest version of the CRT with Boto3.

S3 on Outposts does not cache authentication and authorization data for requests signed with the SigV4 algorithm.

Validating SigV4A signing

You can use AWS CloudTrail to validate that requests were signed with SigV4A. For more information on setting up CloudTrail for S3 on Outposts, see Monitoring S3 on Outposts with AWS CloudTrail logs.

After you have configured CloudTrail, you can verify how a request was signed in the `SignatureVersion` field of the CloudTrail logs. Requests that were signed with SigV4A will have a `SignatureVersion` set to `AWS4-ECDSA-P256-SHA256`. Requests that were signed with SigV4 will have `SignatureVersion` set to `AWS4-HMAC-SHA256`.

Security in S3 on Outposts

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

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

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon S3 on Outposts, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using S3 on Outposts. The following topics show you how to configure S3 on Outposts to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your S3 on Outposts resources.

Topics

- Setting up IAM with S3 on Outposts
- Data encryption in S3 on Outposts
- AWS PrivateLink for S3 on Outposts
- AWS Signature Version 4 (SigV4) authentication-specific policy keys
- AWS managed policies for Amazon S3 on Outposts
- Using service-linked roles for Amazon S3 on Outposts

Setting up IAM with S3 on Outposts

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in)

and authorized (have permissions) to use Amazon S3 on Outposts resources. IAM is an AWS service that you can use with no additional charge. By default, users don't have permissions for S3 on Outposts resources and operations. To grant access permissions for S3 on Outposts resources and API operations, you can use IAM to create users, groups, or roles and attach permissions.

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.
- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Create a role for a third-party identity provider (federation) in the *IAM User Guide*.
- IAM users:
  - Create a role that your user can assume. Follow the instructions in Create a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

In addition to IAM identity-based policies, S3 on Outposts supports both bucket and access point policies. Bucket policies and access point policies are resource-based policies that are attached to the S3 on Outposts resource.

- A bucket policy is attached to the bucket and allows or denies requests to the bucket and the objects in it based on the elements in the policy.
- In contrast, an access point policy is attached to the access point and allows or denies requests to the access point.

The access point policy works with the bucket policy that is attached to the underlying S3 on Outposts bucket. For an application or user to access objects in an S3 on Outposts bucket through an S3 on Outposts access point, both the access point policy and the bucket policy must permit the request.

Restrictions that you include in an access point policy apply only to requests made through that access point. For example, if an access point is attached to a bucket, you can't use the access point

policy to allow or deny requests that are made directly to the bucket. However, restrictions that you apply to a bucket policy can allow or deny requests made directly to the bucket or through the access point.

In an IAM policy or a resource-based policy, you define which S3 on Outposts actions are allowed or denied. S3 on Outposts actions correspond to specific S3 on Outposts API operations. S3 on Outposts actions use the `s3-outposts:` namespace prefix. Requests made to the S3 on Outposts control API in an AWS Region and requests made to the object API endpoints on the Outpost are authenticated by using IAM and authorized against the `s3-outposts:` namespace prefix. To work with S3 on Outposts, configure your IAM users and authorize them against the `s3-outposts:` IAM namespace.

For more information, see Actions, resources, and condition keys for Amazon S3 on Outposts in the *Service Authorization Reference*.

Note

- Access control lists (ACLs) are not supported by S3 on Outposts.
- S3 on Outposts defaults to the bucket owner as object owner to help ensure that the owner of a bucket can't be prevented from accessing or deleting objects.
- S3 on Outposts always has S3 Block Public Access enabled to help ensure that objects can never have public access.

For more information about setting up IAM for S3 on Outposts, see the following topics.

Topics

- Principals for S3 on Outposts policies
- Resource ARNs for S3 on Outposts
- Example policies for S3 on Outposts
- Permissions for S3 on Outposts endpoints
- Service-linked roles for S3 on Outposts

Principals for S3 on Outposts policies

When you create a resource-based policy to grant access to your S3 on Outposts bucket, you must use the `Principal` element to specify the person or application that can make a request for an action or operation on that resource. For S3 on Outposts policies, you can use one of the following principals:

- An AWS account
- An IAM user
- An IAM role
- All principals, by specifying a wildcard character (*) in a policy that uses a `Condition` element to limit access to a specific IP range

> **Important**
>
> You can't write a policy for an S3 on Outposts bucket that uses a wildcard character (*) in the `Principal` element unless the policy also includes a `Condition` that limits access to a specific IP address range. This restriction helps ensure that there is no public access to your S3 on Outposts bucket. For an example, see Example policies for S3 on Outposts.

For more information about the `Principal` element, see AWS JSON policy elements: Principal in the *IAM User Guide*.

Resource ARNs for S3 on Outposts

Amazon Resource Names (ARNs) for S3 on Outposts contain the Outpost ID in addition to the AWS Region that the Outpost is homed to, the AWS account ID, and the resource name. To access and perform actions on your Outposts buckets and objects, you must use one of the ARN formats shown in the following table.

The *`partition`* value in the ARN refers to a group of AWS Regions. Each AWS account is scoped to one partition. The following are the supported partitions:

- `aws` – AWS Regions
- `aws-us-gov` – AWS GovCloud (US) Regions

The following table shows S3 on Outposts ARN formats.

Amazon S3 on Outposts ARN	ARN format	Example
Bucket ARN	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /bucket/*bucket_name*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /bucket/*amzn-s3-demo-bucket1*
Access point ARN	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /accesspoint/ *accesspoint_name*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /accesspoint/ *access-point-name*
Object ARN	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /bucket/*bucket_name* /object/*object_key*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /bucket/*amzn-s3-demo-bucket1 /object/myobject*
S3 on Outposts access point object ARN (used in policies)	arn:*partition* :s3-outposts: *region*: *account_id* :outpost/ *outpost_id* /accesspoint/ *accesspoint_name* /object/*object_key*	arn:*aws*:s3-outposts: *us-west-2* :*123456789012* :outpost/ *op-01ac5d28a6a232904* /accesspoint/ *access-point-name/object/myobject*

Amazon S3 on Outposts ARN	ARN format	Example
S3 on Outposts ARN	`arn:`*`partition`*`:s3-outposts:`*`region`*`:`*`account_id`*`:outpost/`*`outpost_id`*	`arn:`*`aws`*`:s3-outposts:`*`us-west-2`*`:`*`123456789012`*`:outpost/`*`op-01ac5d28a6a232904`*

Example policies for S3 on Outposts

Example: S3 on Outposts bucket policy with an AWS account principal

The following bucket policy uses an AWS account principal to grant access to an S3 on Outposts bucket. To use this bucket policy, replace the *`user input placeholders`* with your own information.

Example: S3 on Outposts bucket policy with a wildcard principal (*) and condition key to limit access to a specific IP address range

The following bucket policy uses a wildcard principal (*) with the `aws:SourceIp` condition to limit access to a specific IP address range. To use this bucket policy, replace the *`user input placeholders`* with your own information.

Permissions for S3 on Outposts endpoints

S3 on Outposts requires its own permissions in IAM to manage S3 on Outposts endpoint actions.

Note

- For endpoints that use the customer-owned IP address pool (CoIP pool) access type, you also must have permissions to work with IP addresses from your CoIP pool, as described in the following table.
- For shared accounts that access S3 on Outposts by using AWS Resource Access Manager, users in these shared accounts can't create their own endpoints on a shared subnet. If a user in a shared account wants to manage their own endpoints, the shared account

must create its own subnet on the Outpost. For more information, see the section called "Sharing S3 on Outposts".

The following table shows S3 on Outposts endpoint-related IAM permissions.

Action	IAM permissions

`CreateEndpoint`	`s3-outposts:CreateEndpoint`
	`ec2:CreateNetworkInterface`
	`ec2:DescribeNetworkInterfaces`
	`ec2:DescribeVpcs`
	`ec2:DescribeSecurityGroups`
	`ec2:DescribeSubnets`
	`ec2:CreateTags`
	`iam:CreateServiceLinkedRole`
	For endpoints that are using the on-premises customer-owned IP address pool (CoIP pool) access type, the following additional permissio ns are required:
	`s3-outposts:CreateEndpoint`
	`ec2:DescribeCoipPools`
	`ec2:GetCoipPoolUsage`
	`ec2:AllocateAddress`
	`ec2:AssociateAddress`
	`ec2:DescribeAddresses`

Action	IAM permissions
	`ec2:DescribeLocalGatewayRouteTableVpcAssociations`

`DeleteEndpoint`	`s3-outposts:DeleteEndpoint`
	
	`ec2:DeleteNetworkInterface`
	
	`ec2:DescribeNetworkInterfaces`
	
	For endpoints that are using the on-premises customer-owned IP address pool (CoIP pool) access type, the following additional permissions are required:
	
	`s3-outposts:DeleteEndpoint`
	
	`ec2:DisassociateAddress`
	
	`ec2:DescribeAddresses`
	
	`ec2:ReleaseAddress`

`ListEndpoints`	`s3-outposts:ListEndpoints`

Note

You can use resource tags in an IAM policy to manage permissions.

Service-linked roles for S3 on Outposts

S3 on Outposts uses IAM service-linked roles to create some network resources on your behalf. For more information, see Using service-linked roles for Amazon S3 on Outposts.

Data encryption in S3 on Outposts

By default, all data stored in Amazon S3 on Outposts is encrypted by using server-side encryption with Amazon S3 managed encryption keys (SSE-S3). For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3) in the *Amazon S3 User Guide*.

You can optionally use server-side encryption with customer-provided encryption keys (SSE-C). To use SSE-C, specify an encryption key as part of your object API requests. Server-side encryption encrypts only the object data, not the object metadata. For more information, see Using server-side encryption with customer-provided keys in the *Amazon S3 User Guide*.

> **Note**
>
> S3 on Outposts doesn't support server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS).

AWS PrivateLink for S3 on Outposts

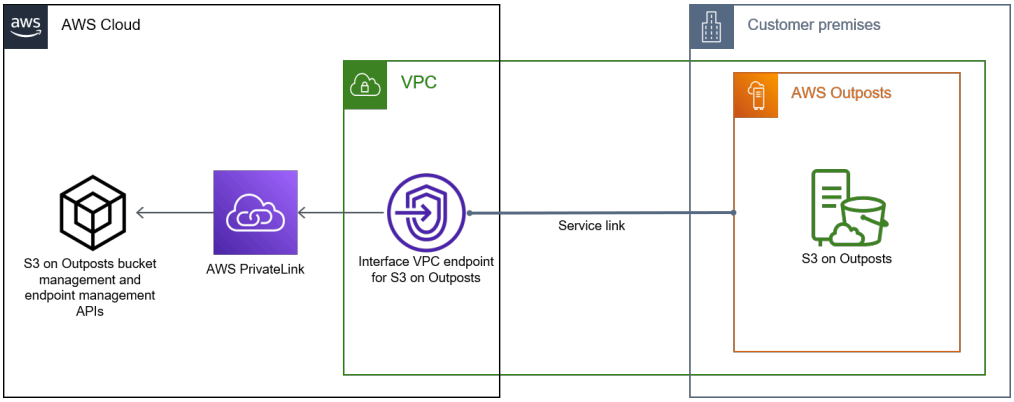
S3 on Outposts supports AWS PrivateLink, which provides direct management access to your S3 on Outposts storage through a private endpoint within your virtual private network. This allows you to simplify your internal network architecture and perform management operations on your Outposts object storage by using private IP addresses in your Virtual Private Cloud (VPC). Using AWS PrivateLink eliminates the need to use public IP addresses or proxy servers.

With AWS PrivateLink for Amazon S3 on Outposts, you can provision *interface VPC endpoints* in your virtual private cloud (VPC) to access your S3 on Outposts bucket management and endpoint management APIs. Interface VPC endpoints are directly accessible from applications deployed in your VPC or on premises over your virtual private network (VPN) or AWS Direct Connect. You can access the bucket and endpoint management APIs through AWS PrivateLink. AWS PrivateLink doesn't support data transfer API operations, such as GET, PUT, and similar APIs. These operations are already transferred privately through the S3 on Outposts endpoint and access point configuration. For more information, see Networking for S3 on Outposts.

Interface endpoints are represented by one or more elastic network interfaces (ENIs) that are assigned private IP addresses from subnets in your VPC. Requests made to interface endpoints for S3 on Outposts are automatically routed to S3 on Outposts bucket and endpoint management APIs on the AWS network. You can also access interface endpoints in your VPC from on-premises

applications through AWS Direct Connect or AWS Virtual Private Network (Site-to-Site VPN). For more information about how to connect your VPC with your on-premises network, see the *Direct Connect User Guide* and the *AWS Site-to-Site VPN User Guide*.

Interface endpoints route requests for S3 on Outposts bucket and endpoint management APIs over the AWS network and through AWS PrivateLink, as illustrated in the following diagram.



For general information about interface endpoints, see Interface VPC endpoints (AWS PrivateLink) in the *AWS PrivateLink Guide*.

Topics

- Restrictions and limitations
- Accessing S3 on Outposts interface endpoints
- Updating an on-premises DNS configuration
- Creating a VPC endpoint for S3 on Outposts
- Creating bucket policies and VPC endpoint policies for S3 on Outposts

Restrictions and limitations

When you access S3 on Outposts bucket and endpoint management APIs through AWS PrivateLink, VPC limitations apply. For more information, see Interface endpoint properties and limitations and AWS PrivateLink quotas in the *AWS PrivateLink Guide*.

In addition, AWS PrivateLink doesn't support the following:

- Federal Information Processing Standard (FIPS) endpoints
- S3 on Outposts data transfer APIs, for example, GET, PUT, and similar object API operations.
- Private DNS

Accessing S3 on Outposts interface endpoints

To access S3 on Outposts bucket and endpoint management APIs using AWS PrivateLink, you *must* update your applications to use endpoint-specific DNS names. When you create an interface endpoint, AWS PrivateLink generates two types of endpoint-specific S3 on Outposts names: *Regional* and *zonal*.

- **Regional DNS names** – include a unique VPC endpoint ID, a service identifier, the AWS Region, and `vpce.amazonaws.com`, for example, *`vpce-1a2b3c4d-5e6f`*`.s3-outposts.us-east-1.vpce.amazonaws.com`.
- **Zonal DNS names** – include a unique VPC endpoint ID, the Availability Zone, a service identifier, the AWS Region, and `vpce.amazonaws.com`, for example, *`vpce-1a2b3c4d-5e6f`*`-us-east-1a.s3-outposts.us-east-1.vpce.amazonaws.com`. You might use this option if your architecture isolates Availability Zones. For example, you could use zonal DNS names for fault containment or to reduce Regional data transfer costs.

> **Important**
>
> S3 on Outposts interface endpoints are resolved from the public DNS domain. S3 on Outposts does not support private DNS. Use the `--endpoint-url` parameter for all bucket and endpoint management APIs.

AWS CLI examples

Use the `--region` and `--endpoint-url` parameters to access bucket management and endpoint management APIs through S3 on Outposts interface endpoints.

Example: Use the endpoint URL to list buckets with the S3 control API

In the following example, replace the Region *`us-east-1`*, VPC endpoint URL *`vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com`*, and account ID *`111122223333`* with appropriate information.

```
aws s3control list-regional-buckets --region us-east-1 --endpoint-url
 https://vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com --account-
id 111122223333
```

AWS SDK examples

Update your SDKs to the latest version, and configure your clients to use an endpoint URL for accessing the S3 control API for S3 on Outposts interface endpoints.

SDK for Python (Boto3)

Example: Use an endpoint URL to access the S3 control API

In the following example, replace the Region *us-east-1* and VPC endpoint URL *vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com* with appropriate information.

```
control_client = session.client(
service_name='s3control',
region_name='us-east-1',
endpoint_url='https://vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com'
)
```

For more information, see AWS PrivateLink for Amazon S3 in the *Boto3 developer guide*.

SDK for Java 2.x

Example: Use an endpoint URL to access the S3 control API

In the following example, replace the VPC endpoint URL *vpce-1a2b3c4d-5e6f.s3-outposts.us-east-1.vpce.amazonaws.com* and the Region *Region.US_EAST_1* with appropriate information.

```
// control client
Region region = Region.US_EAST_1;
s3ControlClient = S3ControlClient.builder().region(region)

 .endpointOverride(URI.create("https://vpce-1a2b3c4d-5e6f.s3-outposts.us-
east-1.vpce.amazonaws.com"))
                                  .build()
```

For more information, see S3ControlClient in the *AWS SDK for Java API Reference*.

Updating an on-premises DNS configuration

When using endpoint-specific DNS names to access the interface endpoints for S3 on Outposts bucket management and endpoint management APIs, you don't have to update your on-premises DNS resolver. You can resolve the endpoint-specific DNS name with the private IP address of the interface endpoint from the public S3 on Outposts DNS domain.

Creating a VPC endpoint for S3 on Outposts

To create a VPC interface endpoint for S3 on Outposts, see Create a VPC endpoint in the *AWS PrivateLink Guide*.

Creating bucket policies and VPC endpoint policies for S3 on Outposts

You can attach an endpoint policy to your VPC endpoint that controls access to S3 on Outposts. You can also use the `aws:sourceVpce` condition in S3 on Outposts bucket policies to restrict access to specific buckets from a specific VPC endpoint. With VPC endpoint policies, you can control access to S3 on Outposts bucket management APIs and endpoint management APIs. With bucket policies, you can control access to the S3 on Outposts bucket management APIs. However, you can't manage access to object actions for S3 on Outposts using `aws:sourceVpce`.

Access policies for S3 on Outposts specify the following information:

- The AWS Identity and Access Management (IAM) principal for which actions are allowed or denied.
- The S3 control actions that are allowed or denied.
- The S3 on Outposts resources on which actions are allowed or denied.

The following examples show policies that restrict access to a bucket or to an endpoint. For more information about VPC connectivity, see Network-to-VPC connectivity options in the AWS whitepaper Amazon Virtual Private Cloud Connectivity Options.

Important

- When you apply the example policies for VPC endpoints described in this section, you might block your access to the bucket without intending to do so. Bucket permissions that limit bucket access to connections originating from your VPC endpoint can block

all connections to the bucket. For information about how to fix this issue, see My bucket policy has the wrong VPC or VPC endpoint ID. How can I fix the policy so that I can access the bucket? in the *Support Knowledge Center*.

- Before using the following example bucket policies, replace the VPC endpoint ID with an appropriate value for your use case. Otherwise, you won't be able to access your bucket.
- If your policy only allows access to an S3 on Outposts bucket from a specific VPC endpoint, it disables console access for that bucket because console requests don't originate from the specified VPC endpoint.

Topics

- Example: Restricting access to a specific bucket from a VPC endpoint
- Example: Denying access from a specific VPC endpoint in an S3 on Outposts bucket policy

Example: Restricting access to a specific bucket from a VPC endpoint

You can create an endpoint policy that restricts access to specific S3 on Outposts buckets only. The following policy restricts access for the GetBucketPolicy action only to the *example-outpost-bucket*. To use this policy, replace the example values with your own.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Policy1415115909151",
    "Statement": [
        {
            "Sid": "Access-to-specific-bucket-only",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3-outposts:GetBucketPolicy",
            "Effect": "Allow",
            "Resource": "arn:aws:s3-outposts:us-
east-1:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket"
        }
    ]
}
```

Example: Denying access from a specific VPC endpoint in an S3 on Outposts bucket policy

The following S3 on Outposts bucket policy denies access to GetBucketPolicy on the *example-outpost-bucket* bucket through the *vpce-1a2b3c4d* VPC endpoint.

The `aws:sourceVpce` condition specifies the endpoint and does not require an Amazon Resource Name (ARN) for the VPC endpoint resource, only the endpoint ID. To use this policy, replace the example values with your own.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Policy1415115909152",
    "Statement": [
        {
            "Sid": "Deny-access-to-specific-VPCE",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3-outposts:GetBucketPolicy",
            "Effect": "Deny",
            "Resource": "arn:aws:s3-outposts:us-
east-1:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket",
            "Condition": {
                "StringEquals": {
                    "aws:sourceVpce": "vpce-1a2b3c4d"
                }
            }
        }
    ]
}
```

AWS Signature Version 4 (SigV4) authentication-specific policy keys

The following table shows the condition keys related to AWS Signature Version 4 (SigV4) authentication that you can use with Amazon S3 on Outposts. In a bucket policy, you can add

these conditions to enforce specific behavior when requests are authenticated by using Signature Version 4. For example policies, see Bucket policy examples that use Signature Version 4-related condition keys. For more information about authenticating requests using Signature Version 4, see Authenticating requests (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*

Applicable keys	Description

`s3-outposts:authType`	S3 on Outposts supports various methods of authentication. To restrict incoming requests to use a specific authentication method, you can use this optional condition key. For example, you can use this condition key to allow only the HTTP `Authorization` header to be used in request authentication.
	
	Valid values:
	
	`REST-HEADER`
	
	`REST-QUERY-STRING`

`s3-outposts:signatureAge`	The length of time, in milliseconds, that a signature is valid in an authenticated request. This condition works only for presigned URLs. In Signature Version 4, the signing key is valid for up to seven days. Therefore, the signatures are also valid for up to seven days. For more information, see Introduction to signing requests in the *Amazon Simple Storage Service API Reference*. You can use this condition to further limit the signature age. Example value: `600000`
`s3-outposts:x-amz-content-sha256`	You can use this condition key to disallow unsigned content in your bucket. When you use Signature Version 4, for requests that use the `Authorization` header, you add the `x-amz-content-sha256` header in the signature calculation and then set its value to the hash payload.

Applicable keys	Description
	You can use this condition key in your bucket policy to deny any uploads where the payloads are not signed. For example: • Deny uploads that use the `Authorization` header to authenticate requests but don't sign the payload. For more information, see Transferring payload in a single chunk in the *Amazon Simple Storage Service API Reference*. • Deny uploads that use presigned URLs. Presigned URLs always have an `UNSIGNED_PAYLOAD`. For more information, see Authenticating requests and Authentication methods in the *Amazon Simple Storage Service API Reference*. Valid value: `UNSIGNED-PAYLOAD`

Bucket policy examples that use Signature Version 4-related condition keys

To use the following examples, replace the *`user input placeholders`* with your own information.

Example: s3-outposts:signatureAge

The following bucket policy denies any S3 on Outposts presigned URL request on objects in `example-outpost-bucket` if the signature is more than 10 minutes old.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Deny a presigned URL request if the signature is more than 10
 minutes old",
            "Effect": "Deny",
            "Principal": {"AWS":"444455556666"},
            "Action": "s3-outposts:*",
```

```
            "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/
object/*",
            "Condition": {
                "NumericGreaterThan": {"s3-outposts:signatureAge": 600000},
                "StringEquals": {"s3-outposts:authType": "REST-QUERY-STRING"}
            }
        }
    ]
}
```

Example: s3-outposts:authType

The following bucket policy allows only requests that use the `Authorization` header for request authentication. Any presigned URL requests will be denied since presigned URLs use query parameters to provide request and authentication information. For more information, see Authentication methods in the *Amazon Simple Storage Service API Reference*.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
         {
               "Sid": "Allow only requests that use the Authorization header for
 request authentication. Deny presigned URL requests.",
               "Effect": "Deny",
               "Principal": {"AWS":"111122223333"},
               "Action": "s3-outposts:*",
               "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/
object/*",
               "Condition": {
                     "StringNotEquals": {
                           "s3-outposts:authType": "REST-HEADER"
                     }
               }
         }
   ]
}
```

Example: s3-outposts:x-amz-content-sha256

The following bucket policy denies any uploads with unsigned payloads, such as uploads that are using presigned URLs. For more information, see Authenticating requests and Authentication methods in the *Amazon Simple Storage Service API Reference*.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
         {
               "Sid": "Deny uploads with unsigned payloads.",
               "Effect": "Deny",
               "Principal": {"AWS":"111122223333"},
               "Action": "s3-outposts:*",
               "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/bucket/example-outpost-bucket/
object/*",
               "Condition": {
                     "StringEquals": {
                           "s3-outposts:x-amz-content-sha256": "UNSIGNED-PAYLOAD"
                     }
               }
         }
   ]
}
```

AWS managed policies for Amazon S3 on Outposts

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AWSS3OnOutpostsServiceRolePolicy

Helps manage network resources for you as part of the service-linked role `AWSServiceRoleForS3OnOutposts`.

To view the permissions for this policy, see AWSS3OnOutpostsServiceRolePolicy.

S3 on Outposts updates to AWS managed policies

View details about updates to AWS managed policies for S3 on Outposts since this service began tracking these changes.

Change	Description	Date
S3 on Outposts added `AWSS3OnOutpostsServiceRolePolicy`	S3 on Outposts added `AWSS3OnOutpostsServiceRolePolicy` as part of the service-linked role `AWSServiceRoleForS3OnOutposts`, which helps manage network resources for you.	October 3, 2023
S3 on Outposts started tracking changes	S3 on Outposts started tracking changes for its AWS managed policies.	October 3, 2023

Using service-linked roles for Amazon S3 on Outposts

Amazon S3 on Outposts uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to S3 on Outposts. Service-linked roles are predefined by S3 on Outposts and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up S3 on Outposts easier because you don't have to manually add the necessary permissions. S3 on Outposts defines the permissions of its service-linked roles, and unless defined otherwise, only S3 on Outposts can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your S3 on Outposts resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have **Yes** in the **Service-linked roles** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for S3 on Outposts

S3 on Outposts uses the service-linked role named **AWSServiceRoleForS3OnOutposts** to help manage network resources for you.

The `AWSServiceRoleForS3OnOutposts` service-linked role trusts the following services to assume the role:

- `s3-outposts.amazonaws.com`

The role permissions policy named `AWSS3OnOutpostsServiceRolePolicy` allows S3 on Outposts to complete the following actions on the specified resources:

JSON

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

```
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeSubnets",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcs",
                "ec2:DescribeCoipPools",
                "ec2:GetCoipPoolUsage",
                "ec2:DescribeAddresses",
                "ec2:DescribeLocalGatewayRouteTableVpcAssociations"
            ],
            "Resource": "*",
            "Sid": "DescribeVpcResources"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:security-group/*"
            ],
            "Sid": "CreateNetworkInterface"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateNetworkInterface"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:network-interface/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "CreateTagsForCreateNetworkInterface"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AllocateAddress"
```

```
            ],
            "Resource": [
                "arn:aws:ec2:*:*:ipv4pool-ec2/*"
            ],
            "Sid": "AllocateIpAddress"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AllocateAddress"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:elastic-ip/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "CreateTagsForAllocateIpAddress"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:CreateNetworkInterfacePermission",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteNetworkInterfacePermission",
                "ec2:DisassociateAddress",
                "ec2:ReleaseAddress",
                "ec2:AssociateAddress"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CreatedBy": "S3 On Outposts"
                }
            },
            "Sid": "ReleaseVpcResources"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
```

```
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": [
                        "CreateNetworkInterface",
                        "AllocateAddress"
                    ],
                    "aws:RequestTag/CreatedBy": [
                        "S3 On Outposts"
                    ]
                }
            },
            "Sid": "CreateTags"
        }
    ]
}
```

You must configure permissions to allow an IAM entity (such as a role) to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating a service-linked role for S3 on Outposts

You don't need to manually create a service-linked role. When you create an S3 on Outposts endpoint in the AWS Management Console, the AWS CLI, or the AWS API, S3 on Outposts creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you create an S3 on Outposts endpoint, S3 on Outposts creates the service-linked role for you again.

You can also use the IAM console to create a service-linked role with the **S3 on Outposts** use case. In the AWS CLI or the AWS API, create a service-linked role with the `s3-outposts.amazonaws.com` service name. For more information, see Creating a service-linked role in the *IAM User Guide*. If you delete this service-linked role, you can use this same process to create the role again.

Editing a service-linked role for S3 on Outposts

S3 on Outposts does not allow you to edit the `AWSServiceRoleForS3OnOutposts` service-linked role. This includes the name of the role because various entities might reference it. However,

you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting a service-linked role for S3 on Outposts

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

> **Note**
>
> If the S3 on Outposts service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To delete S3 on Outposts resources used by the AWSServiceRoleForS3OnOutposts role

1. Delete the S3 on Outposts endpoints in your AWS account across all AWS Regions.
2. Delete the service-linked role using IAM.

   Use the IAM console, the AWS CLI, or the AWS API to delete the `AWSServiceRoleForS3OnOutposts` service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported Regions for S3 on Outposts service-linked roles

S3 on Outposts supports using service-linked roles in all of the AWS Regions where the service is available. For more information, see S3 on Outposts Regions and endpoints.

Managing S3 on Outposts storage

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

For more information about managing and sharing your Amazon S3 on Outposts storage capacity, see the following topics.

Topics

- Managing S3 Versioning for your S3 on Outposts bucket
- Creating and managing a lifecycle configuration for your Amazon S3 on Outposts bucket
- Replicating objects for S3 on Outposts
- Sharing S3 on Outposts by using AWS RAM
- Other AWS services that use S3 on Outposts

Managing S3 Versioning for your S3 on Outposts bucket

When enabled, S3 Versioning saves multiple distinct copies of an object in the same bucket. You can use S3 Versioning to preserve, retrieve, and restore every version of every object stored in your Outposts buckets. S3 Versioning helps you recover from unintended user actions and application failures.

Amazon S3 on Outposts buckets have three versioning states:

- **Unversioned** – If you've never enabled or suspended S3 Versioning on your bucket, it is unversioned and returns no S3 Versioning status. For more information about S3 Versioning, see Managing S3 Versioning for your S3 on Outposts bucket.

- **Enabled** – Enables S3 Versioning for the objects in the bucket. All objects added to the bucket receive a unique version ID. Objects that already existed in the bucket at the time that you enable versioning have a version ID of `null`. If you modify these (or any other) objects with other operations, such as PutObject, the new objects get a unique version ID.
- **Suspended** – Suspends S3 Versioning for the objects in the bucket. All objects added to the bucket after versioning is suspended receive the version ID `null`. For more information, see Adding objects to versioning-suspended buckets in the *Amazon S3 User Guide*.

After you enable S3 Versioning for an S3 on Outposts bucket, it can never return to an unversioned state. However, you can suspend versioning. For more information about S3 Versioning, see Managing S3 Versioning for your S3 on Outposts bucket.

For each object in your bucket, you have a current version and zero or more noncurrent versions. To reduce storage costs, you can configure your bucket S3 Lifecycle rules to expire noncurrent versions after a specified time period. For more information, see Creating and managing a lifecycle configuration for your Amazon S3 on Outposts bucket.

The following examples show you how to enable or suspend versioning for an existing S3 on Outposts bucket by using the AWS Management Console and the AWS Command Line Interface (AWS CLI). To create a bucket with S3 Versioning enabled, see Creating an S3 on Outposts bucket.

> **Note**
>
> The AWS account that creates the bucket owns it and is the only one that can commit actions to it. Buckets have configuration properties, such as Outpost, tag, default encryption, and access point settings. The access point settings include the virtual private cloud (VPC), the access point policy for accessing the objects in the bucket, and other metadata. For more information, see S3 on Outposts specifications.

Using the S3 console

To edit the S3 Versioning settings for your bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to enable S3 Versioning for.

4. Choose the **Properties** tab.
5. Under **Bucket Versioning**, choose **Edit**.
6. Edit the S3 Versioning settings for the bucket by choosing one of the following options:
   - To suspend S3 Versioning and stop the creation of new object versions, choose **Suspend**.
   - To enable S3 Versioning and save multiple distinct copies of each object, choose **Enable**.
7. Choose **Save changes**.

Using the AWS CLI

To enable or suspend S3 Versioning for your bucket by using the AWS CLI, use the `put-bucket-versioning` command, as shown in the following examples. To use these examples, replace each *`user input placeholder`* with your own information.

For more information, see put-bucket-versioning in the *AWS CLI Reference*.

Example: To enable S3 Versioning

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket --versioning-configuration Status=Enabled
```

Example: To suspend S3 Versioning

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/bucket/example-outposts-bucket --versioning-configuration Status=Suspended
```

Creating and managing a lifecycle configuration for your Amazon S3 on Outposts bucket

You can use S3 Lifecycle to optimize storage capacity for Amazon S3 on Outposts. You can create lifecycle rules to expire objects as they age or are replaced by newer versions. You can create, enable, disable, or delete a lifecycle rule.

For more information about S3 Lifecycle, see Creating and managing a lifecycle configuration for your Amazon S3 on Outposts bucket.

> **Note**
>
> The AWS account that creates the bucket owns it and is the only one that can create, enable, disable, or delete a lifecycle rule.

To create and manage the lifecycle configuration for your S3 on Outposts bucket, see the following topics.

Topics

- Creating and managing a lifecycle rule by using the AWS Management Console
- Creating and managing a lifecycle configuration by using the AWS CLI and SDK for Java

Creating and managing a lifecycle rule by using the AWS Management Console

You can use S3 Lifecycle to optimize storage capacity for Amazon S3 on Outposts. You can create lifecycle rules to expire objects as they age or are replaced by newer versions. You can create, enable, disable, or delete a lifecycle rule.

For more information about S3 Lifecycle, see Creating and managing a lifecycle configuration for your Amazon S3 on Outposts bucket.

> **Note**
>
> The AWS account that creates the bucket owns it and is the only one that can create, enable, disable, or delete a lifecycle rule.

To create and manage a lifecycle rule for an S3 on Outposts by using the AWS Management Console, see the following topics.

Topics

- Creating a lifecycle rule
- Enabling a lifecycle rule
- Editing a lifecycle rule

- Deleting a lifecycle rule

Creating a lifecycle rule

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to create a lifecycle rule for.
4. Choose the **Management** tab, and then choose **Create Lifecycle rule**.
5. Enter a value for **Lifecycle rule name**.
6. Under **Rule scope**, choose one of the following options:

   - To limit the scope to specific filters, choose **Limit the scope of this rule using one or more filters**. Then, add a prefix filter, tags, or object size.
   - To apply the rule to all objects in the bucket, choose **Apply to all objects in the bucket**.
7. Under **Lifecycle rule actions**, choose one of the following options:

   - **Expire current versions of objects** – For versioning-enabled buckets, S3 on Outposts adds a delete marker and retains the objects as noncurrent versions. For buckets that don't use S3 Versioning, S3 on Outposts permanently deletes the objects.
   - **Permanently delete noncurrent versions of objects** – S3 on Outposts permanently deletes noncurrent versions of objects.
   - **Delete expired object delete markers or incomplete multipart uploads** – S3 on Outposts permanently deletes expired object delete markers or incomplete multipart uploads.

     If you limit the scope of your Lifecycle rule by using object tags, you can't choose **Delete expired object delete markers**. You also can't choose **Delete expired object delete markers** if you choose **Expire current object versions**.

   > **Note**
   >
   > Size-based filters can't be used with delete markers and incomplete multipart uploads.
8. If you chose **Expire current versions of objects** or **Permanently delete noncurrent versions of objects**, configure the rule trigger based on a specific date or the object's age.

9. If you chose **Delete expired object delete markers**, to confirm that you want to delete expired object delete markers, select **Delete expired object delete markers**.
10. Under **Timeline Summary**, review your Lifecycle rule, and choose **Create rule**.

Enabling a lifecycle rule

To enable or disable a bucket lifecycle rule

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to enable or disable a lifecycle rule for.
4. Choose the **Management** tab, and then under **Lifecycle rule**, choose the rule that you want to enable or disable.
5. For **Action**, choose **Enable or disable rule**.

Editing a lifecycle rule

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to edit a lifecycle rule for.
4. Choose the **Management** tab, and then choose the **Lifecycle rule** that you want to edit.
5. (Optional) Update the value for **Lifecycle rule name**.
6. Under **Rule scope**, edit the scope as needed:
   - To limit the scope to specific filters, choose **Limit the scope of this rule using one or more filters**. Then, add a prefix filter, tags, or object size.
   - To apply the rule to all objects in the bucket, choose **Apply to all objects in the bucket**.
7. Under **Lifecycle rule actions**, choose one of the following options:
   - **Expire current versions of objects** – For versioning-enabled buckets, S3 on Outposts adds a delete marker and retains the objects as noncurrent versions. For buckets that don't use S3 Versioning, S3 on Outposts permanently deletes the objects.
   - **Permanently delete noncurrent versions of objects** – S3 on Outposts permanently deletes noncurrent versions of objects.

- **Delete expired object delete markers or incomplete multipart uploads** – S3 on Outposts permanently deletes expired object delete markers or incomplete multipart uploads.

  If you limit the scope of your Lifecycle rule by using object tags, you can't choose **Delete expired object delete markers**. You also can't choose **Delete expired object delete markers** if you choose **Expire current object versions**.

> **Note**
>
> Size-based filters can't be used with delete markers and incomplete multipart uploads.

8. If you chose **Expire current versions of objects** or **Permanently delete noncurrent versions of objects**, configure the rule trigger based on a specific date or the object age.
9. If you chose **Delete expired object delete markers**, to confirm that you want to delete expired object delete markers, select **Delete expired object delete markers**.
10. Choose **Save**.

Deleting a lifecycle rule

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the Outposts bucket that you want to delete a lifecycle rule for.
4. Choose the **Management** tab, and then under **Lifecycle rule**, choose the rule that you want to delete.
5. Choose **Delete**.

Creating and managing a lifecycle configuration by using the AWS CLI and SDK for Java

You can use S3 Lifecycle to optimize storage capacity for Amazon S3 on Outposts. You can create lifecycle rules to expire objects as they age or are replaced by newer versions. You can create, enable, disable, or delete a lifecycle rule.

For more information about S3 Lifecycle, see Creating and managing a lifecycle configuration for your Amazon S3 on Outposts bucket.

> **Note**
>
> The AWS account that creates the bucket owns it and is the only one that can create, enable, disable, or delete a lifecycle rule.

To create and manage a lifecycle configuration for an S3 on Outposts bucket by using the AWS Command Line Interface (AWS CLI) and the AWS SDK for Java, see the following examples.

Topics

- PUT a lifecycle configuration
- GET the lifecycle configuration on an S3 on Outposts bucket

PUT a lifecycle configuration

AWS CLI

The following AWS CLI example puts a lifecycle configuration policy on an Outposts bucket. This policy specifies that all objects that have the flagged prefix (*myprefix*) and tags expire after 10 days. To use this example, replace each *user input placeholder* with your own information.

1. Save the lifecycle configuration policy to a JSON file. In this example, the file is named `lifecycle1.json`.

```
{
    "Rules": [
        {
            "ID": "id-1",
            "Filter": {
                "And": {
                    "Prefix": "myprefix",
                    "Tags": [
                        {
                            "Value": "mytagvalue1",
                            "Key": "mytagkey1"
                        },
                        {
                            "Value": "mytagvalue2",
                            "Key": "mytagkey2"
```

```
                        }
                    ],
                    "ObjectSizeGreaterThan": 1000,
                    "ObjectSizeLessThan": 5000
                }
            },
            "Status": "Enabled",
            "Expiration": {
                "Days": 10
            }
        }
    ]
}
```

2. Submit the JSON file as part of the `put-bucket-lifecycle-configuration` CLI command. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see put-bucket-lifecycle-configuration in the *AWS CLI Reference*.

```
aws s3control put-bucket-lifecycle-configuration --account-id 123456789012 --
bucket arn:aws:s3-outposts:region:123456789012:outpost/op-01ac5d28a6a232904/
bucket/example-outposts-bucket --lifecycle-configuration file://lifecycle1.json
```

SDK for Java

The following SDK for Java example puts a lifecycle configuration on an Outposts bucket. This lifecycle configuration specifies that all objects that have the flagged prefix (*`myprefix`*) and tags expire after 10 days. To use this example, replace each *`user input placeholder`* with your own information. For more information, see PutBucketLifecycleConfiguration in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.services.s3control.model.*;

public void putBucketLifecycleConfiguration(String bucketArn) {

    S3Tag tag1 = new S3Tag().withKey("mytagkey1").withValue("mytagkey1");
    S3Tag tag2 = new S3Tag().withKey("mytagkey2").withValue("mytagkey2");

    LifecycleRuleFilter lifecycleRuleFilter = new LifecycleRuleFilter()
            .withAnd(new LifecycleRuleAndOperator()
                    .withPrefix("myprefix")
```

```
                    .withTags(tag1, tag2))
                    .withObjectSizeGreaterThan(1000)
                    .withObjectSizeLessThan(5000);

    LifecycleExpiration lifecycleExpiration = new LifecycleExpiration()
            .withExpiredObjectDeleteMarker(false)
            .withDays(10);

    LifecycleRule lifecycleRule = new LifecycleRule()
            .withStatus("Enabled")
            .withFilter(lifecycleRuleFilter)
            .withExpiration(lifecycleExpiration)
            .withID("id-1");

    LifecycleConfiguration lifecycleConfiguration = new LifecycleConfiguration()
            .withRules(lifecycleRule);

    PutBucketLifecycleConfigurationRequest reqPutBucketLifecycle = new
 PutBucketLifecycleConfigurationRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn)
            .withLifecycleConfiguration(lifecycleConfiguration);

    PutBucketLifecycleConfigurationResult respPutBucketLifecycle =
 s3ControlClient.putBucketLifecycleConfiguration(reqPutBucketLifecycle);
    System.out.printf("PutBucketLifecycleConfiguration Response: %s%n",
 respPutBucketLifecycle.toString());

}
```

GET the lifecycle configuration on an S3 on Outposts bucket

AWS CLI

The following AWS CLI example gets a lifecycle configuration on an Outposts bucket. To use this command, replace each *`user input placeholder`* with your own information. For more information about this command, see get-bucket-lifecycle-configuration in the *AWS CLI Reference*.

```
aws s3control get-bucket-lifecycle-configuration --account-id 123456789012 --bucket
 arn:aws:s3-outposts:<your-region>:123456789012:outpost/op-01ac5d28a6a232904/
bucket/example-outposts-bucket
```

SDK for Java

The following SDK for Java example gets a lifecycle configuration for an Outposts bucket. For more information, see GetBucketLifecycleConfiguration in the *Amazon Simple Storage Service API Reference*.

```
import com.amazonaws.services.s3control.model.*;

public void getBucketLifecycleConfiguration(String bucketArn) {

    GetBucketLifecycleConfigurationRequest reqGetBucketLifecycle = new
 GetBucketLifecycleConfigurationRequest()
            .withAccountId(AccountId)
            .withBucket(bucketArn);

    GetBucketLifecycleConfigurationResult respGetBucketLifecycle =
 s3ControlClient.getBucketLifecycleConfiguration(reqGetBucketLifecycle);
    System.out.printf("GetBucketLifecycleConfiguration Response: %s%n",
 respGetBucketLifecycle.toString());

}
```

Replicating objects for S3 on Outposts

With S3 Replication on AWS Outposts, you can configure Amazon S3 on Outposts to automatically replicate S3 objects across different Outposts, or between buckets on the same Outpost. You can use S3 Replication on Outposts to maintain multiple replicas of your data in the same or different Outposts, or across different accounts, to help meet data-residency needs. S3 Replication on Outposts helps power your compliant storage needs and data sharing across accounts. If you need to ensure that your replicas are identical to the source data, you can use S3 Replication on Outposts to make replicas of your objects that retain all metadata, such as the original object creation time, tags, and version IDs.

S3 Replication on Outposts also provides detailed metrics and notifications to monitor the status of object replication between buckets. You can use Amazon CloudWatch to monitor

replication progress by tracking bytes pending replication, operations pending replication, and replication latency between your source and destination buckets. To quickly diagnose and correct configuration issues, you can also set up Amazon EventBridge to receive notifications about replication object failures. To learn more, see Managing your replication.

Topics

- Replication configuration
- Requirements for S3 Replication on Outposts
- What is replicated?
- What isn't replicated?
- What isn't supported by S3 Replication on Outposts?
- Setting up replication
- Managing your replication

Replication configuration

S3 on Outposts stores a replication configuration as XML. In the replication configuration XML file, you specify an AWS Identity and Access Management (IAM) role and one or more rules.

```
<ReplicationConfiguration>
    <Role>IAM-role-ARN</Role>
    <Rule>
        ...
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
</ReplicationConfiguration>
```

S3 on Outposts can't replicate objects without your permission. You grant S3 on Outposts permissions with the IAM role that you specify in the replication configuration. S3 on Outposts assumes that IAM role to replicate objects on your behalf. You must grant the required permissions to the IAM role before starting replication. For more information about these permissions for S3 on Outposts, see Creating an IAM role.

You add one rule in a replication configuration in the following scenarios:

- You want to replicate all objects.
- You want to replicate one subset of objects. You identify the object subset by adding a filter in the rule. In the filter, you specify an object key prefix, tags, or a combination of both, to identify the subset of objects that the rule applies to.

You add multiple rules in a replication configuration if you want to replicate different subsets of objects. In each rule, you specify a filter that selects a different subset of objects. For example, you might choose to replicate objects that have either `tax/` or `document/` key prefixes. To do this, you add two rules, one that specifies the `tax/` key prefix filter and another that specifies the `document/` key prefix.

For more information about S3 on Outposts replication configuration and replication rules, see ReplicationConfiguration in the *Amazon Simple Storage Service API Reference*.

Requirements for S3 Replication on Outposts

Replication requires the following:

- The destination Outpost CIDR range must be associated in your source Outpost subnet table. For more information, see Prerequisites for creating replication rules.
- Both the source and destination buckets must have S3 Versioning enabled. For more information about versioning, see Managing S3 Versioning for your S3 on Outposts bucket.
- Amazon S3 on Outposts must have permission to replicate objects from the source bucket to the destination bucket on your behalf. That means you must create a service role to delegate `GET` and `PUT` permissions to S3 on Outposts.
  1. Before creating the service role, you must have `GET` permission on the source bucket and `PUT` permission on the destination bucket.
  2. To create the service role to delegate permissions to S3 on Outposts, you must first configure permissions to allow an IAM entity (a user or role) to perform the `iam:CreateRole` and `iam:PassRole` actions. Then, you allow the IAM entity to create a service role. To make S3 on Outposts assume the service role on your behalf and delegate `GET` and `PUT` permissions to S3 on Outposts, you must assign the necessary trust and permissions policies to the role. For more information about these permissions for S3 on Outposts, see Creating an IAM role. For more information about creating a service role, see Creating a service role.

What is replicated?

By default, S3 on Outposts replicates the following:

- Objects created after you add a replication configuration.
- Object metadata from the source objects to the replicas. For information about how to replicate metadata from the replicas to the source objects, see Replication status if Amazon S3 replica modification sync on Outposts is enabled.
- Object tags, if there are any.

How delete operations affect replication

If you delete an object from the source bucket, the following actions occur by default:

- If you make a DELETE request without specifying an object version ID, S3 on Outposts adds a delete marker. S3 on Outposts deals with the delete marker as follows:
  - S3 on Outposts does not replicate the delete marker by default.
  - However, you can add *delete marker replication* to non-tag-based rules. For more information about how to enable delete marker replication in your replication configuration, see Using the S3 console.
- If you specify an object version ID to delete in a DELETE request, S3 on Outposts permanently deletes that object version in the source bucket. However, it doesn't replicate the deletion in the destination buckets. In other words, it doesn't delete the same object version from the destination buckets. This behavior protects data from malicious deletions.

What isn't replicated?

By default, S3 on Outposts doesn't replicate the following:

- Objects in the source bucket that are replicas that were created by another replication rule. For example, suppose you configure replication where bucket A is the source and bucket B is the destination. Now suppose that you add another replication configuration where bucket B is the source and bucket C is the destination. In this case, objects in bucket B that are replicas of objects in bucket A are not replicated to bucket C.

- Objects in the source bucket that have already been replicated to a different destination. For example, if you change the destination bucket in an existing replication configuration, S3 on Outposts won't replicate the objects again.
- Objects that are created with server-side encryption with customer-provided encryption keys (SSE-C).
- Updates to bucket-level subresources.

  For example, if you change the lifecycle configuration or add a notification configuration to your source bucket, these changes are not applied to the destination bucket. This feature makes it possible to have different configurations on the source and destination buckets.
- Actions performed by lifecycle configuration.

  For example, if you enable a lifecycle configuration only on your source bucket and configure expiration actions, S3 on Outposts creates delete markers for expired objects in the source bucket but doesn't replicate those markers to the destination buckets. If you want the same lifecycle configuration applied to both the source and destination buckets, enable the same lifecycle configuration on both. For more information about lifecycle configuration, see Creating and managing a lifecycle configuration for your Amazon S3 on Outposts bucket.

What isn't supported by S3 Replication on Outposts?

The following S3 Replication features are currently not supported by S3 on Outposts:

- S3 Replication Time Control (S3 RTC). S3 RTC is not supported because the object traffic in S3 Replication on Outposts travels over your on-premises network (the local gateway). For more information about local gateways, see Working with the local gateway in the *AWS Outposts User Guide*.
- S3 Replication for Batch Operations.

Setting up replication

> **Note**
>
> Objects that existed in your bucket before you set up a replication rule aren't replicated automatically. In other words, Amazon S3 on Outposts doesn't replicate objects retroactively. To replicate objects that were created before your replication configuration,

you can use the `CopyObject` API operation to copy them to the same bucket. After the objects are copied, they appear as "new" objects in the bucket and the replication configuration will apply to them. For more information about copying an object, see Copying an object in an Amazon S3 on Outposts bucket using the AWS SDK for Java and CopyObject in the *Amazon Simple Storage Service API Reference*.

To enable S3 Replication on Outposts, add a replication rule to your source Outposts bucket. The replication rule tells S3 on Outposts to replicate objects as specified. In the replication rule, you must provide the following:

- **The source Outposts bucket access point** – The access point Amazon Resource Name (ARN) or access point alias of the bucket from which you want S3 on Outposts to replicate the objects. For more information about using access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.
- **The objects that you want to replicate** – You can replicate all of the objects in the source Outposts bucket or a subset. You identify a subset by providing a key name prefix, one or more object tags, or both in the configuration.

  For example, if you configure a replication rule to replicate only objects with the key name prefix `Tax/`, S3 on Outposts replicates objects with keys such as `Tax/doc1` or `Tax/doc2`. But it doesn't replicate objects with the key `Legal/doc3`. If you specify both a prefix and one or more tags, S3 on Outposts replicates only objects that have the specific key prefix and tags.
- **The destination Outposts bucket** – The ARN or access point alias of the bucket to which you want S3 on Outposts to replicate the objects.

You can configure the replication rule by using the REST API, AWS SDKs, AWS Command Line Interface (AWS CLI), or Amazon S3 console.

S3 on Outposts also provides API operations to support setting up replication rules. For more information, see the following topics in the *Amazon Simple Storage Service API Reference*:

- PutBucketReplication
- GetBucketReplication
- DeleteBucketReplication

Topics

- Prerequisites for creating replication rules
- Creating replication rules on Outposts

Prerequisites for creating replication rules

Topics

- Connecting your source and destination Outpost subnets
- Creating an IAM role

Connecting your source and destination Outpost subnets

To have your replication traffic go from your source Outpost to your destination Outpost over your local gateway, you must add a new route to set up networking. You must connect the Classless Inter-Domain Routing (CIDR) networking ranges of your access points together. For each pair of access points, you need to set up this connection only once.

Some steps to set up the connection are different, depending on the access type of your Outposts endpoints that are associated with your access points. The access type for endpoints is either **Private** (direct virtual private cloud [VPC] routing for AWS Outposts) or **Customer owned IP** (a customer-owned IP address pool [CoIP pool] within your on-premises network).

Step 1: Find the CIDR range of your source Outposts endpoint

To find the CIDR range of your source endpoint that's associated with your source access point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. In the **Outposts buckets** list, choose the source bucket that you want for replication.
4. Choose the **Outposts access points** tab, and choose the Outposts access point for the source bucket for your replication rule.
5. Choose the Outposts endpoint.
6. Copy the subnet ID for use in Step 5.
7. The method that you use to find the CIDR range of the source Outposts endpoint depends on the access type of your endpoint.

In the **Outposts endpoint overview** section, see the **Access Type**.

- If the access type is **Private**, copy the **Classless inter-domain routing (CIDR)** value to use in Step 6.
- If the access type is **Customer Owned IP**, do the following:
  1. Copy the **Customer owned IPv4 pool** value to use as the ID of the address pool later on.
  2. Open the AWS Outposts console at https://console.aws.amazon.com/outposts/.
  3. In the navigation pane, choose **Local gateway route tables**.
  4. Choose the **Local gateway route table ID** value of your source Outpost.
  5. In the details pane, choose the **CoIP pools** tab. Paste the value of your CoIP pool ID that you copied previously in the search box.
  6. For the matched CoIP pool, copy the corresponding **CIDRs** value of your source Outposts endpoint for use in Step 6.

Step 2: Find the subnet ID and the CIDR range of your destination Outposts endpoint

To find the subnet ID and the CIDR range of your destination endpoint that's associated with your destination access point, follow the same substeps in Step 1 and change your source Outposts endpoint to your destination Outposts endpoint when you apply those substeps. Copy the subnet ID value of your destination Outposts endpoint for use in Step 6. Copy the CIDR value of your destination Outposts endpoint for use in Step 5.

Step 3: Find the local gateway ID of your source Outpost

To find the local gateway ID of your source Outpost

1. Open the AWS Outposts console at https://console.aws.amazon.com/outposts/.
2. In the left navigation pane, choose **Local gateways**.
3. On the **Local gateways** page, find the Outpost ID of your source Outpost that you want to use for replication.
4. Copy the local gateway ID value of your source Outpost for use in Step 5.

For more information about local gateway, see Local gateway in the *AWS Outposts User Guide*.

Step 4: Find the local gateway ID of your destination Outpost

To find the local gateway ID of your destination Outpost, follow the same substeps in Step 3, except look for the Outpost ID for your destination Outpost. Copy the local gateway ID value of your destination Outpost for use in Step 6.

Step 5: Set up the connection from your source Outpost subnet to your destination Outpost subnet

To connect from your source Outpost subnet to your destination Outpost subnet

1. Sign in to the AWS Management Console and open the VPC console at https://console.aws.amazon.com/vpc/.
2. In the left navigation pane, choose **Subnets**.
3. In the search box, enter the subnet ID for your source Outposts endpoint that you found in Step 1. Choose the subnet with the matched subnet ID.
4. For the matched subnet item, choose the **Route table** value of this subnet.
5. On the page with a selected route table, choose **Actions**, and then choose **Edit routes**.
6. On the **Edit routes** page, choose **Add route**.
7. Under **Destination**, enter the CIDR range of your destination Outposts endpoint that you found in Step 2.
8. Under **Target**, choose **Outpost Local Gateway**, and enter the local gateway ID of your source Outpost that you found in Step 3.
9. Choose **Save changes**.
10. Make sure the **Status** for the route is **Active**.

Step 6: Set up the connection from your destination Outpost subnet to your source Outpost subnet

1. Sign in to the AWS Management Console and open the VPC console at https://console.aws.amazon.com/vpc/.
2. In the left navigation pane, choose **Subnets**.
3. In the search box, enter the subnet ID for your destination Outposts endpoint that you found in Step 2. Choose the subnet with the matched subnet ID.
4. For the matched subnet item, choose the **Route table** value of this subnet.

5. On the page with a selected route table, choose **Actions**, and then choose **Edit routes**.
6. On the **Edit routes** page, choose **Add route**.
7. Under **Destination**, enter the CIDR range of your source Outposts endpoint that you found in Step 1.
8. Under **Target**, choose **Outpost Local Gateway**, and enter the local gateway ID of your destination Outpost that you found in Step 4.
9. Choose **Save changes**.
10. Make sure the **Status** for the route is **Active**.

After you connect the CIDR networking ranges of your source and destination access points, you must create an AWS Identity and Access Management (IAM) role.

Creating an IAM role

By default, all S3 on Outposts resources—buckets, objects, and related subresources—are private, and only the resource owner can access the resource. S3 on Outposts needs permissions to read and replicate objects from the source Outposts bucket. You grant these permissions by creating an IAM *service role* and specifying that role in your replication configuration.

This section explains the trust policy and minimum required permissions policy. The example walkthroughs provide step-by-step instructions to create an IAM role. For more information, see Creating replication rules on Outposts. For more information about IAM roles, see IAM roles in the *IAM User Guide*.

- The following example shows a *trust policy*, where you identify S3 on Outposts as the service principal that can assume the role.

  JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3-outposts.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
```

```
    ]
}
```

- The following example shows an *access policy*, where you grant the role permissions to perform replication tasks on your behalf. When S3 on Outposts assumes the role, it has the permissions that you specify in this policy. To use this policy, replace the *`user input placeholders`* with your own information. Make sure to replace them with the Outpost IDs of your source and destination Outposts and the bucket names and access point names of your source and destination Outposts buckets.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3-outposts:GetObjectVersionForReplication",
                "s3-outposts:GetObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3-outposts:us-east-1:123456789012:outpost/SOURCE-
OUTPOST-ID/bucket/SOURCE-OUTPOSTS-BUCKET/object/*",
                "arn:aws:s3-outposts:us-east-1:123456789012:outpost/SOURCE-
OUTPOST-ID/accesspoint/SOURCE-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3-outposts:ReplicateObject",
                "s3-outposts:ReplicateDelete"
            ],
            "Resource": [
                "arn:aws:s3-outposts:us-
east-1:123456789012:outpost/DESTINATION-OUTPOST-ID/bucket/DESTINATION-OUTPOSTS-
BUCKET/object/*",
                "arn:aws:s3-outposts:us-
east-1:123456789012:outpost/DESTINATION-OUTPOST-ID/accesspoint/DESTINATION-
OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
            ]
        }
```

```
    ]
}
```

The access policy grants permissions for the following actions:

- `s3-outposts:GetObjectVersionForReplication` – Permission for this action is granted on all objects to allow S3 on Outposts to get a specific object version that's associated with each object.

- `s3-outposts:GetObjectVersionTagging` – Permission for this action on objects in the *`SOURCE-OUTPOSTS-BUCKET`* bucket (the source bucket) allows S3 on Outposts to read object tags for replication. For more information, see Adding tags for S3 on Outposts buckets. If S3 on Outposts doesn't have this permission, it replicates the objects, but not the object tags.
- `s3-outposts:ReplicateObject` and `s3-outposts:ReplicateDelete` – Permissions for these actions on all objects in the *`DESTINATION-OUTPOSTS-BUCKET`* bucket (the destination bucket) allow S3 on Outposts to replicate objects or delete markers to the destination Outposts bucket. For information about delete markers, see How delete operations affect replication.

> **Note**
>
> - Permission for the `s3-outposts:ReplicateObject` action on the *`DESTINATION-OUTPOSTS-BUCKET`* bucket (the destination bucket) also allows replication of object tags. Therefore, you don't need to explicitly grant permission for the `s3-outposts:ReplicateTags` action.
> - For cross-account replication, the owner of the destination Outposts bucket must update its bucket policy to grant permission for the `s3-outposts:ReplicateObject` action on the*`DESTINATION-OUTPOSTS-BUCKET`*. The `s3-outposts:ReplicateObject` action allows S3 on Outposts to replicate objects and object tags to the destination Outposts bucket.

For a list of S3 on Outposts actions, see Actions defined by S3 on Outposts.

> **Important**
>
> The AWS account that owns the IAM role must have permissions for the actions that it grants to the IAM role.

For example, suppose that the source Outposts bucket contains objects owned by another AWS account. The owner of the objects must explicitly grant the AWS account that owns the IAM role the required permissions through the bucket policy and the access point policy. Otherwise, S3 on Outposts can't access the objects, and replication of the objects fails.
The permissions described here are related to the minimum replication configuration. If you choose to add optional replication configurations, you must grant additional permissions to S3 on Outposts.

Granting permissions when the source and destination Outposts buckets are owned by different AWS accounts

When the source and destination Outposts buckets aren't owned by the same accounts, the owner of the destination Outposts bucket must update the bucket and access point policies for the destination bucket. These policies must grant the owner of the source Outposts bucket and the IAM service role permissions to perform replication actions, as shown in the following policy examples, or replication will fail. In these policy examples, *DESTINATION-OUTPOSTS-BUCKET* is the destination bucket. To use these policy examples, replace the *user input placeholders* with your own information.

If you're creating the IAM service role manually, set the role path as `role/service-role/`, as shown in the following policy examples. For more information, see IAM ARNs in the *IAM User Guide*.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "PolicyForDestinationBucket",
    "Statement": [
        {
            "Sid": "Permissions on objects",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/service-role/source-
account-IAM-role"
            },
            "Action": [
                "s3-outposts:ReplicateDelete",
```

```
                "s3-outposts:ReplicateObject"
            ],
            "Resource": [
                "arn:aws:s3-outposts:us-east-1:444455556666:outpost/DESTINATION-
OUTPOST-ID/bucket/DESTINATION-OUTPOSTS-BUCKET/object/*"
            ]
        }
    ]
}
```

JSON

```
{
    "Version":"2012-10-17",
    "Id": "PolicyForDestinationAccessPoint",
    "Statement": [
        {
            "Sid": "Permissions on objects",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/service-role/source-
account-IAM-role"
            },
            "Action": [
                "s3-outposts:ReplicateDelete",
                "s3-outposts:ReplicateObject"
            ],
            "Resource": [
                "arn:aws:s3-outposts:us-east-1:111122223333:outpost/DESTINATION-
OUTPOST-ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
            ]
        }
    ]
}
```

Note

If objects in the source Outposts bucket are tagged, note the following:
If the source Outposts bucket owner grants S3 on Outposts permission for the `s3-outposts:GetObjectVersionTagging` and `s3-outposts:ReplicateTags` actions to

replicate object tags (through the IAM role), Amazon S3 replicates the tags along with the objects. For information about the IAM role, see Creating an IAM role.

Creating replication rules on Outposts

S3 Replication on Outposts is the automatic, asynchronous replication of objects across buckets in the same or different AWS Outposts. Replication copies newly created objects and object updates from a source Outposts bucket to a destination Outposts bucket or buckets. For more information, see Replicating objects for S3 on Outposts.

Note

Objects that existed in the source Outposts bucket before you set up replication rules aren't replicated. In other words, S3 on Outposts doesn't replicate objects retroactively. To replicate objects that were created before your replication configuration, you can use the `CopyObject` API operation to copy them to the same bucket. After the objects are copied, they appear as "new" objects in the bucket and the replication configuration will apply to them. For more information about copying an object, see Copying an object in an Amazon S3 on Outposts bucket using the AWS SDK for Java and CopyObject in the *Amazon Simple Storage Service API Reference*.

When you configure replication, you add replication rules to the source Outposts bucket. Replication rules define which source Outposts bucket objects to replicate and the destination Outposts bucket or buckets where the replicated objects will be stored. You can create a rule to replicate all the objects in a bucket or a subset of objects with a specific key name prefix, one or more object tags, or both. A destination Outposts bucket can be in the same Outpost as the source Outposts bucket, or it can be in a different Outpost.

For S3 on Outposts replication rules, you must provide both the source Outposts bucket's access point Amazon Resource Name (ARN) and the destination Outposts bucket's access point ARN instead of the source and destination Outposts bucket names.

If you specify an object version ID to delete, S3 on Outposts deletes that object version in the source Outposts bucket. But it doesn't replicate the deletion to the destination Outposts bucket. In other words, it doesn't delete the same object version from the destination Outposts bucket. This behavior protects data from malicious deletions.

When you add a replication rule to an Outposts bucket, the rule is enabled by default, so the rule starts working as soon as you save it.

In this example, you set up replication for source and destination Outposts buckets that are on different Outposts and are owned by the same AWS account. Examples are provided for using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), and the AWS SDK for Java and AWS SDK for .NET. For information about cross-account S3 Replication on Outposts permissions, see Granting permissions when the source and destination Outposts buckets are owned by different AWS accounts.

For prerequisites to set up S3 on Outposts replication rules, see Prerequisites for creating replication rules.

Using the S3 console

Follow these steps to configure a replication rule when the destination Amazon S3 on Outposts bucket is in a different Outpost from the source Outposts bucket.

If the destination Outposts bucket is in a different account from the source Outposts bucket, you must add a bucket policy to the destination Outposts bucket to grant the owner of the source Outposts bucket account permission to replicate objects in the destination Outposts bucket.

To create a replication rule

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Outposts Buckets** list, choose the name of the bucket that you want to use as the source bucket.
3. Choose the **Management** tab, scroll down to the **Replication rules** section, and then choose **Create replication rule**.
4. For **Replication rule name**, enter a name for your rule to help identify the rule later. The name is required and must be unique within the bucket.
5. Under **Status**, **Enabled** is chosen by default. An enabled rule starts to work as soon as you save it. If you want to enable the rule later, choose **Disabled**.
6. Under **Priority**, the rule's priority value determines which rule to apply if there are overlapping rules. When objects are included in the scope of more than one replication rule, S3 on Outposts uses these priority values to avoid conflicts. By default, new rules are added to the replication configuration at the highest priority. The higher the number, the higher the priority.

To change the priority for the rule, after you save the rule, choose the rule name from the replication rule list, choose **Actions**, and then choose **Edit priority**.

7. Under **Source bucket**, you have the following options for setting the replication source:

   - To replicate the whole bucket, choose **Apply to *all* objects in the bucket**.
   - To apply prefix or tag filtering to the replication source, choose **Limit the scope of this rule by using one or more filters**. You can combine a prefix and tags.
     - To replicate all objects that have the same prefix, under **Prefix**, enter a prefix in the box. Using the **Prefix** filter limits replication to all objects that have names that begin with the same string (for example, `pictures`).

       If you enter a prefix that is the name of a folder, you must use a **/** (forward slash) as the last character (for example, `pictures/`).
     - To replicate all objects that have one or more of the same object tags, choose **Add tag**, and then enter the key-value pair in the boxes. To add another tag, repeat the procedure. For more information about object tags, see Adding tags for S3 on Outposts buckets.
8. To access your S3 on Outposts source bucket for replication, under **Source access point name**, choose an access point that is attached to the source bucket.
9. Under **Destination**, choose the access point ARN of the destination Outposts bucket where you want S3 on Outposts to replicate objects. The destination Outposts bucket can be in the same or a different AWS account as the source Outposts bucket.

   If the destination bucket is in a different account from the source Outposts bucket, you must add a bucket policy to the destination Outposts bucket to grant the owner of the source Outposts bucket account permission to replicate objects to the destination Outposts bucket. For more information, see Granting permissions when the source and destination Outposts buckets are owned by different AWS accounts.

   > **Note**
   >
   > If versioning is not enabled on the destination Outposts bucket, you get a warning that contains an **Enable versioning** button. Choose this button to enable versioning on the bucket.

10. Set up an AWS Identity and Access Management (IAM) service role that S3 on Outposts can assume to replicate objects on your behalf.

To set up an IAM role, under **IAM role**, do one of the following:

- To have S3 on Outposts create a new IAM role for your replication configuration, choose **Choose from existing IAM roles**, and then choose **Create new role**. When you save the rule, a new policy is generated for the IAM role that matches the source and destination Outposts buckets that you choose. We recommend that you choose **Create new role**.
- You can also choose to use an existing IAM role. If you do, you must choose a role that grants S3 on Outposts the necessary permissions for replication. If this role doesn't grant S3 on Outposts sufficient permissions to follow your replication rule, replication fails.

  To choose an existing role, choose **Choose from existing IAM roles**, and then choose the role from the dropdown menu. You can also choose **Enter an IAM role ARN** and then enter the IAM role's Amazon Resource Name (ARN).

> **Important**
>
> When you add a replication rule to an S3 on Outposts bucket, you must have the `iam:CreateRole` and `iam:PassRole` permissions to be able to create and pass the IAM role that grants S3 on Outposts replication permissions. For more information, see Granting a user permissions to pass a role to an AWS service in the *IAM User Guide*.

11. All objects in Outposts buckets are encrypted by default. For more information about S3 on Outposts encryption, see Data encryption in S3 on Outposts. Only objects that are encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3) can be replicated. The replication of objects that are encrypted by using server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) or server-side encryption with customer-provided encryption keys (SSE-C) is not supported.
12. As needed, enable the following additional options while setting the replication rule configuration:

    - If you want to enable S3 on Outposts replication metrics in your replication configuration, select **Replication metrics**. For more information, see Monitoring progress with replication metrics.
    - If you want to enable delete marker replication in your replication configuration, select **Delete marker replication**. For more information, see How delete operations affect replication.

- If you want to replicate metadata changes made to the replicas back to the source objects, select **Replica modification sync**. For more information, see Replication status if Amazon S3 replica modification sync on Outposts is enabled.

13. To finish, choose **Create rule**.

After you save your rule, you can edit, enable, disable, or delete your rule. To do so, go to the **Management** tab for the source Outposts bucket, scroll down to the **Replication rules** section, choose your rule, and then choose **Edit rule**.

Using the AWS CLI

To use the AWS CLI to set up replication when the source and destination Outposts buckets are owned by the same AWS account, you do the following:

- Create source and destination Outposts buckets.
- Enable versioning on both of the buckets.
- Create an IAM role that gives S3 on Outposts permission to replicate objects.
- Add the replication configuration to the source Outposts bucket.

To verify your setup, you test it.

To set up replication when the source and destination Outposts buckets are owned by the same AWS account

1. Set a credentials profile for the AWS CLI. In this example, we use the profile name `acctA`. For information about setting credential profiles, see Named profiles in the *AWS Command Line Interface User Guide*.

> **Important**
>
> The profile that you use for this exercise must have the necessary permissions. For example, in the replication configuration, you specify the IAM service role that S3 on Outposts can assume. You can do this only if the profile that you use has the `iam:CreateRole` and `iam:PassRole` permissions. For more information, see Granting a user permissions to pass a role to an AWS service in the *IAM User Guide*.

If you use administrator credentials to create a named profile, the named profile will have the necessary permission to perform all the tasks.

2. Create a source bucket and enable versioning on it. The following `create-bucket` command creates a *SOURCE-OUTPOSTS-BUCKET* bucket in the US East (N. Virginia) (`us-east-1`) Region. To use this command, replace the *user input placeholders* with your own information.

```
aws s3control create-bucket --bucket SOURCE-OUTPOSTS-BUCKET --outpost-id SOURCE-OUTPOST-ID --profile acctA --region us-east-1
```

The following `put-bucket-versioning` command enables versioning on the *SOURCE-OUTPOSTS-BUCKET* bucket. To use this command, replace the *user input placeholders* with your own information.

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/bucket/SOURCE-OUTPOSTS-BUCKET --versioning-configuration Status=Enabled --profile acctA
```

3. Create a destination bucket and enable versioning on it. The following `create-bucket` command creates a *DESTINATION-OUTPOSTS-BUCKET* bucket in the US West (Oregon) (`us-west-2`) Region. To use this command, replace the *user input placeholders* with your own information.

Note

To set up a replication configuration when both the source and destination Outposts buckets are in the same AWS account, you use the same named profile. This example uses `acctA`. To test the replication configuration when the buckets are owned by different AWS accounts, you specify different profiles for each bucket.

```
aws s3control create-bucket --bucket DESTINATION-OUTPOSTS-BUCKET --create-bucket-configuration LocationConstraint=us-west-2 --outpost-id DESTINATION-OUTPOST-ID --profile acctA --region us-west-2
```

The following `put-bucket-versioning` command enables versioning on the *DESTINATION-OUTPOSTS-BUCKET* bucket. To use this command, replace the *user input placeholders* with your own information.

```
aws s3control put-bucket-versioning --account-id 123456789012 --bucket arn:aws:s3-
outposts:region:123456789012:outpost/DESTINATION-OUTPOST-ID/bucket/DESTINATION-
OUTPOSTS-BUCKET --versioning-configuration Status=Enabled --profile acctA
```

4. Create an IAM service role. Later in the replication configuration, you add this service role to the *SOURCE-OUTPOSTS-BUCKET* bucket. S3 on Outposts assumes this role to replicate objects on your behalf. You create an IAM role in two steps:

   a. Create an IAM role.

      i. Copy the following trust policy and save it to a file named `s3-on-outposts-role-trust-policy.json` in the current directory on your local computer. This policy grants S3 on Outposts service principal permissions to assume the service role.

      JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3-outposts.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

      ii. Run the following command to create the role. Replace the *user input placeholders* with your own information.

```
aws iam create-role --role-name replicationRole --assume-role-policy-
document file://s3-on-outposts-role-trust-policy.json --profile acctA
```

   b. Attach a permissions policy to the service role.

      i. Copy the following permissions policy and save it to a file named `s3-on-outposts-role-permissions-policy.json` in the current directory on your local computer. This policy grants permissions for various S3 on Outposts bucket and object

actions. To use this policy, replace the *user input placeholders* with your own information.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3-outposts:GetObjectVersionForReplication",
                "s3-outposts:GetObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3-outposts:us-
east-1:123456789012:outpost/SOURCE-OUTPOST-ID/bucket/SOURCE-OUTPOSTS-
BUCKET/object/*",
                "arn:aws:s3-outposts:us-
east-1:123456789012:outpost/SOURCE-OUTPOST-ID/accesspoint/SOURCE-
OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3-outposts:ReplicateObject",
                "s3-outposts:ReplicateDelete"
            ],
            "Resource": [
                "arn:aws:s3-outposts:us-
east-1:123456789012:outpost/DESTINATION-OUTPOST-ID/
bucket/DESTINATION-OUTPOSTS-BUCKET/object/*",
                "arn:aws:s3-outposts:us-
east-1:123456789012:outpost/DESTINATION-OUTPOST-ID/
accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT/object/*"
            ]
        }
    ]
}
```

ii. Run the following command to create a policy and attach it to the role. Replace the *user input placeholders* with your own information.

```
aws iam put-role-policy --role-name replicationRole --policy-
document file://s3-on-outposts-role-permissions-policy.json --policy-
name replicationRolePolicy --profile acctA
```

5. Add a replication configuration to the *SOURCE-OUTPOSTS-BUCKET* bucket.

a. Although the S3 on Outposts API requires a replication configuration in XML format, the AWS CLI requires that you specify the replication configuration in JSON format. Save the following JSON in a file called `replication.json` to the local directory on your computer. To use this configuration, replace the *user input placeholders* with your own information.

```
{
  "Role": "IAM-role-ARN",
  "Rules": [
    {
      "Status": "Enabled",
      "Priority": 1,
      "DeleteMarkerReplication": { "Status": "Disabled" },
      "Filter" : { "Prefix": "Tax"},
      "Destination": {
        "Bucket":
        "arn:aws:s3-outposts:region:123456789012:outpost/DESTINATION-OUTPOST-
ID/accesspoint/DESTINATION-OUTPOSTS-BUCKET-ACCESS-POINT"
      }
    }
  ]
}
```

b. Run the following `put-bucket-replication` command to add the replication configuration to your source Outposts bucket. To use this command, replace the *user input placeholders* with your own information.

```
aws s3control put-bucket-replication --account-id 123456789012 --
bucket arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-
ID/bucket/SOURCE-OUTPOSTS-BUCKET --replication-configuration file://
replication.json --profile acctA
```

c. To retrieve the replication configuration, use the `get-bucket-replication` command. To use this command, replace the *`user input placeholders`* with your own information.

```
aws s3control get-bucket-replication --account-id 123456789012 --bucket
 arn:aws:s3-outposts:region:123456789012:outpost/SOURCE-OUTPOST-ID/
bucket/SOURCE-OUTPOSTS-BUCKET --profile acctA
```

6. Test the setup in the Amazon S3 console:

   a. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   b. In the *`SOURCE-OUTPOSTS-BUCKET`* bucket, create a folder named `Tax`.

   c. Add sample objects to the `Tax` folder in the *`SOURCE-OUTPOSTS-BUCKET`* bucket.

   d. In the *`DESTINATION-OUTPOSTS-BUCKET`* bucket, verify the following:

   - S3 on Outposts replicated the objects.

     Note

     The amount of time that it takes for S3 on Outposts to replicate an object depends on the size of the object. For information about how to see the status of replication, see Getting replication status information.

   - On the object's **Properties** tab, the **Replication status** is set to **Replica** (identifying this as a replica object).

Managing your replication

This section describes additional replication configuration options that are available in S3 on Outposts, how to determine the replication status, and how to troubleshot replication. For information about core replication configuration, see Setting up replication.

Topics

- Monitoring progress with replication metrics
- Getting replication status information
- Troubleshooting replication

- Using EventBridge for S3 Replication on Outposts

Monitoring progress with replication metrics

S3 Replication on Outposts provides detailed metrics for the replication rules in your replication configuration. With replication metrics, you can monitor in 5-minute intervals the progress of replication by tracking bytes pending replication, replication latency replication, and operations pending. To assist in troubleshooting any configuration issues, you can also set up Amazon EventBridge to receive notifications about replication failures.

When replication metrics are enabled, S3 Replication on Outposts publishes the following metrics to Amazon CloudWatch:

- **Bytes Pending Replication** – The total number of bytes of objects that are pending replication for a given replication rule.
- **Replication Latency** – The maximum number of seconds by which the replication destination bucket is behind the source bucket for a given replication rule.
- **Operations Pending Replication** – The number of operations that are pending replication for a given replication rule. Operations include objects, delete markers, and tags.

> **Note**
>
> S3 Replication on Outposts metrics are billed at the same rate as CloudWatch custom metrics. For more information, see CloudWatch pricing.

Getting replication status information

The replication status can help you determine the current state of an object that's being replicated by Amazon S3 on Outposts. The replication status of a source object will return either `PENDING`, `COMPLETED`, or `FAILED`. The replication status of a replica will return `REPLICA`.

Replication status overview

In a replication scenario, you have a source bucket on which you configure replication and a destination bucket to which S3 on Outposts replicates objects. When you request an object (using `GetObject`) or object metadata (using `HeadObject`) from these buckets, S3 on Outposts returns the `x-amz-replication-status` header in the response as follows:

- When you request an object from the source bucket, S3 on Outposts returns the `x-amz-replication-status` header if the object in your request is eligible for replication.

  For example, suppose that you specify the object prefix `TaxDocs` in your replication configuration to tell S3 on Outposts to replicate only objects with the key name prefix `TaxDocs`. Any objects that you upload that have this key name prefix—for example, `TaxDocs/document1.pdf`—will be replicated. For object requests with this key name prefix, S3 on Outposts returns the `x-amz-replication-status` header with one of the following values for the object's replication status: `PENDING`, `COMPLETED`, or `FAILED`.

  > **Note**
  >
  > If object replication fails after you upload an object, you can't retry replication. You must upload the object again. Objects transition to a `FAILED` state for issues such as missing replication role permissions or missing bucket permissions. For temporary failures, such as if a bucket or your Outpost is unavailable, the replication status doesn't transition to `FAILED`, but remains `PENDING`. After the resource is back online, S3 on Outposts resumes replicating those objects.

- When you request an object from a destination bucket, if the object in your request is a replica that S3 on Outposts created, S3 on Outposts returns the `x-amz-replication-status` header with the value `REPLICA`.

> **Note**
>
> Before deleting an object from a source bucket that has replication enabled, check the object's replication status to ensure that the object has been replicated.

Replication status if Amazon S3 replica modification sync on Outposts is enabled

When your replication rules enable S3 on Outposts replica modification sync, replicas can report statuses other than `REPLICA`. If metadata changes are in the process of replicating, the `x-amz-replication-status` header for the replica returns `PENDING`. If replica modification sync fails to replicate metadata, the header for the replica returns `FAILED`. If metadata is replicated correctly, the header for the replica returns the value `REPLICA`.

Troubleshooting replication

If object replicas don't appear in the destination Amazon S3 on Outposts bucket after you configure replication, use these troubleshooting tips to identify and fix issues.

- The time it takes S3 on Outposts to replicate an object depends on several factors, including the distance between the source and destination Outposts, and the size of the object.

  You can check the source object's replication status. If the object's replication status is `PENDING`, S3 on Outposts hasn't completed the replication. If the object's replication status is `FAILED`, check the replication configuration that you set on the source bucket.
- In the replication configuration on the source bucket, verify the following:
  - The access point Amazon Resource Name (ARN) of the destination bucket is correct.
  - The key name prefix is correct. For example, if you set the configuration to replicate objects with the prefix `Tax`, then only objects with key names such as `Tax/document1` or `Tax/document2` are replicated. An object with the key name `document3` is not replicated.
  - The status is `Enabled`.
- Verify that versioning hasn't been suspended on either bucket. Both the source and destination buckets must have versioning enabled.
- If the destination bucket is owned by another AWS account, verify that the bucket owner has a bucket policy on the destination bucket that allows the source bucket owner to replicate objects. For an example, see Granting permissions when the source and destination Outposts buckets are owned by different AWS accounts.
- If an object replica doesn't appear in the destination bucket, the following issues might prevent replication:
  - S3 on Outposts doesn't replicate an object in a source bucket that is a replica created by another replication configuration. For example, if you set a replication configuration from bucket A to bucket B to bucket C, S3 on Outposts doesn't replicate object replicas in bucket B to bucket C.

    If you want to replicate objects in bucket A to bucket B and bucket C, set multiple bucket destinations in different replication rules for your source bucket replication configuration. For example, create two replication rules on source bucket A, with one rule to replicate to destination bucket B and the other rule to replicate to destination bucket C.
  - A source bucket owner can grant other AWS accounts permission to upload objects. By default, the source bucket owner doesn't have permissions for the objects created by other accounts.

The replication configuration replicates only the objects for which the source bucket owner has access permissions. To avoid replication issues, the source bucket owner can grant other AWS accounts permissions to create objects conditionally, requiring explicit access permissions on those objects.

- Suppose that in the replication configuration, you add a rule to replicate a subset of objects that have a specific tag. In this case, you must assign the specific tag key and value at the time that the object is created in order for S3 on Outposts to replicate the object. If you first create an object and then add the tag to that existing object, S3 on Outposts doesn't replicate the object.
- Replication fails if the bucket policy denies access to the replication role for any of the following actions:

  Source bucket:

  ```
  "s3-outposts:GetObjectVersionForReplication",
  "s3-outposts:GetObjectVersionTagging"
  ```

  Destination buckets:

  ```
  "s3-outposts:ReplicateObject",
  "s3-outposts:ReplicateDelete",
  "s3-outposts:ReplicateTags"
  ```

- Amazon EventBridge can notify you when objects don't replicate to their destination Outposts. For more information, see Using EventBridge for S3 Replication on Outposts.

Using EventBridge for S3 Replication on Outposts

Amazon S3 on Outposts is integrated with Amazon EventBridge and uses the `s3-outposts` namespace. EventBridge is a serverless event bus service that you can use to connect your applications with data from a variety of sources. For more information, see What is Amazon EventBridge? in the *Amazon EventBridge User Guide*.

To assist in troubleshooting any replication configuration issues, you can set up Amazon EventBridge to receive notifications about replication failure events. EventBridge can notify you in instances when objects don't replicate to their destination Outposts. For more information about the current state of an object that's being replicated, see Replication status overview.

Whenever certain events happen in your Outposts bucket, S3 on Outposts can send events to EventBridge. Unlike other destinations, you don't need to select which event types that you want to deliver. You can also use EventBridge rules to route events to additional targets. After EventBridge is enabled, S3 on Outposts sends all of the following events to EventBridge.

Event type	Description	Namespace
`Operation FailedReplication`	The replication of an object within a replication rule failed. For more information about S3 Replication on Outposts failure reasons, see Using EventBridge to view S3 Replication on Outposts failure reasons.	`s3-outposts`

Using EventBridge to view S3 Replication on Outposts failure reasons

The following table lists S3 Replication on Outposts failure reasons. You can configure an EventBridge rule to publish and view the failure reason through Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), AWS Lambda, or Amazon CloudWatch Logs. For more information about the permissions that are required to use these resources for EventBridge, see Using resource-based policies for EventBridge.

Replication failure reason	Description
`AssumeRoleNotPermitted`	S3 on Outposts can't assume the AWS Identity and Access Management (IAM) role that's specified in the replication configuration.
`DstBucketNotFound`	S3 on Outposts can't find the destination bucket that's specified in the replication configuration.
`DstBucketUnversioned`	Versioning isn't enabled on the Outposts destination bucket. To replicate objects with S3 Replication on Outposts, you must enable versioning on the destination bucket.

Replication failure reason	Description
DstDelObjNotPermitted	S3 on Outposts can't replicate deletes to the destination bucket. The s3-outposts:ReplicateDelete permission might be missing for the destination bucket.
DstMultipartCompleteNotPermitted	S3 on Outposts can't complete a multipart upload of objects in the destination bucket. The s3-outposts:ReplicateObject permission might be missing for the destination bucket.
DstMultipartInitNotPermitted	S3 on Outposts can't initiate a multipart upload of objects to the destination bucket. The s3-outposts:ReplicateObject permission might be missing for the destination bucket.
DstMultipartPartUploadNotPermitted	S3 on Outposts can't upload multipart upload objects in the destination bucket. The s3-outposts:ReplicateObject permission might be missing for the destination bucket.
DstOutOfCapacity	S3 on Outposts can't replicate to the destination Outpost because the Outpost is out of S3 storage capacity.
DstPutObjNotPermitted	S3 on Outposts can't replicate objects to the destination bucket. The s3-outposts:ReplicateObject permission might be missing for the destination bucket.
DstPutTaggingNotPermitted	S3 on Outposts can't replicate object tags to the destination bucket. The s3-outposts:ReplicateObject permission might be missing for the destination bucket.

Replication failure reason	Description
`DstVersionNotFound`	S3 on Outposts can't find the required object version in the destination bucket in order to replicate that object version's metadata.
`SrcBucketReplicationConfigMissing`	S3 on Outposts can't find a replication configuration for the access point that's associated with the source Outposts bucket.
`SrcGetObjNotPermitted`	S3 on Outposts can't access the object in the source bucket for replication. The `s3-outposts:GetObjectVersionForReplication` permission might be missing for the source bucket.
`SrcGetTaggingNotPermitted`	S3 on Outposts can't access the object tag information from the source bucket. The `s3-outposts:GetObjectVersionTagging` permission might be missing for the source bucket.
`SrcHeadObjectNotPermitted`	S3 on Outposts can't retrieve object metadata from the source bucket. The `s3-outposts:GetObjectVersionForReplication` permission might be missing for the source bucket.
`SrcObjectNotEligible`	The object isn't eligible for replication. The object or its object tags don't match the replication configuration.

For more information about troubleshooting replication, see the following topics:

- Creating an IAM role
- Troubleshooting replication

Monitoring EventBridge with CloudWatch

For monitoring, Amazon EventBridge integrates with Amazon CloudWatch. EventBridge automatically sends metrics to CloudWatch every minute. These metrics include the number of events that have been matched by a rule and the number of times a target is invoked by a rule. When a rule runs in EventBridge, all of the targets associated with the rule are invoked. You can monitor your EventBridge behavior through CloudWatch in the following ways.

- You can monitor the available EventBridge metrics for your EventBridge rules from the CloudWatch dashboard. Then, you can use CloudWatch features, such as CloudWatch alarms, to set alarms on certain metrics. If those metrics reach the custom threshold values that you've specified in the alarms, you receive notifications and can take action accordingly.
- You can set Amazon CloudWatch Logs as a target of your EventBridge rule. Then, EventBridge creates log streams and CloudWatch Logs stores the text from the events as log entries. For more information, see EventBridge and CloudWatch Logs.

For more information about debugging EventBridge event delivery and archiving events, see the following topics:

- Event retry policy and using dead-letter queues
- Archiving EventBridge events

Sharing S3 on Outposts by using AWS RAM

Amazon S3 on Outposts supports sharing S3 capacity across multiple accounts within an organization by using AWS Resource Access Manager (AWS RAM). With S3 on Outposts sharing, you can allow others to create and manage buckets, endpoints, and access points on your Outpost.

This topic demonstrates how to use AWS RAM to share S3 on Outposts and related resources with another AWS account in your AWS organization.

Prerequisites

- The Outpost owner account has an organization configured in AWS Organizations. For more information, see Creating an organization in the *AWS Organizations User Guide*.

- The organization includes the AWS account that you want to share your S3 on Outposts capacity with. For more information, see Sending invitations to AWS accounts in the *AWS Organizations User Guide*.
- Select one of the following options that you want to share. The second resource (either **Subnets** or **Outposts**) must be selected so that endpoints are also accessible. Endpoints are a networking requirement in order to access data stored in S3 on Outposts.

Option 1	Option 2
S3 on Outposts Allows the user to create buckets on your Outposts and access points and to add objects to those buckets.	**S3 on Outposts** Allows the user to create buckets on your Outposts and access points and to add objects to those buckets.
Subnets Allows the user to use your virtual private cloud (VPC) and the endpoints that are associated with your subnet.	**Outposts** Allows the user to see S3 capacity charts and the AWS Outposts console home page. Also allows users to create subnets on shared Outposts and create endpoints.

Procedure

1. Sign in to the AWS Management Console by using the AWS account that owns the Outpost, and then open the AWS RAM console at https://console.aws.amazon.com/ram/home.
2. Make sure that you have enabled sharing with AWS Organizations in AWS RAM. For information, see Enable resource sharing within AWS Organizations in the *AWS RAM User Guide*.
3. Use either Option 1 or Option 2 in the prerequisites to create a resource share. If you have multiple S3 on Outposts resources, select the Amazon Resource Names (ARNs) of the resources that you want to share. To enable endpoints, share either your subnet or Outpost.

   For more information about how to create a resource share, see Create a resource share in the *AWS RAM User Guide*.

4. The AWS account that you shared your resources with should now be able to use S3 on Outposts. Depending on the option that you selected in the prerequisites, provide the following information to the account user:

Option 1	Option 2
The Outpost ID	The Outpost ID
The VPC ID	
The subnet ID	
The security group ID	

> **Note**
>
> The user can confirm that the resources have been shared with them by using the AWS RAM console, the AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. The user can view their existing resource shares by using the get-resource-shares CLI command.

Usage examples

After you have shared your S3 on Outposts resources with another account, that account can manage buckets and objects on your Outpost. If you shared the **Subnets** resource, then that account can use the endpoint that you created. The following examples demonstrate how a user can use the AWS CLI to interact with your Outpost after you share these resources.

Example: Create a bucket

The following example creates a bucket named *amzn-s3-demo-bucket1* on the Outpost *op-01ac5d28a6a232904*. Before using this command, replace each *user input placeholder* with the appropriate values for your use case.

```
aws s3control create-bucket --bucket amzn-s3-demo-bucket1 --outpost-
id op-01ac5d28a6a232904
```

For more information about this command, see create-bucket in the *AWS CLI Reference*.

Example: Create an access point

The following example creates an access point on an Outpost by using the example parameters in the following table. Before using this command, replace these *user input placeholder* values and the AWS Region code with the appropriate values for your use case.

Parameter	Value
Account ID	*111122223333*
Access point name	*example-outpost-access-point*
Outpost ID	*op-01ac5d28a6a232904*
Outpost bucket name	*amzn-s3-demo-bucket1*
VPC ID	*vpc-1a2b3c4d5e6f7g8h9*

> **Note**
>
> The Account ID parameter must be the AWS account ID of the bucket owner, which is the shared user.

```
aws s3control create-access-point --account-id 111122223333 --name example-outpost-
access-point \
--bucket arn:aws:s3-outposts:us-east-1:111122223333:outpost/op-01ac5d28a6a232904/
bucket/amzn-s3-demo-bucket1 \
--vpc-configuration VpcId=vpc-1a2b3c4d5e6f7g8h9
```

For more information about this command, see create-access-point in the *AWS CLI Reference*.

Example: Upload an object

The following example uploads the file *my_image.jpg* from the user's local file system to an object named *images/my_image.jpg* through the access point *example-outpost-access-point* on the Outpost *op-01ac5d28a6a232904*, owned by the AWS account *111122223333*. Before using this command, replace these *user input placeholder* values and the AWS Region code with the appropriate values for your use case.

```
aws s3api put-object --bucket arn:aws:s3-outposts:us-
east-1:111122223333:outpost/op-01ac5d28a6a232904/accesspoint/example-outpost-access-
point \
--body my_image.jpg --key images/my_image.jpg
```

For more information about this command, see put-object in the *AWS CLI Reference*.

> **Note**
>
> If this operation results in a Resource not found error or is unresponsive, your VPC might not have a shared endpoint.
> To check whether there is a shared endpoint, use the list-shared-endpoints AWS CLI command. If there is no shared endpoint, work with the Outpost owner to create one.
> For more information, see ListSharedEndpoints in the *Amazon Simple Storage Service API Reference*.

Example: Create an endpoint

The following example creates an endpoint on a shared Outpost. Before using this command, replace the *`user input placeholder`* values for the Outpost ID, subnet ID, and security group ID with the appropriate values for your use case.

> **Note**
>
> The user can perform this operation only if the resource share includes the **Outposts** resource.

```
aws s3outposts create-endpoint --outposts-id op-01ac5d28a6a232904 --subnet-id XXXXXX --
security-group-id XXXXXXX
```

For more information about this command, see create-endpoint in the *AWS CLI Reference*.

Other AWS services that use S3 on Outposts

Other AWS services that run local to your AWS Outposts can also use your Amazon S3 on Outposts capacity. In Amazon CloudWatch the `S3Outposts` namespace shows detailed metrics for buckets

within S3 on Outposts, but these metrics don't include usage for other AWS services. To manage your S3 on Outposts capacity that is consumed by other AWS services, see the information in the following table.

AWS service	Description	Learn more
Amazon S3	All direct S3 on Outposts usage has a matching account and bucket CloudWatch metric.	See metrics
Amazon Elastic Block Store (Amazon EBS)	For Amazon EBS on Outposts, you can choose an AWS Outpost as your snapshot destination and store locally in your S3 on Outpost.	Learn more
Amazon Relationa l Database Service (Amazon RDS)	You can use Amazon RDS local backups to store your RDS backups locally on your Outpost.	Learn more

Monitoring S3 on Outposts

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

For more information about monitoring your Amazon S3 on Outposts storage capacity, see the following topics.

Topics

- Managing S3 on Outposts capacity with Amazon CloudWatch metrics
- Receiving S3 on Outposts event notifications by using Amazon CloudWatch Events
- Monitoring S3 on Outposts with AWS CloudTrail logs

Managing S3 on Outposts capacity with Amazon CloudWatch metrics

To help manage the fixed S3 capacity on your Outpost, we recommend that you create CloudWatch alerts that tell you when your storage utilization exceeds a certain threshold. For more information about the CloudWatch metrics for S3 on Outposts, see CloudWatch metrics. If there is not enough space to store an object on your Outpost, the API returns an insufficient capacity exemption (ICE). To free up space, you can create CloudWatch alarms that trigger explicit data deletion, or use a lifecycle expiration policy to expire objects. To save data before deletion, you can use AWS DataSync to copy data from your Amazon S3 on Outposts bucket to an S3 bucket in an AWS Region. For more information about using DataSync, see Getting Started with AWS DataSync in the *AWS DataSync User Guide*.

CloudWatch metrics

The `S3Outposts` namespace includes the following metrics for Amazon S3 on Outposts buckets. You can monitor the total number of S3 on Outposts bytes provisioned, the total free bytes available for objects, and the total size of all objects for a given bucket. Bucket or account-related metrics exist for all direct S3 usage. Indirect S3 usage, such as storing Amazon Elastic Block Store local snapshots or Amazon Relational Database Service backups on an Outpost, consumes S3 capacity, but is not included in bucket or account-related metrics. For more information about Amazon EBS local snapshots, see Amazon EBS local snapshots on Outposts. To see your Amazon EBS cost report, visit https://console.aws.amazon.com/costmanagement/.

> **Note**
>
> S3 on Outposts supports only the following metrics, and no other Amazon S3 metrics. Because S3 on Outposts has a fixed capacity limit, we recommend creating CloudWatch alarms to notify you when your storage utilization exceeds a certain threshold.

Metric	Description	Time Period	Units	Type
`Outpost` `talByte`	The total provisioned capacity in bytes for an Outpost.	5 minutes	Bytes	S3 on Outposts
`Outpostl` `eeBytes`	The count of free bytes available on an Outpost to store customer data.	5 minutes	Bytes	S3 on Outposts
`BucketU` `dBytes`	The total size of all objects for the given bucket.	5 minutes	Bytes	S3 on Outposts. Direct S3 usage only.
`Accountl` `edBytes`	The total size of all objects for the specified Outposts account.	5 minutes	Bytes	S3 on Outposts. Direct S3 usage only.
`BytesPe` `ingRepl` `ation`	The total number of bytes of objects that are pending replication for a given	5 minutes	Bytes	Optional. For S3 Replication on Outposts.

Metric	Description	Time Period	Units	Type
	replication rule. For more information about how to enable replication metrics, see Creating replication rules between Outposts.			
OperationsPendingReplication	The total number of operations that are pending replication for a given replication rule. For more information about how to enable replication metrics, see Creating replication rules between Outposts.	5 minutes	Counts	Optional. For S3 Replication on Outposts.
ReplicationLatency	The current number of seconds delay by which the replication destination bucket is behind the source bucket for a given replication rule. For more information about how to enable replication metrics, see Creating replication rules between Outposts.	5 minutes	Seconds	Optional. For S3 Replication on Outposts.

Receiving S3 on Outposts event notifications by using Amazon CloudWatch Events

You can use CloudWatch Events to create a rule for any Amazon S3 on Outposts API event. When you create a rule, you can choose to get notified through all supported CloudWatch targets, including Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), and AWS Lambda. For more information, see the list of AWS services that can be targets for CloudWatch Events in the *Amazon CloudWatch Events User Guide*. To choose a target

service to work with your S3 on Outposts, see Creating a CloudWatch Events rule that triggers on an AWS API call using AWS CloudTrail in the *Amazon CloudWatch Events User Guide*.

> **Note**
>
> For S3 on Outposts object operations, AWS API call events sent by CloudTrail will match your rules only if you have trails (optionally with event selectors) configured to receive those events. For more information, see Working with CloudTrail log files in the *AWS CloudTrail User Guide*.

Example

The following is a sample rule for the `DeleteObject` operation. To use this sample rule, replace *amzn-s3-demo-bucket1* with the name of your S3 on Outposts bucket.

```
{
  "source": [
    "aws.s3-outposts"
  ],
  "detail-type": [
    "AWS API call through CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "s3-outposts.amazonaws.com"
    ],
    "eventName": [
      "DeleteObject"
    ],
    "requestParameters": {
      "bucketName": [
        "amzn-s3-demo-bucket1"
      ]
    }
  }
}
```

Monitoring S3 on Outposts with AWS CloudTrail logs

Amazon S3 on Outposts is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in S3 on Outposts. You can use AWS CloudTrail to get information about S3 on Outposts bucket-level and object-level requests to audit and log your S3 on Outposts event activity.

To enable CloudTrail data events for all your Outposts buckets or for a list of specific Outposts buckets, you must create a trail manually in CloudTrail. For more information about CloudTrail log file entries, see S3 on Outposts log file entries.

For a complete list of CloudTrail data events for S3 on Outposts, see Amazon S3 data events in CloudTrail in the *Amazon S3 User Guide*.

Note

- It's a best practice to create a lifecycle policy for your AWS CloudTrail data event Outposts bucket. Configure the lifecycle policy to periodically remove log files after the period of time that you need to audit them. Doing so reduces the amount of data that Amazon Athena analyzes for each query. For more information, see Creating and managing a lifecycle configuration for your Amazon S3 on Outposts bucket.
- For examples of how to query CloudTrail logs, see the *AWS Big Data Blog* post Analyze Security, Compliance, and Operational Activity Using AWS CloudTrail and Amazon Athena.

Enable CloudTrail logging for objects in an S3 on Outposts bucket

You can use the Amazon S3 console to configure an AWS CloudTrail trail to log data events for objects in an Amazon S3 on Outposts bucket. CloudTrail supports logging S3 on Outposts object-level API operations such as `GetObject`, `DeleteObject`, and `PutObject`. These events are called *data events*.

By default, CloudTrail trails don't log data events. However, you can configure trails to log data events for S3 on Outposts buckets that you specify or to log data events for all the S3 on Outposts buckets in your AWS account.

CloudTrail does not populate data events in the CloudTrail event history. Additionally, not all S3 on Outposts bucket–level API operations are populated in the CloudTrail event history. For more information about how to query CloudTrail logs, see Using Amazon CloudWatch Logs filter patterns and Amazon Athena to query CloudTrail logs on the AWS Knowledge Center.

To configure a trail to log data events for an S3 on Outposts bucket, you can use either the AWS CloudTrail console or the Amazon S3 console. If you are configuring a trail to log data events for all the S3 on Outposts buckets in your AWS account, it's easier to use the CloudTrail console. For information about using the CloudTrail console to configure a trail to log S3 on Outposts data events, see Data events in the *AWS CloudTrail User Guide*.

Important

Additional charges apply for data events. For more information, see AWS CloudTrail pricing.

The following procedure shows how to use the Amazon S3 console to configure a CloudTrail trail to log data events for an S3 on Outposts bucket.

Note

The AWS account that creates the bucket owns it and is the only one that can configure S3 on Outposts data events to be sent to AWS CloudTrail.

To enable CloudTrail data events logging for objects in an S3 on Outposts bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Outposts buckets**.
3. Choose the name of the Outposts bucket whose data events you want to log by using CloudTrail.
4. Choose **Properties**.
5. In the **AWS CloudTrail data events** section, choose **Configure in CloudTrail**.

   The AWS CloudTrail console opens.

You can create a new CloudTrail trail or reuse an existing trail and configure S3 on Outposts data events to be logged in your trail.

6. On the CloudTrail console **Dashboard** page, choose **Create trail**.
7. On the **Step 1 Choose trail attributes** page, provide a name for the trail, choose an S3 bucket to store trail logs, specify any other settings that you want, and then choose **Next**.
8. On the **Step 2 Choose log events** page, under **Event type**, choose **Data events**.

   For **Data event type**, choose **S3 Outposts**. Choose **Next**.

   > **Note**
   >
   > - When you create a trail and configure data event logging for S3 on Outposts, you must specify the data event type correctly.
   >   - If you use the CloudTrail console, choose **S3 Outposts** for **Data event type**. For information about how to create trails in the CloudTrail console, see Creating and updating a trail with the console in the *AWS CloudTrail User Guide*. For information about how to configure S3 on Outposts data event logging in the CloudTrail console, see Logging data events for Amazon S3 Objects in the *AWS CloudTrail User Guide*.
   >   - If you use the AWS Command Line Interface (AWS CLI) or the AWS SDKs, set the `resources.type` field to `AWS::S3Outposts::Object`. For more information about how to log S3 on Outposts data events with the AWS CLI, see Log S3 on Outposts events in the *AWS CloudTrail User Guide*.
   > - If you use the CloudTrail console or the Amazon S3 console to configure a trail to log data events for an S3 on Outposts bucket, the Amazon S3 console shows that object-level logging is enabled for the bucket.

9. On the **Step 3 Review and create** page, review the trail attributes and the log events that you configured. Then, choose **Create trail**.

To disable CloudTrail data events logging for objects in an S3 on Outposts bucket

1. Sign in to the AWS Management Console and open the CloudTrail console at https://console.aws.amazon.com/cloudtrail/.
2. In the left navigation pane, choose **Trails**.

3. Choose the name of the trail that you created to log events for your S3 on Outposts bucket.
4. On the details page for your trail, choose **Stop logging** in the upper-right corner.
5. In the dialog box that appears, choose **Stop logging**.

Amazon S3 on Outposts AWS CloudTrail log file entries

Amazon S3 on Outposts management events are available via AWS CloudTrail. In addition, you can optionally enable logging for data events in AWS CloudTrail.

A *trail* is a configuration that enables delivery of events as log files to an S3 bucket in a Region that you specify. CloudTrail logs for your Outposts buckets include a new field, `edgeDeviceDetails`, which identifies the Outpost where the specified bucket is located.

Additional log fields include the requested action, the date and time of the action, and the request parameters. CloudTrail log files are not an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates a PutObject action on `s3-outposts`.

```
{
      "eventVersion": "1.08",
      "userIdentity": {
        "type": "IAMUser",
        "principalId": "111122223333",
        "arn": "arn:aws:iam::111122223333:user/yourUserName",
        "accountId": "222222222222",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "yourUserName"
      },
      "eventTime": "2020-11-30T15:44:33Z",
      "eventSource": "s3-outposts.amazonaws.com",
      "eventName": "PutObject",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "26.29.66.20",
      "userAgent": "aws-cli/1.18.39 Python/3.4.10 Darwin/18.7.0 botocore/1.15.39",
      "requestParameters": {
        "expires": "Wed, 21 Oct 2020 07:28:00 GMT",
        "Content-Language": "english",
        "x-amz-server-side-encryption-customer-key-MD5": "wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY",
```

```
        "ObjectCannedACL": "BucketOwnerFullControl",
        "x-amz-server-side-encryption": "Aes256",
        "Content-Encoding": "gzip",
        "Content-Length": "10",
        "Cache-Control": "no-cache",
        "Content-Type": "text/html; charset=UTF-8",
        "Content-Disposition": "attachment",
        "Content-MD5": "je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY",
        "x-amz-storage-class": "Outposts",
        "x-amz-server-side-encryption-customer-algorithm": "Aes256",
        "bucketName": "amzn-s3-demo-bucket1",
        "Key": "path/upload.sh"
      },
      "responseElements": {
        "x-amz-server-side-encryption-customer-key-MD5": "wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY",
        "x-amz-server-side-encryption": "Aes256",
        "x-amz-version-id": "001",
        "x-amz-server-side-encryption-customer-algorithm": "Aes256",
        "ETag": "d41d8cd98f00b204e9800998ecf8427f"
      },
      "additionalEventData": {
        "CipherSuite": "ECDHE-RSA-AES128-SHA",
        "bytesTransferredIn": 10,
        "x-amz-id-2": "29xXQBV20
+xOHKItvzY1suLv1i6A52E0zOX159fpfsItYd58JhXwKxXAXI4IQkp6",
        "SignatureVersion": "SigV4",
        "bytesTransferredOut": 20,
        "AuthenticationMethod": "AuthHeader"
      },
      "requestID": "8E96D972160306FA",
      "eventID": "ee3b4e0c-ab12-459b-9998-0a5a6f2e4015",
      "readOnly": false,
      "resources": [
        {
          "accountId": "222222222222",
          "type": "AWS::S3Outposts::Object",
          "ARN": "arn:aws:s3-outposts:us-east-1:YYY:outpost/op-01ac5d28a6a232904/
bucket/path/upload.sh"
        },
        {
          "accountId": "222222222222",
          "type": "AWS::S3Outposts::Bucket",
```

```
          "ARN": "arn:aws:s3-outposts:us-east-1:YYY:outpost/op-01ac5d28a6a232904/
bucket/"
        }
      ],
      "eventType": "AwsApiCall",
      "managementEvent": false,
      "recipientAccountId": "444455556666",
      "sharedEventID": "02759a4c-c040-4758-b84b-7cbaaf17747a",
      "edgeDeviceDetails": {
        "type": "outposts",
        "deviceId": "op-01ac5d28a6a232904"
      },
      "eventCategory": "Data"
    }
```

Developing with Amazon S3 on Outposts

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts and easily store and retrieve objects on premises for applications that require local data access, local data processing, and data residency. S3 on Outposts provides a new storage class, S3 Outposts (0UTP0STS), which uses the Amazon S3 APIs, and is designed to store data durably and redundantly across multiple devices and servers on your AWS Outposts. You communicate with your Outpost bucket by using an access point and endpoint connection over a virtual private cloud (VPC). You can use the same APIs and features on Outpost buckets as you do on Amazon S3 buckets, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. For more information, see What is Amazon S3 on Outposts?

The following topics provide information about developing with S3 on Outposts.

Topics

- S3 on Outposts supported regions
- Amazon S3 on Outposts API operations
- Configure the S3 control client for S3 on Outposts by using the SDK for Java
- Making requests to S3 on Outposts over IPv6

S3 on Outposts supported regions

S3 on Outposts is supported in the following AWS Regions.

- US East (N. Virginia) (us-east-1)
- US East (Ohio) (us-east-2)
- US West (N. California) (us-west-1)
- US West (Oregon) (us-west-2)
- Africa (Cape Town) (af-south-1)
- Asia Pacific (Jakarta) (ap-sourtheast-3)
- Asia Pacific (Mumbai) (ap-south-1)
- Asia Pacific (Osaka) (ap-northeast-3)

- Asia Pacific (Seoul) (ap-northeast-2)
- Asia Pacific (Singapore) (ap-southeast-1)
- Asia Pacific (Sydney) (ap-southeast-2)
- Asia Pacific (Tokyo) (ap-northeast-1)
- Canada (Central) (ca-central-1)
- Europe (Frankfurt) (eu-central-1)
- Europe (Ireland) (eu-west-1)
- Europe (London) (eu-west-2)
- Europe (Milan) (eu-south-1)
- Europe (Paris) (eu-west-3)
- Europe (Stockholm) (eu-north-1)
- Israel (Tel Aviv) (il-central-1)
- Middle East (Bahrain) (me-south-1)
- South America (São Paulo) (sa-east-1)
- AWS GovCloud (US-East) (us-gov-east-1)
- AWS GovCloud (US-West) (us-gov-west-1)

Amazon S3 on Outposts API operations

This topic lists the Amazon S3, Amazon S3 Control, and Amazon S3 on Outposts API operations that you can use with Amazon S3 on Outposts.

Topics

- Amazon S3 API operations for managing objects
- Amazon S3 Control API operations for managing buckets
- S3 on Outposts API operations for managing Outposts

Amazon S3 API operations for managing objects

S3 on Outposts is designed to use the same object API operations as Amazon S3. You must use access points to access any object in an Outpost bucket. When you use an object API operation with S3 on Outposts, you provide either the Outposts access point Amazon Resource Name (ARN) or the

access point alias. For more information about access point aliases, see Using a bucket-style alias for your S3 on Outposts bucket access point.

Amazon S3 on Outposts supports the following Amazon S3 API operations:

- AbortMultipartUpload
- CompleteMultipartUpload
- CopyObject
- CreateMultipartUpload
- DeleteObject
- DeleteObjects
- DeleteObjectTagging
- GetObject
- GetObjectTagging
- HeadBucket
- HeadObject
- ListMultipartUploads
- ListObjects
- ListObjectsV2
- ListObjectVersions
- ListParts
- PutObject
- PutObjectTagging
- UploadPart
- UploadPartCopy

Amazon S3 Control API operations for managing buckets

S3 on Outposts supports the following Amazon S3 Control API operations for working with buckets.

- CreateAccessPoint
- CreateBucket

- DeleteAccessPoint
- DeleteAccessPointPolicy
- DeleteBucket
- DeleteBucketLifecycleConfiguration
- DeleteBucketPolicy
- DeleteBucketReplication
- DeleteBucketTagging
- GetAccessPoint
- GetAccessPointPolicy
- GetBucket
- GetBucketLifecycleConfiguration
- GetBucketPolicy
- GetBucketReplication
- GetBucketTagging
- GetBucketVersioning
- ListAccessPoints
- ListRegionalBuckets
- PutAccessPointPolicy
- PutBucketLifecycleConfiguration
- PutBucketPolicy
- PutBucketReplication
- PutBucketTagging
- PutBucketVersioning

S3 on Outposts API operations for managing Outposts

S3 on Outposts supports the following Amazon S3 on Outposts API operations for managing endpoints.

- CreateEndpoint
- DeleteEndpoint
- ListEndpoints

- ListOutpostsWithS3
- ListSharedEndpoints

Configure the S3 control client for S3 on Outposts by using the SDK for Java

The following example configures the Amazon S3 control client for Amazon S3 on Outposts by using the AWS SDK for Java. To use this example, replace each *user input placeholder* with your own information.

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;

public AWSS3Control createS3ControlClient() {

    String accessKey = AWSAccessKey;
    String secretKey = SecretAccessKey;
    BasicAWSCredentials awsCreds = new BasicAWSCredentials(accessKey, secretKey);

    return AWSS3ControlClient.builder().enableUseArnRegion()
            .withCredentials(new AWSStaticCredentialsProvider(awsCreds))
            .build();

}
```

Making requests to S3 on Outposts over IPv6

Amazon S3 on Outposts and S3 on Outposts dual-stack endpoints support requests to S3 on Outposts buckets using either the IPv6 or IPv4 protocol. With IPv6 support for S3 on Outposts, you can access and operate your buckets and control plane resources through S3 on Outposts APIs over IPv6 networks.

> **Note**
>
> S3 on Outposts object actions (such as `PutObject` or `GetObject`) aren't supported over IPv6 networks.

There are no additional charges for accessing S3 on Outposts over IPv6 networks. For more information about S3 on Outposts, see S3 on Outposts pricing.

Topics

- Getting started with IPv6
- Using dual-stack endpoints to make requests over an IPv6 network
- Using IPv6 addresses in IAM policies
- Testing IP address compatibility
- Using IPv6 with AWS PrivateLink
- Using S3 on Outposts dual-stack endpoints

Getting started with IPv6

To make a request to an S3 on Outposts bucket over IPv6, you must use a dual-stack endpoint. The next section describes how to make requests over IPv6 by using dual-stack endpoints.

The following are important considerations before trying to access an S3 on Outposts bucket over IPv6:

- The client and the network accessing the bucket must be enabled to use IPv6.
- Both virtual hosted-style and path style requests are supported for IPv6 access. For more information, see Using S3 on Outposts dual-stack endpoints.
- If you use source IP address filtering in your AWS Identity and Access Management (IAM) user or S3 on Outposts bucket policies, you must update the policies to include IPv6 address ranges.

> **Note**
>
> This requirement only applies to S3 on Outposts bucket operations and control plane resources across IPv6 networks. Amazon S3 on Outposts object actions aren't supported across IPv6 networks.

- When using IPv6, server access log files output IP addresses in an IPv6 format. You must update existing tools, scripts, and software that you use to parse S3 on Outposts log files, so that they can parse the IPv6 formatted remote IP addresses. The updated tools, scripts, and software will then correctly parse the IPv6 formatted remote IP addresses.

Using dual-stack endpoints to make requests over an IPv6 network

To make requests with S3 on Outposts API calls over IPv6, you can use dual-stack endpoints via AWS CLI or AWS SDK. The Amazon S3 control API operations and S3 on Outposts API operations work the same way whether you're accessing S3 on Outposts over an IPv6 protocol or IPv4 protocol. However, be aware that S3 on Outposts object actions (such as `PutObject` or `GetObject`) aren't supported over IPv6 networks.

When using the AWS Command Line Interface (AWS CLI) and AWS SDKs, you can use a parameter or flag to change to a dual-stack endpoint. You can also specify the dual-stack endpoint directly as an override of the S3 on Outposts endpoint in the configuration file.

You can use a dual-stack endpoint to access an S3 on Outposts bucket over IPv6 from any of the following:

- The AWS CLI, see Using dual-stack endpoints from the AWS CLI.
- The AWS SDKs, see Using S3 on Outposts dual-stack endpoints from the AWS SDKs.

Using IPv6 addresses in IAM policies

Before trying to access an S3 on Outposts bucket using an IPv6 protocol, make sure that IAM users or S3 on Outposts bucket policies used for IP address filtering are updated to include IPv6 address ranges. If IP address filtering policies aren't updated to handle IPv6 addresses, you can lose access to an S3 on Outposts bucket while trying to use the IPv6 protocol.

IAM policies that filter IP addresses use IP address condition operators. The following S3 on Outposts bucket policy identifies the 54.240.143.* IP range of allowed IPv4 addresses by using IP address condition operators. Any IP addresses outside of this range will be denied access to the S3 on Outposts bucket (`DOC-EXAMPLE-BUCKET`). Since all IPv6 addresses are outside of the allowed range, this policy prevents IPv6 addresses from being able to access `DOC-EXAMPLE-BUCKET`.

JSON

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

```
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3-outposts:*",
            "Resource": "arn:aws:s3-outposts:us-
east-1:111122223333:outpost/OUTPOSTS-ID/bucket/DOC-EXAMPLE-BUCKET/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "54.240.143.0/24"
                }
            }
        }
    ]
}
```

You can modify the S3 on Outposts bucket policy's `Condition` element to allow both IPv4 (`54.240.143.0/24`) and IPv6 (`2001:DB8:1234:5678::/64`) address ranges as shown in the following example. You can use the same type of `Condition` block shown in the example to update both your IAM user and bucket policies.

```
"Condition": {
  "IpAddress": {
     "aws:SourceIp": [
       "54.240.143.0/24",
        "2001:DB8:1234:5678::/64"
      ]
   }
 }
```

Before using IPv6 you must update all relevant IAM user and bucket policies that use IP address filtering to allow IPv6 address ranges. We recommend that you update your IAM policies with your organization's IPv6 address ranges in addition to your existing IPv4 address ranges. For an example of a bucket policy that allows access over both IPv6 and IPv4, see Restrict access to specific IP addresses.

You can review your IAM user policies using the IAM console at https://console.aws.amazon.com/iam/. For more information about IAM, see the IAM User Guide. For information about editing S3 on Outposts bucket policies, see Adding or editing a bucket policy for an Amazon S3 on Outposts bucket.

Testing IP address compatibility

If you're using a Linux or Unix instance, or macOS X platform, you can test your access to a dual-stack endpoint over IPv6. For example, to test the connection to Amazon S3 on Outposts endpoints over IPv6, use the `dig` command:

```
dig s3-outposts.us-west-2.api.aws AAAA +short
```

If your dual-stack endpoint over an IPv6 network is properly set up, the `dig` command returns the connected IPv6 addresses. For example:

```
dig s3-outposts.us-west-2.api.aws AAAA +short

2600:1f14:2588:4800:b3a9:1460:159f:ebce

2600:1f14:2588:4802:6df6:c1fd:ef8a:fc76

2600:1f14:2588:4801:d802:8ccf:4e04:817
```

Using IPv6 with AWS PrivateLink

S3 on Outposts supports the IPv6 protocol for AWS PrivateLink services and endpoints. With AWS PrivateLink support for the IPv6 protocol, you can connect to service endpoints within your VPC over IPv6 networks, from either on-premises or other private connections. The IPv6 support for AWS PrivateLink for S3 on Outposts also allows you to integrate AWS PrivateLink with dual-stack endpoints. For steps on how to enable IPv6 for AWS PrivateLink, see Expedite your IPv6 adoption with AWS PrivateLink services and endpoints.

> **Note**
>
> To update the supported IP address type from IPv4 to IPv6, see Modify the supported IP address type in the *AWS PrivateLink User Guide*.

Using IPv6 with AWS PrivateLink

If you're using AWS PrivateLink with IPv6, you must create an IPv6 or dual-stack VPC interface endpoint. For general steps on how to create a VPC endpoint using the AWS Management Console, see Access an AWS service using an interface VPC endpoint in the *AWS PrivateLink User Guide*.

AWS Management Console

Use the following procedure to create an interface VPC endpoint that connects to S3 on Outposts.

1. Sign in to the AWS Management Console and open the VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Endpoints**.
3. Choose **Create endpoint**.
4. For **Service category**, choose **AWS services**.
5. For **Service name**, choose the S3 on Outposts service (**com.amazonaws.us-east-1.s3-outposts**).
6. For VPC, choose the VPC from which you'll access S3 on Outposts.
7. For **Subnets**, choose one subnet per Availability Zone from which you'll access S3 on Outposts. You can't select multiple subnets from the same Availability Zone. For each subnet that you select, a new endpoint network interface is created. By default, IP addresses from the subnet IP address ranges are assigned to the endpoint network interfaces. To designate an IP address for an endpoint network interface, choose **Designate IP addresses** and enter an IPv6 address from the subnet address range.
8. For **IP address type**, choose **Dualstack**. Assign both IPv4 and IPv6 addresses to your endpoint network interfaces. This option is supported only if all selected subnets have both IPv4 and IPv6 address ranges.
9. For **Security groups**, choose the security groups to associate with the endpoint network interfaces for the VPC endpoint. By default, the default security group is associated with the VPC.
10. For **Policy**, choose **Full access** to allow all operations by all principals on all resources over the VPC endpoint. Otherwise, choose **Custom** to attach a VPC endpoint policy that controls the permissions that principals have toperform actions on resources over the VPC endpoint. This option is available only if the service supports VPC endpoint policies. For more information, see Endpoint policies.
11. (Optional) To add a tag, choose **Add new tag** and enter the tag key and the tag value.
12. Choose **Create endpoint**.

Example– S3 on Outposts bucket policy

To allow S3 on Outposts to interact with your VPC endpoints, you can then update your S3 on Outposts policy like this:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3-outposts:*",
            "Resource": "*",
            "Principal": "*"
        }
    ]
}
```

AWS CLI

> **Note**
>
> To enable the IPv6 network on your VPC endpoint, you must have `IPv6` set for the `SupportedIpAddressType` filter for S3 on Outposts.

The following example uses the `create-vpc-endpoint` command to create a new dual-stack interface endpoint.

```
aws ec2 create-vpc-endpoint \
--vpc-id vpc-12345678 \
--vpc-endpoint-type Interface \
--service-name com.amazonaws.us-east-1.s3-outposts \
--subnet-id subnet-12345678 \
--security-group-id sg-12345678 \
--ip-address-type dualstack \
--dns-options "DnsRecordIpType=dualstack"
```

Depending on the AWS PrivateLink service configuration, newly created endpoint connections might need to be accepted by the VPC endpoint service provider before they can be used. For more information, see Accept and reject endpoint connection requests in the *AWS PrivateLink User Guide*.

The following example uses the `modify-vpc-endpoint` command to update the IPv-only VPC endpoint to a dual-stack endpoint. The dual-stack endpoint allows access to both the IPv4 and IPv6 networks.

```
aws ec2 modify-vpc-endpoint \
--vpc-endpoint-id vpce-12345678 \
--add-subnet-ids subnet-12345678 \
--remove-subnet-ids subnet-12345678 \
--ip-address-type dualstack \
--dns-options "DnsRecordIpType=dualstack"
```

For more information about how to enable the IPv6 network for AWS PrivateLink, see Expedite your IPv6 adoption with AWS PrivateLink services and endpoints.

Using S3 on Outposts dual-stack endpoints

S3 on Outposts dual-stack endpoints support requests to S3 on Outposts buckets over IPv6 and IPv4. This section describes how to use S3 on Outposts dual-stack endpoints.

Topics

- S3 on Outposts dual-stack endpoints
- Using dual-stack endpoints from the AWS CLI
- Using S3 on Outposts dual-stack endpoints from the AWS SDKs

S3 on Outposts dual-stack endpoints

When you make a request to a dual-stack endpoint, the S3 on Outposts bucket URL resolves to an IPv6 or an IPv4 address. For more information about accessing an S3 on Outposts bucket over IPv6, see Making requests to S3 on Outposts over IPv6.

To access an S3 on Outposts bucket through a dual-stack endpoint, use a path-style endpoint name. S3 on Outposts supports only Regional dual-stack endpoint names, which means that you must specify the Region as part of the name.

For a dual-stack path-style FIPs endpoint, use the following naming convention:

```
s3-outposts-fips.region.api.aws
```

For dual-stack non-FIPS endpoint, use the following naming convention:

```
s3-outposts.region.api.aws
```

> **Note**
>
> Virtual hosted-style endpoint names aren't supported in S3 on Outposts.

Using dual-stack endpoints from the AWS CLI

This section provides examples of AWS CLI commands used to make requests to a dual-stack endpoint. For instructions on setting up the AWS CLI, see Getting started by using the AWS CLI and SDK for Java.

You set the configuration value `use_dualstack_endpoint` to `true` in a profile in your AWS Config file to direct all Amazon S3 requests made by the `s3` and `s3api` AWS CLI commands to the dual-stack endpoint for the specified Region. You specify the Region in the configuration file or in a command using the `--region` option.

When using dual-stack endpoints with the AWS CLI, only `path` addressing style is supported. The addressing style, set in the configuration file, determines whether the bucket name is in the hostname or in the URL. For more information, see s3outposts in the *AWS CLI User Guide*.

To use a dual-stack endpoint via the AWS CLI, use the `--endpoint-url` parameter with the `http://s3.dualstack.region.amazonaws.com` or `https://s3-outposts-fips.region.api.aws` endpoint for any `s3control` or `s3outposts` commands.

For example:

```
$  aws s3control list-regional-buckets --endpoint-url https://s3-
outposts.region.api.aws
```

Using S3 on Outposts dual-stack endpoints from the AWS SDKs

This section provides examples of how to access a dual-stack endpoint by using the AWS SDKs.

AWS SDK for Java 2.x dual-stack endpoint example

The following examples show how to use the `S3ControlClient` and `S3OutpostsClient` classes to enable dual-stack endpoints when creating an S3 on Outposts client using the AWS SDK for Java

2.x. For instructions on creating and testing a working Java example for Amazon S3 on Outposts, see Getting started by using the AWS CLI and SDK for Java.

Example– Create an S3ControlClient class with dual-stack endpoints enabled

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListRegionalBucketsRequest;
import software.amazon.awssdk.services.s3control.model.ListRegionalBucketsResponse;
import software.amazon.awssdk.services.s3control.model.S3ControlException;


public class DualStackEndpointsExample1 {

    public static void main(String[] args) {
        Region clientRegion = Region.of("us-east-1");
        String accountId = "111122223333";
        String navyId = "9876543210";

        try {
            // Create an S3ControlClient with dual-stack endpoints enabled.
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                                                             .region(clientRegion)
                                                             .dualstackEnabled(true)
                                                             .build();
            ListRegionalBucketsRequest listRegionalBucketsRequest =
 ListRegionalBucketsRequest.builder()

       .accountId(accountId)

       .outpostId(navyId)

       .build();

            ListRegionalBucketsResponse listBuckets =
 s3ControlClient.listRegionalBuckets(listRegionalBucketsRequest);
            System.out.printf("ListRegionalBuckets Response: %s%n",
 listBuckets.toString());
       } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 on Outposts
 couldn't process
            // it, so it returned an error response.
```

```
            e.printStackTrace();
        }
        catch (S3ControlException e) {
            // Unknown exceptions will be thrown as an instance of this type.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 on Outposts couldn't be contacted for a response, or the
 client
            // couldn't parse the response from Amazon S3 on Outposts.
            e.printStackTrace();
        }
    }
}
```

Example– Create an S3OutpostsClient with dual-stack endpoints enabled

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3outposts.S3OutpostsClient;
import software.amazon.awssdk.services.s3outposts.model.ListEndpointsRequest;
import software.amazon.awssdk.services.s3outposts.model.ListEndpointsResponse;
import software.amazon.awssdk.services.s3outposts.model.S3OutpostsException;


public class DualStackEndpointsExample2 {

    public static void main(String[] args) {
        Region clientRegion = Region.of("us-east-1");

        try {
            // Create an S3OutpostsClient with dual-stack endpoints enabled.
            S3OutpostsClient s3OutpostsClient = S3OutpostsClient.builder()
                                                                .region(clientRegion)
                                                                .dualstackEnabled(true)
                                                                .build();
            ListEndpointsRequest listEndpointsRequest =
 ListEndpointsRequest.builder().build();

            ListEndpointsResponse listEndpoints =
 s3OutpostsClient.listEndpoints(listEndpointsRequest);
            System.out.printf("ListEndpoints Response: %s%n",
 listEndpoints.toString());
```

```
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 on Outposts
 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        }
        catch (S3OutpostsException e) {
            // Unknown exceptions will be thrown as an instance of this type.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 on Outposts couldn't be contacted for a response, or the
 client
            // couldn't parse the response from Amazon S3 on Outposts.
            e.printStackTrace();
        }
    }
}
```

If you're using the AWS SDK for Java 2.x on Windows, you might have to set the following Java virtual machine (JVM) property:

```
java.net.preferIPv6Addresses=true
```