AWS Marketplace Metering Service

API Reference

API Version 2016-01-14

AWS Marketplace Metering Service: API Reference

Table of Contents

Welcome

This reference provides descriptions of the low-level AWS Marketplace Metering Service API.

AWS Marketplace sellers can use this API to submit usage data for custom usage dimensions.

For information on the permissions you need to use this API, see AWS Marketplace metering and entitlement API permissions in the *AWS Marketplace Seller Guide.*

Submitting Metering Records

- *MeterUsage* - Submits the metering record for an AWS Marketplace product. MeterUsage is called from an EC2 instance or a container running on EKS or ECS.
- *BatchMeterUsage* - Submits the metering record for a set of customers. BatchMeterUsage is called from a software-as-a-service (SaaS) application.

Accepting New Customers

- *ResolveCustomer* - Called by a SaaS application during the registration process. When a buyer visits your website during the registration process, the buyer submits a Registration Token through the browser. The Registration Token is resolved through this API to obtain a CustomerIdentifier and Product Code.

Entitlement and Metering for Paid Container Products

- Paid container software products sold through AWS Marketplace must integrate with the AWS Marketplace Metering Service and call the RegisterUsage operation for software entitlement and metering. Free and BYOL products for Amazon ECS or Amazon EKS aren't required to call RegisterUsage, but you can do so if you want to receive usage data in your seller reports. For more information on using the RegisterUsage operation, see Container-Based Products.

BatchMeterUsage API calls are captured by AWS CloudTrail. You can use Cloudtrail to verify that the SaaS metering records that you sent are accurate by searching for records with the eventName of BatchMeterUsage. You can also use CloudTrail to audit records over time. For more information, see the *AWS CloudTrail User Guide*.

This document was last published on October 6, 2021.

Actions

The following actions are supported:

BatchMeterUsage

BatchMeterUsage is called from a SaaS application listed on the AWS Marketplace to post metering records for a set of customers.

For identical requests, the API is idempotent; requests can be retried with the same records or a subset of the input records.

Every request to BatchMeterUsage is for one product. If you need to meter usage for multiple products, you must make multiple calls to BatchMeterUsage.

BatchMeterUsage can process up to 25 UsageRecords at a time.

A UsageRecord can optionally include multiple usage allocations, to provide customers with usagedata split into buckets by tags that you define (or allow the customer to define).

BatchMeterUsage requests must be less than 1MB in size.

Note
For an example of using BatchMeterUsage, see BatchMeterUsage code example in the *AWS Marketplace Seller Guide*.

Request Syntax

```
{
   "ProductCode": "string",
   "UsageRecords": [
      {
         "CustomerIdentifier": "string",
         "Dimension": "string",
         "Quantity": number,
         "Timestamp": number,
         "UsageAllocations": [
            {
               "AllocatedUsageQuantity": number,
               "Tags": [
                  {
                     "Key": "string",
                     "Value": "string"
                  }
               ]
            }
         ]
      }
   ]
}
```

Request Parameters

The request accepts the following data in JSON format.

Note
In the following list, the required parameters are described first.

ProductCode (p. 3)

Product code is used to uniquely identify a product in AWS Marketplace. The product code should be the same as the one used during the publishing of a new product.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[\s\S]+`

Required: Yes

UsageRecords (p. 3)

The set of UsageRecords to submit. BatchMeterUsage accepts up to 25 UsageRecords at a time.

Type: Array of UsageRecord (p. 21) objects

Array Members: Minimum number of 0 items. Maximum number of 25 items.

Required: Yes

Response Syntax

```
{
   "Results": [
      {
         "MeteringRecordId": "string",
         "Status": "string",
         "UsageRecord": {
            "CustomerIdentifier": "string",
            "Dimension": "string",
            "Quantity": number,
            "Timestamp": number,
            "UsageAllocations": [
               {
                  "AllocatedUsageQuantity": number,
                  "Tags": [
                     {
                        "Key": "string",
                        "Value": "string"
                     }
                  ]
               }
            ]
         }
      }
   ],
   "UnprocessedRecords": [
      {
         "CustomerIdentifier": "string",
         "Dimension": "string",
         "Quantity": number,
         "Timestamp": number,
         "UsageAllocations": [
            {
               "AllocatedUsageQuantity": number,
               "Tags": [
                  {
                     "Key": "string",
                     "Value": "string"
                  }
               ]
            }
         ]
      }
   ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Results (p. 4)

Contains all UsageRecords processed by BatchMeterUsage. These records were either honored by AWS Marketplace Metering Service or were invalid.

Type: Array of UsageRecordResult (p. 23) objects

UnprocessedRecords (p. 4)

Contains all UsageRecords that were not processed by BatchMeterUsage. This is a list of UsageRecords. You can retry the failed request by making another BatchMeterUsage call with this list as input in the BatchMeterUsageRequest.

Type: Array of UsageRecord (p. 21) objects

Array Members: Minimum number of 0 items. Maximum number of 25 items.

Errors

For information about the errors that are common to all actions, see Common Errors (p. 24).

DisabledApiException

The API is disabled in the Region.

HTTP Status Code: 400

InternalServiceErrorException

An internal error has occurred. Retry your request. If the problem persists, post a message with details on the AWS forums.

HTTP Status Code: 500

InvalidCustomerIdentifierException

You have metered usage for a CustomerIdentifier that does not exist.

HTTP Status Code: 400

InvalidProductCodeException

The product code passed does not match the product code used for publishing the product.

HTTP Status Code: 400

InvalidTagException

The tag is invalid, or the number of tags is greater than 5.

HTTP Status Code: 400

InvalidUsageAllocationsException

The usage allocation objects are invalid, or the number of allocations is greater than 500 for a single usage record.

HTTP Status Code: 400

InvalidUsageDimensionException

The usage dimension does not match one of the UsageDimensions associated with products.

HTTP Status Code: 400

ThrottlingException

The calls to the API are throttled.

HTTP Status Code: 400

TimestampOutOfBoundsException

The timestamp value passed in the meterUsage() is out of allowed range.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

MeterUsage

API to emit metering records. For identical requests, the API is idempotent. It simply returns the metering record ID.

MeterUsage is authenticated on the buyer's AWS account using credentials from the EC2 instance, ECS task, or EKS pod.

MeterUsage can optionally include multiple usage allocations, to provide customers with usage data split into buckets by tags that you define (or allow the customer to define).

Request Syntax

```
{
   "DryRun": boolean,
   "ProductCode": "string",
   "Timestamp": number,
   "UsageAllocations": [
      {
         "AllocatedUsageQuantity": number,
         "Tags": [
            {
               "Key": "string",
               "Value": "string"
            }
         ]
      }
   ],
   "UsageDimension": "string",
   "UsageQuantity": number
}
```

Request Parameters

The request accepts the following data in JSON format.

Note
In the following list, the required parameters are described first.

ProductCode (p. 7)

Product code is used to uniquely identify a product in AWS Marketplace. The product code should be the same as the one used during the publishing of a new product.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[\s\S]+`

Required: Yes

Timestamp (p. 7)

Timestamp, in UTC, for which the usage is being reported. Your application can meter usage for up to one hour in the past. Make sure the timestamp value is not before the start of the software usage.

Type: Timestamp

Required: Yes

UsageDimension (p. 7)

It will be one of the fcp dimension name provided during the publishing of the product.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[\s\S]+`

Required: Yes

DryRun (p. 7)

Checks whether you have the permissions required for the action, but does not make the request. If you have the permissions, the request returns DryRunOperation; otherwise, it returns UnauthorizedException. Defaults to `false` if not specified.

Type: Boolean

Required: No

UsageAllocations (p. 7)

The set of UsageAllocations to submit.

The sum of all UsageAllocation quantities must equal the UsageQuantity of the MeterUsage request, and each UsageAllocation must have a unique set of tags (include no tags).

Type: Array of UsageAllocation (p. 20) objects

Array Members: Minimum number of 1 item. Maximum number of 500 items.

Required: No

UsageQuantity (p. 7)

Consumption value for the hour. Defaults to `0` if not specified.

Type: Integer

Valid Range: Minimum value of 0. Maximum value of 2147483647.

Required: No

Response Syntax

```
{
   "MeteringRecordId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

MeteringRecordId (p. 8)

Metering record id.

Type: String

Errors

For information about the errors that are common to all actions, see Common Errors (p. 24).

CustomerNotEntitledException

Exception thrown when the customer does not have a valid subscription for the product.

HTTP Status Code: 400

DuplicateRequestException

A metering record has already been emitted by the same EC2 instance, ECS task, or EKS pod for the given {usageDimension, timestamp} with a different usageQuantity.

HTTP Status Code: 400

InternalServiceErrorException

An internal error has occurred. Retry your request. If the problem persists, post a message with details on the AWS forums.

HTTP Status Code: 500

InvalidEndpointRegionException

The endpoint being called is in a AWS Region different from your EC2 instance, ECS task, or EKS pod. The Region of the Metering Service endpoint and the AWS Region of the resource must match.

HTTP Status Code: 400

InvalidProductCodeException

The product code passed does not match the product code used for publishing the product.

HTTP Status Code: 400

InvalidTagException

The tag is invalid, or the number of tags is greater than 5.

HTTP Status Code: 400

InvalidUsageAllocationsException

The usage allocation objects are invalid, or the number of allocations is greater than 500 for a single usage record.

HTTP Status Code: 400

InvalidUsageDimensionException

The usage dimension does not match one of the UsageDimensions associated with products.

HTTP Status Code: 400

ThrottlingException

The calls to the API are throttled.

HTTP Status Code: 400

TimestampOutOfBoundsException

The timestamp value passed in the meterUsage() is out of allowed range.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

RegisterUsage

Paid container software products sold through AWS Marketplace must integrate with the AWS Marketplace Metering Service and call the RegisterUsage operation for software entitlement and metering. Free and BYOL products for Amazon ECS or Amazon EKS aren't required to call RegisterUsage, but you may choose to do so if you would like to receive usage data in your seller reports. The sections below explain the behavior of RegisterUsage. RegisterUsage performs two primary functions: metering and entitlement.

- *Entitlement*: RegisterUsage allows you to verify that the customer running your paid software is subscribed to your product on AWS Marketplace, enabling you to guard against unauthorized use. Your container image that integrates with RegisterUsage is only required to guard against unauthorized use at container startup, as such a CustomerNotSubscribedException/PlatformNotSupportedException will only be thrown on the initial call to RegisterUsage. Subsequent calls from the same Amazon ECS task instance (e.g. task-id) or Amazon EKS pod will not throw a CustomerNotSubscribedException, even if the customer unsubscribes while the Amazon ECS task or Amazon EKS pod is still running.
- *Metering*: RegisterUsage meters software use per ECS task, per hour, or per pod for Amazon EKS with usage prorated to the second. A minimum of 1 minute of usage applies to tasks that are short lived. For example, if a customer has a 10 node Amazon ECS or Amazon EKS cluster and a service configured as a Daemon Set, then Amazon ECS or Amazon EKS will launch a task on all 10 cluster nodes and the customer will be charged: (10 * hourly_rate). Metering for software use is automatically handled by the AWS Marketplace Metering Control Plane -- your software is not required to perform any metering specific actions, other than call RegisterUsage once for metering of software use to commence. The AWS Marketplace Metering Control Plane will also continue to bill customers for running ECS tasks and Amazon EKS pods, regardless of the customers subscription state, removing the need for your software to perform entitlement checks at runtime.

Request Syntax

```
{
   "Nonce": "string",
   "ProductCode": "string",
   "PublicKeyVersion": number
}
```

Request Parameters

The request accepts the following data in JSON format.

Note
In the following list, the required parameters are described first.

ProductCode (p. 11)

Product code is used to uniquely identify a product in AWS Marketplace. The product code should be the same as the one used during the publishing of a new product.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[\s\S]+`

Required: Yes

PublicKeyVersion (p. 11)

Public Key Version provided by AWS Marketplace

Type: Integer

Valid Range: Minimum value of 1.

Required: Yes

Nonce (p. 11)

(Optional) To scope down the registration to a specific running software instance and guard against replay attacks.

Type: String

Length Constraints: Maximum length of 255.

Pattern: `[\s\S]*`

Required: No

Response Syntax

```
{
   "PublicKeyRotationTimestamp": number,
   "Signature": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

PublicKeyRotationTimestamp (p. 12)

(Optional) Only included when public key version has expired

Type: Timestamp

Signature (p. 12)

JWT Token

Type: String

Pattern: `[\s\S]+`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 24).

CustomerNotEntitledException

Exception thrown when the customer does not have a valid subscription for the product.

HTTP Status Code: 400

DisabledApiException

The API is disabled in the Region.

HTTP Status Code: 400

InternalServiceErrorException

An internal error has occurred. Retry your request. If the problem persists, post a message with details on the AWS forums.

HTTP Status Code: 500

InvalidProductCodeException

The product code passed does not match the product code used for publishing the product.

HTTP Status Code: 400

InvalidPublicKeyVersionException

Public Key version is invalid.

HTTP Status Code: 400

InvalidRegionException

RegisterUsage must be called in the same AWS Region the ECS task was launched in. This prevents a container from hardcoding a Region (e.g. withRegion("us-east-1") when calling RegisterUsage.

HTTP Status Code: 400

PlatformNotSupportedException

AWS Marketplace does not support metering usage from the underlying platform. Currently, Amazon ECS, Amazon EKS, and AWS Fargate are supported.

HTTP Status Code: 400

ThrottlingException

The calls to the API are throttled.

HTTP Status Code: 400

Examples

Example

Below are the sample request and response for RegisterUsage

Sample Request

```
{
    "ProductCode" : "cqcvf9f0ugw8rkbgmf1c9dxyz",
    "PublicKeyVersion": 1,
    "Nonce": "2ead20e4-3e6d-42cd-8f56-24f02d1cc4e1"
}
```

Sample Response

```
{
   "PublicKeyRotationTimestamp": null,
   "Signature": "eyJhbGciOiJQUzI1Ni..."
}

// Where the signature is composed of 3 dot-separated,
// base-64 URL Encoded sections.
// e.g. eyJhbGcVCJ9.eyJzdWIMzkwMjJ9.rrO9Qw0SXRWTe

// Section 1: Header/Algorithm

{
  "alg": "PS256",
  "typ": "JWT"
}

// Section 2: Payload

{
   "ProductCode" : "cqcvf9f0ugw8rkbgmf1c9dxyz",
   "PublicKeyVersion": 1,
   "Nonce": "2ead20e4-3e6d-42cd-8f56-24f02d1cc4e1",
   "PublicKeyRotationTimestamp": null
}

// Section 3: RSA-PSS SHA256 signature

"rrO9Q4FEi3gweH3X4lrt2okf5zwIatUUwERlw016wTy_21Nv8S..."
```

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ResolveCustomer

ResolveCustomer is called by a SaaS application during the registration process. When a buyer visits your website during the registration process, the buyer submits a registration token through their browser. The registration token is resolved through this API to obtain a CustomerIdentifier and product code.

Note
The API needs to called from the seller account id used to publish the SaaS application to successfully resolve the token.
For an example of using ResolveCustomer, see ResolveCustomer code example in the *AWS Marketplace Seller Guide*.

Request Syntax

```
{
   "RegistrationToken": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

Note
In the following list, the required parameters are described first.

RegistrationToken (p. 15)

When a buyer visits your website during the registration process, the buyer submits a registration token through the browser. The registration token is resolved to obtain a CustomerIdentifier and product code.

Type: String

Pattern: `[\s\S]+`

Required: Yes

Response Syntax

```
{
   "CustomerIdentifier": "string",
   "ProductCode": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

CustomerIdentifier (p. 15)

The CustomerIdentifier is used to identify an individual customer in your application. Calls to BatchMeterUsage require CustomerIdentifiers for each UsageRecord.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[\s\S]+`

ProductCode (p. 15)

The product code is returned to confirm that the buyer is registering for your product. Subsequent BatchMeterUsage calls should be made using this product code.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[\s\S]+`

Errors

For information about the errors that are common to all actions, see Common Errors (p. 24).

DisabledApiException

The API is disabled in the Region.

HTTP Status Code: 400

ExpiredTokenException

The submitted registration token has expired. This can happen if the buyer's browser takes too long to redirect to your page, the buyer has resubmitted the registration token, or your application has held on to the registration token for too long. Your SaaS registration website should redeem this token as soon as it is submitted by the buyer's browser.

HTTP Status Code: 400

InternalServiceErrorException

An internal error has occurred. Retry your request. If the problem persists, post a message with details on the AWS forums.

HTTP Status Code: 500

InvalidTokenException

Registration token is invalid.

HTTP Status Code: 400

ThrottlingException

The calls to the API are throttled.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Data Types

The AWSMarketplace Metering API contains several data types that various actions use. This section describes each data type in detail.

Note
The order of each element in a data type structure is not guaranteed. Applications should not assume a particular order.

The following data types are supported:

Tag

Metadata assigned to an allocation. Each tag is made up of a key and a value.

Contents

Note
In the following list, the required parameters are described first.

Key

One part of a key-value pair that makes up a tag. A key is a label that acts like a category for the specific tag values.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 100.

Pattern: `^[a-zA-Z0-9+ -=._:\/@]+$`

Required: Yes

Value

One part of a key-value pair that makes up a tag. A value acts as a descriptor within a tag category (key). The value can be empty or null.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Pattern: `^[a-zA-Z0-9+ -=._:\/@]+$`

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3

UsageAllocation

Usage allocations allow you to split usage into buckets by tags.

Each UsageAllocation indicates the usage quantity for a specific set of tags.

Contents

Note
In the following list, the required parameters are described first.

AllocatedUsageQuantity

The total quantity allocated to this bucket of usage.

Type: Integer

Valid Range: Minimum value of 0. Maximum value of 2147483647.

Required: Yes

Tags

The set of tags that define the bucket of usage. For the bucket of items with no tags, this parameter can be left out.

Type: Array of Tag (p. 19) objects

Array Members: Minimum number of 1 item. Maximum number of 5 items.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3

UsageRecord

A UsageRecord indicates a quantity of usage for a given product, customer, dimension and time.

Multiple requests with the same UsageRecords as input will be deduplicated to prevent double charges.

Contents

Note
In the following list, the required parameters are described first.

CustomerIdentifier

The CustomerIdentifier is obtained through the ResolveCustomer operation and represents an individual buyer in your application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[\s\S]+`

Required: Yes

Dimension

During the process of registering a product on AWS Marketplace, dimensions are specified. These represent different units of value in your application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 255.

Pattern: `[\s\S]+`

Required: Yes

Timestamp

Timestamp, in UTC, for which the usage is being reported.

Your application can meter usage for up to one hour in the past. Make sure the timestamp value is not before the start of the software usage.

Type: Timestamp

Required: Yes

Quantity

The quantity of usage consumed by the customer for the given dimension and time. Defaults to `0` if not specified.

Type: Integer

Valid Range: Minimum value of 0. Maximum value of 2147483647.

Required: No

UsageAllocations

The set of UsageAllocations to submit. The sum of all UsageAllocation quantities must equal the Quantity of the UsageRecord.

Type: Array of UsageAllocation (p. 20) objects

Array Members: Minimum number of 1 item. Maximum number of 500 items.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3

UsageRecordResult

A UsageRecordResult indicates the status of a given UsageRecord processed by BatchMeterUsage.

Contents

Note
In the following list, the required parameters are described first.

MeteringRecordId

The MeteringRecordId is a unique identifier for this metering event.

Type: String

Required: No

Status

The UsageRecordResult Status indicates the status of an individual UsageRecord processed by BatchMeterUsage.

- *Success*- The UsageRecord was accepted and honored by BatchMeterUsage.
- *CustomerNotSubscribed*- The CustomerIdentifier specified is not subscribed to your product. The UsageRecord was not honored. Future UsageRecords for this customer will fail until the customer subscribes to your product.
- *DuplicateRecord*- Indicates that the UsageRecord was invalid and not honored. A previously metered UsageRecord had the same customer, dimension, and time, but a different quantity.

Type: String

Valid Values: `Success | CustomerNotSubscribed | DuplicateRecord`

Required: No

UsageRecord

The UsageRecord that was part of the BatchMeterUsage request.

Type: UsageRecord (p. 21) object

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK for Ruby V3

Common Errors

This section lists the errors common to the API actions of all AWS services. For errors specific to an API action for this service, see the topic for that API action.

AccessDeniedException

You do not have sufficient access to perform this action.

HTTP Status Code: 400

IncompleteSignature

The request signature does not conform to AWS standards.

HTTP Status Code: 400

InternalFailure

The request processing has failed because of an unknown error, exception or failure.

HTTP Status Code: 500

InvalidAction

The action or operation requested is invalid. Verify that the action is typed correctly.

HTTP Status Code: 400

InvalidClientTokenId

The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

InvalidParameterCombination

Parameters that must not be used together were used together.

HTTP Status Code: 400

InvalidParameterValue

An invalid or out-of-range value was supplied for the input parameter.

HTTP Status Code: 400

InvalidQueryParameter

The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400

MalformedQueryString

The query string contains a syntax error.

HTTP Status Code: 404

MissingAction

The request is missing an action or a required parameter.

HTTP Status Code: 400

MissingAuthenticationToken

The request must contain either a valid (registered) AWS access key ID or X.509 certificate.

HTTP Status Code: 403

MissingParameter

A required parameter for the specified action is not supplied.

HTTP Status Code: 400

NotAuthorized

You do not have permission to perform this action.

HTTP Status Code: 400

OptInRequired

The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403

RequestExpired

The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

ServiceUnavailable

The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

ValidationError

The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400