Onboarding guide for AWS Partner Revenue Measurement

AWS Partner Revenue Measurement

AWS Partner Revenue Measurement: Onboarding guide for AWS Partner Revenue Measurement

Table of Contents

What is Partner Revenue Measurement?

Partner Revenue Measurement is AWS Partner Network's capability to measure and quantify the impact AWS partners have on overall AWS revenue. Partner Revenue Measurement empowers AWS Partners to demonstrate their revenue impact through automated and streamlined capabilities. Partner Revenue Measurement enables partners to understand customer utilization of their solutions and helps AWS improve support for feature requests and Go-to-Market funding.

Getting started with Partner Revenue Measurement

Partner Revenue Measurement measures AWS service consumption driven by partner products. This enables AWS to attribute revenue to partner solutions and provide aggregated consumption data back to partners.

Partner Revenue Measurement supports three implementation methods:

- AWS Marketplace Metering — Zero-touch revenue attribution for Amazon Machine Image (AMI) and Machine Learning (ML) products listed on AWS Marketplace
- Resource Tagging — Tag AWS resources with your product code for revenue attribution
- User Agent String — Include a User Agent string in regular AWS API/CLI calls for revenue attribution

To implement Partner Revenue Measurement, consider the following requirements:

- What AWS services does your product use (Amazon EC2, Amazon S3, Amazon ECS, Amazon RDS)?
- Do you have a SaaS, AMI, ML, or Professional Services product listed on AWS Marketplace?
- Which architecture pattern does your solution follow (Partner account, Customer account, or Hybrid)?

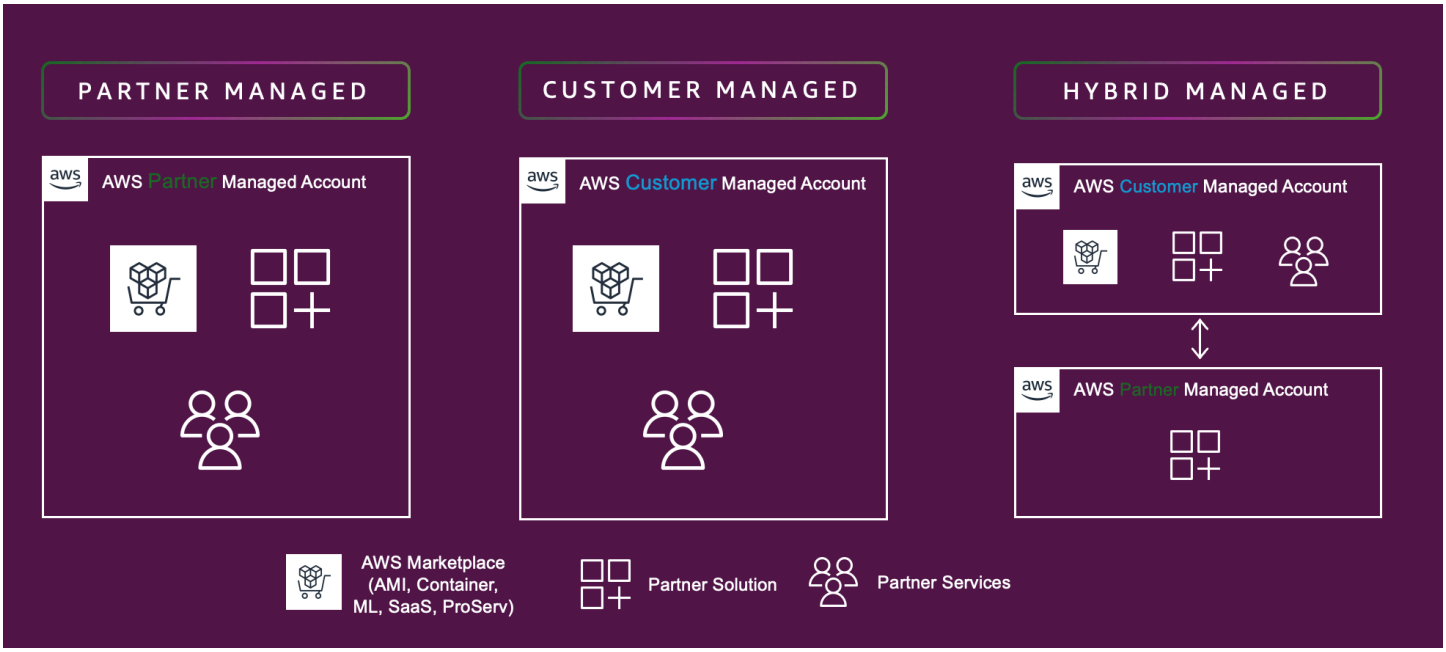
> **Note**
>
> Partner Revenue Measurement requires implementation of one or more methods to enable revenue attribution for supported AWS services.

Partner Revenue Measurement Architecture Patterns

Pattern #1 - Partner Account: All components reside within the partner's AWS account or VPC.

Pattern #2 - Customer Account: All components are deployed in the customer's AWS account and VPC.

Pattern #3 - Hybrid: Components are distributed across both partner and customer AWS accounts and VPCs.



Implementation Steps

Partner Revenue Measurement Implementation Process

1. **Step 1: Complete Prerequisites**

   Review Prerequisites

2. **Step 2: Retrieve Product Code**

   Retrieve your product code from AWS Marketplace Management Portal

3. **Step 3: Choose Implementation Method**

   Select one or more implementation methods based on your product type and architecture

4. **Step 4: Complete Method-Specific Requirements and Implement**

   Review the method-specific requirements for your chosen method, then follow the implementation guide. For a low-effort approach at scale, consider using automation where available.

Prerequisites

Before implementing Partner Revenue Measurement, you must have:

1. **AWS Partner Central and AWS account linking**

   Before linking accounts, AWS Partners must understand that this linked AWS account will become the primary account for managing all APN activities in Partner Central. This account will:

   - Determine Partner Revenue Measurement compliance for APN funding benefits eligibility
   - Serve as the primary account for the migration to the new Partner Central experience and all Partner Central users will need to be provisioned access
   - Be billed for the annual APN membership fee

   When determining which AWS account to link, consider these options:

   Option A. Partners with one AWS Marketplace account

   Evaluate your existing AWS Marketplace account against the account selection guidance criteria (Partner Central login required). Link your AWS Marketplace account if you're comfortable with:

   - Provisioning Partner Central user access to this account
   - Using this account as your primary account for APN engagements
   - Being billed the APN membership fee on this account

   If you prefer not to designate your AWS Marketplace account as your primary account, create a new AWS account or link an existing non-AWS Marketplace account that meets the above criteria.

   Option B. Partners with multiple AWS Marketplace accounts

   Evaluate your AWS Marketplace accounts against the account selection guidance criteria (Partner Central login required).

   > **Note**
   >
   > AWS recommends creating and linking a new AWS account to represent your global business, then connecting all individual AWS Marketplace accounts to your primary account using Subsidiary Account Connections (accessible only after migrating to the new Partner Central experience).

Complete these three steps:

a. Create and link an AWS account (which will become your primary account) to your Partner Central account

b. Migrate to the new Partner Central experience (Partner Central login required)

c. Link all AWS Marketplace accounts to your primary AWS account through Subsidiary Account Connections (Partner Central login required)

2. **Product listing on AWS Marketplace (Public)**

   See the AWS Marketplace Seller Guide for general information. For product-specific listing guide instructions:

Product Type	Listing Guide
SaaS	SaaS Product Listing Guide
Amazon Machine Image (AMI)	AMI Product Listing Guide
Machine Learning (ML)	ML Product Listing Guide
Professional Services	Professional Services Listing Guide

3. **Product that uses one or more of the supported AWS services**
4. **Ensure that Cost Explorer is enabled**

   To learn how to enable Cost Explorer, see Enabling Cost Explorer.

Product Code Retrieval

Your AWS Marketplace product code is required for all implementation methods. This code uniquely identifies your product for revenue attribution.

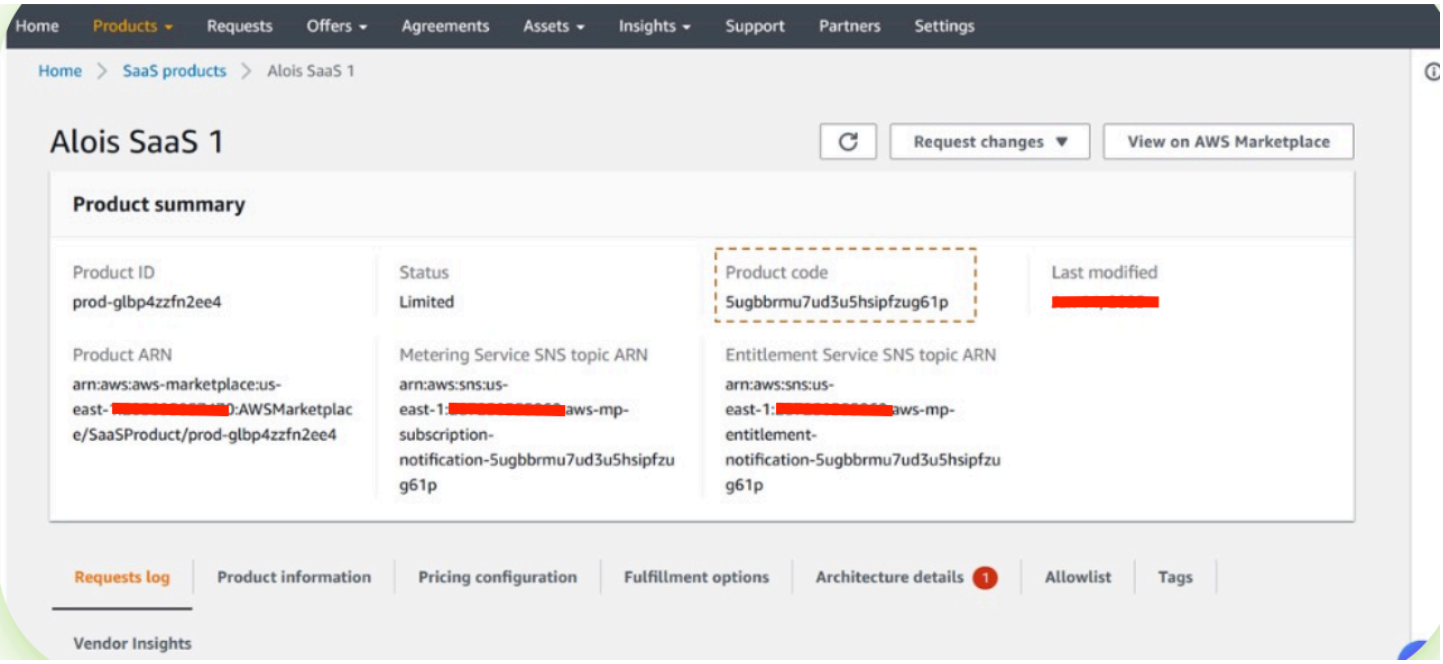
To retrieve your product code

1. Open the AWS Marketplace Management Portal, and then sign in to your seller account

2. Navigate to **Products** page
3. Select your product
4. Find the product code in the **Product Summary** section

Example product code format: `5ugbbrmu7ud3u5hsipfzug61p`

The following screenshot shows where to find the product code in the AWS Marketplace Management Portal:



Implementation Methods

Partner Revenue Measurement supports the following implementation methods. Choose the method that best fits your product type and architecture.

Methods Overview

Method	Description	Best For
AWS Marketplace Metering	Zero-touch revenue attribution through AWS Marketplace product metadata	Amazon Machine Image (AMI) and Machine Learning

Method	Description	Best For
		(ML) products listed on AWS Marketplace
Resource Tagging	Tag AWS resources with your product code using the `aws-apn-id` tag key	SaaS, Professional Services, and any product type where you can tag AWS resources
User Agent String	Include a User Agent string in regular AWS API/CLI calls with your product code	Products with direct regular AWS API/CLI access; AMI/ML products seeking attribution beyond EC2/SageMaker

Note

You can implement multiple methods simultaneously. For example, an AMI product can benefit from AWS Marketplace Metering for EC2 attribution and User Agent String for attribution on additional AWS services used by the product.

Choosing the Right Method

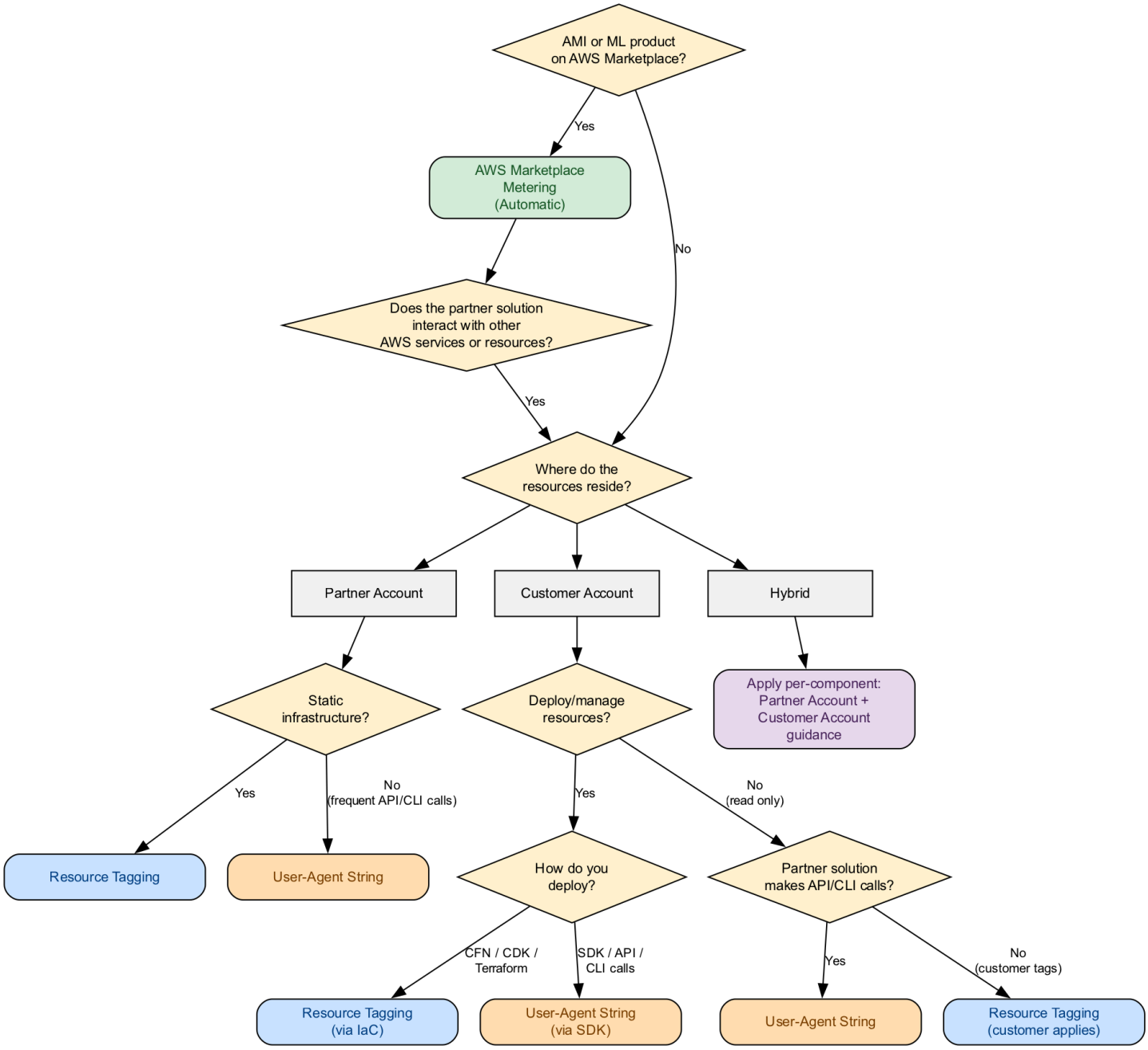
The following table provides guidance on choosing the right method based on your scenario:

Scenario	Use Resource Tagging when...	Use User Agent String when...	Marketplace Metering
Marketplace AMI/ML	—	—	Automatic
Partner Account (SaaS)	Static infrastructure, deploy once and runs long-term	Partner solution makes frequent regular AWS API/CLI calls	—
Customer Account (deploy/manage)	Deploy via IaC (CloudFormation, Terraform), customer allows tags	Partner solution makes ongoing regular AWS API/	—

Scenario	Use Resource Tagging when...	Use User Agent String when...	Marketplace Metering
		CLI calls, or customer has strict tag policies	
Customer Account (read/access only)	Customer willing to tag their resources	Your solution makes read API calls	—
Hybrid (Partner + Customer)	Taggable resources in either account	Partner solution makes regular AWS API/CLI calls in either account	—

Decision Tree

The following decision tree helps you determine the right implementation method:



Special Scenarios

Scenario	Use Resource Tagging when...	Use User Agent String when...	Marketplace Metering
Managed Services Provider (MSP) or	Where write access allows. Provide IaC templates	In management regular AWS API/CLI calls. Modify	—

Scenario	Use Resource Tagging when...	Use User Agent String when...	Marketplace Metering
Systems Integrator (SI/GSI) with limited access	with tags embedded for customer to run	agents or installers to include UA strings	
Multi-partner on same AWS resource	Single `aws-apn-id` tag limit per resource — may conflict with other partners	Each partner uses own UA string in their respective regular AWS API/CLI calls, no conflicts	—
Backfill existing resources	Bulk tag via Tag Editor or scripted bulk tagging for large-scale backfill. Revenue attribution will only be moving forward from the time tags are applied	Add UA to existing regular AWS API/CLI calls — often the lowest-effort path. Revenue attribution will only be moving forward from the time User Agent strings are implemented	—

Key Considerations

- **Resource Tagging:** Tags are key-value pairs applied to a resource to hold metadata about that resource, in this case a partner identifier. The tag type is a user-defined tag, and can be managed (updated or removed), and counts against the 50-tag-per-resource limit. Only one partner identifier is allowed per resource. Attribution is continuous while the tag is present.
- **User Agent String:** Ephemeral, specific to the regular AWS API/CLI operation carried out on a specific AWS resource, and is read-only. Allows multiple partners to independently use their identifier within their respective regular AWS API/CLI calls. Visible in CloudTrail, enabling customers to govern partner-solution interactions within their AWS resources. Requires at least one regular AWS API/CLI call per resource per month for attribution.
- **Multi-partner:** User Agent String avoids the single `aws-apn-id` tag-per-resource limitation when multiple partners operate on the same AWS resource.
- **Production only:** Partner Revenue Measurement is intended to measure production workloads. Dev/test/staging environments can be used for validating your implementation before rolling out to production.

AWS Marketplace Metering

Partner Revenue Measurement integrates with AWS Marketplace Metering to provide revenue attribution for Amazon Machine Image (AMI) and Machine Learning (ML) products listed on AWS Marketplace.

Method-Specific Requirements

In addition to the general prerequisites, ensure you meet the following requirements for AWS Marketplace Metering:

- Linked AWS Partner Central and AWS account ID. Refer to the AWS Partner Central and AWS account linking - How to Get Started guide (Requires AWS Partner Central login)
- Product listing on AWS Marketplace (Public). See the AWS Marketplace Seller Guide for general information. For product-specific listing guide instructions:

Product Type	Listing Guide
Amazon Machine Image (AMI)	AMI Product Listing Guide
Machine Learning (ML)	ML Product Listing Guide

- Product is one of the following AWS Marketplace product types only: Amazon Machine Image (AMI) or Machine Learning (ML)
- Product needs to be purchased and used by buyers on AWS Marketplace

Implementation

Partner Revenue Measurement integrates with AWS Marketplace Metering to measure service usage and attribute the consumption to your product, without requiring any additional inputs such as tagging resources or implementing user agents. This is a zero-touch experience that leverages existing product metadata for revenue attribution.

To avail attribution via this method, you need to list your AMI or ML product on AWS Marketplace and ensure customers purchase and use it via AWS Marketplace.

Amazon Machine Image (AMI) products

Revenue attribution for AMI products works as follows:

1. When you list an AMI product on AWS Marketplace, a unique product code is assigned to it.
2. This product code is automatically attached as instance metadata to every Amazon EC2 instance launched from your AMI.
3. When a buyer purchases and starts using your AMI product, Partner Revenue Measurement uses the product code to measure Amazon EC2 usage and attribute the consumption to your product.

Refer to Understanding AMI-based products and AMI product codes for more details.

Machine Learning (ML) products

When a buyer purchases, deploys, and starts consuming an ML product, AWS Marketplace can measure the usage of the product to bill the customer. Partner Revenue Measurement integrates with AWS Marketplace Metering to measure the customers' usage of Amazon SageMaker AI resources created by your product and automatically attributes the consumption to your product.

Included AWS Services

Partner Revenue Measurement automatically measures service consumption for Amazon Machine Image (AMI) and Machine Learning (ML) products listed on AWS Marketplace. The following services are supported:

AWS Marketplace Metering Included Services

Service Name	Product Service Code	Notes
Amazon EC2	AmazonEC2	Amazon Machine Image (AMI) products
Amazon SageMaker AI	AmazonSageMaker	Machine Learning (ML) products

Note

Partner Revenue Measurement intends to support all AWS services. We recommend that you instrument PRM on all AWS services and resources that your partner solution interacts with to avoid on-going operational changes as service coverage expands. At this time, revenue attribution data is surfaced for the services listed above. Any partial spend captured on services not listed above is aggregated as "Other".

Resource Tagging

Partner product resource tagging enables AWS Partners to demonstrate how their products drive service consumption and AWS revenue by tagging resources with partner product identifiers.

This self-service capability allows partners to tag resources in their own account or customer's account to measure AWS revenue driven by the partner's product.

Method-Specific Requirements

In addition to the general prerequisites, ensure you meet the following requirements for resource tagging:

- Ability to tag AWS resources in your own account or in customer's account
- Product that uses one or more of the supported AWS services for resource tagging

Manual Implementation

> **Note**
>
> Partner Revenue Measurement is intended to measure production workloads. Dev/test/staging environments can be used for validating your implementation before rolling out to production.

3-Step Implementation Process

1. Get your product code from AWS Marketplace Management Portal
2. Add the tag to your resources:
   - Tag Key: `aws-apn-id`
   - Tag Value: `pc:<product-code>`
   - Example: `pc:5ugbbrmu7ud3u5hsipfzug61p`
3. Apply via automated or manual methods listed below

> **Note**
>
> Ensure the `aws-apn-id` tag fits within the 50-tag-per-resource limit and does not conflict with existing customer tag policies.

Tagging via AWS Management Console

You can manually tag your resources using the AWS Management Console.

To get started

1. Go to your AWS Management Console.
2. Go to the resources you want to tag. Example: Amazon RDS.
3. Choose **Add tags**.
4. Enter `aws-apn-id` as the **Tag key**.
5. Enter `pc:5ugbbrmu7ud3u5hsipfzug61p` as the **Tag value** (replace `5ugbbrmu7ud3u5hsipfzug61p` with your product code).
6. Choose **Save**.

Repeat the steps above for all associated resources such as Snapshots. For more information about tagging resources, see the Tag your Amazon EC2 resources in the Amazon Elastic Compute Cloud user guide for Linux instances.

> **Important**
>
> If the resource is managed by infrastructure as code (CloudFormation, Terraform, CDK), adding tags via the console causes drift detection on the next IaC run. For IaC-managed resources, always apply tags through the IaC tool instead.

Tagging via AWS CLI

You can tag specific resources using the AWS CLI.

Note

Replace `5ugbbrmu7ud3u5hsipfzug61p` with your product code in the following example.

```
aws ec2 create-tags --resources i-1234567890abcdef0 \
  --tags Key=aws-apn-id,Value=pc:5ugbbrmu7ud3u5hsipfzug61p
```

Important

If the resource is managed by infrastructure as code (CloudFormation, Terraform, CDK), adding tags via CLI causes drift detection on the next IaC run. For IaC-managed resources, always apply tags through the IaC tool instead.

Best Practices

- Always use lowercase 'aws-apn-id' key
- Ensure 'pc:' prefix in value
- Validate product code format
- Document all tagged resources
- Monitor tag compliance and revenue attribution

Note

Once a resource is tagged with a partner's product code, AWS continues to attribute revenue to the product until the tag is removed or the resource is shut down.

Tag Management

Tag Conflicts: Since an AWS resource can only have one tag with the `aws-apn-id` key, only one partner identifier is allowed per resource. For multi-partner scenarios, consider using the User Agent String method instead. If you must use resource tagging, coordinate with the other partner and the customer to determine tag ownership before making changes.

Tag Removal: Any user with account access can remove tags. Both customers and partners (with account access) can remove tags.

Automated Tagging

We encourage you to automate your tagging resources as much as possible. You can automate your tagging by adding a tag line to your AWS CloudFormation template, AWS Cloud Development Kit (AWS CDK), Terraform, or by using the AWS Tag Editor.

> **Warning**
>
> Only tag resources that are directly used or influenced by your partner solution.

AWS CloudFormation

Stack-level propagation (recommended): Tags applied at the stack level propagate automatically to all resources in the stack that support tagging. This is the simplest approach when your entire stack belongs to your partner solution.

> **Note**
>
> Replace `5ugbbrmu7ud3u5hsipfzug61p` with your product code in the following examples.

```
aws cloudformation create-stack --stack-name my-stack \
  --template-body file://template.yaml \
  --tags Key=aws-apn-id,Value=pc:5ugbbrmu7ud3u5hsipfzug61p
```

> **Note**
>
> Most common resource types (EC2, S3, RDS, Lambda, ECS, etc.) support stack-level tag propagation. However, some resources do not — for example, EBS volumes created from block device mappings. Use per-resource tagging as a fallback for those specific resources.

Per-resource tagging: Add the tag directly to individual resource definitions when you need to tag only specific resources within a stack, or when different resources need different attribution.

```
Resources:
  MyResource:
    Type: 'AWS::EC2::Instance'
    Properties:
      Tags:
        - Key: "aws-apn-id"
          Value: "pc:5ugbbrmu7ud3u5hsipfzug61p"
```

For information about which resources you can tag with CloudFormation, see the AWS resources and property types reference in the AWS CloudFormation user guide.

AWS Cloud Development Kit (AWS CDK)

A tag in AWS CDK is applied to a given construct and all of its taggable children. For more information, see Tagging in the AWS CDK v2 developer guide.

Stack-level propagation (recommended): Tag the stack to propagate to all resources.

> **Note**
>
> Replace `5ugbbrmu7ud3u5hsipfzug61p` with your product code in the following examples.

```
Tags.of(stack).add('aws-apn-id', 'pc:5ugbbrmu7ud3u5hsipfzug61p');
```

Per-resource tagging:

```
{
    "Key" : "aws-apn-id",
    "Value" : "pc:5ugbbrmu7ud3u5hsipfzug61p"
}
```

Terraform

Provider-level default tags (recommended): The `default_tags` block applies the tag to every resource Terraform creates or manages through that provider.

Note

Replace `5ugbbrmu7ud3u5hsipfzug61p` with your product code in the following examples.

```
provider "aws" {
  default_tags {
    tags = {
      aws-apn-id = "pc:5ugbbrmu7ud3u5hsipfzug61p"
    }
  }
}
```

Note

Some Terraform resource types do not support `default_tags` (e.g., `aws_autoscaling_group`). If you also define the same tag key in a resource's `tags` block, Terraform raises a conflict. Use either `default_tags` or per-resource tags for the `aws-apn-id` tag, not both on the same resource.

Per-resource tagging: Add the tag to individual resource blocks when you need to tag only specific resources.

```
resource "aws_instance" "example" {
  tags = {
    aws-apn-id = "pc:5ugbbrmu7ud3u5hsipfzug61p"
  }
}
```

AWS Tag Editor

You can use the AWS Resource Groups and Tag Editor to tag resources in bulk using the console.

Important

For resources provisioned by infrastructure as code templates (CloudFormation, CDK, Terraform, etc.), it is recommended to update the templates instead of using Tag Editor.

Note

The AWS Tag Editor only works for resources running in the account.

To get started

1. Open the AWS Management Console.
2. Go to the Resource Groups & Tag Editor, Tagging, Tag Editor page.
3. Specify the Region(s) your resources are located. Example: us-east-1.
4. Choose the type of resources you want to bulk tag. Example: EC2, Lambda, and S3.
5. Choose **Search resources** to view the resources that meet the conditions you have selected.
6. Select all or a few of the listed resources you want to tag.
7. Choose **Manage tags of selected resources**.
8. Enter `aws-apn-id` in the **Tag Key** field.
9. Enter `pc:5ugbbrmu7ud3u5hsipfzug61p` in the **Tag Value** field (replace `5ugbbrmu7ud3u5hsipfzug61p` with your product code).
10. Choose **Review**.
11. Choose **Apply tag changes**.

AWS CLI

You can use the AWS CLI to tag resources in bulk using the command line.

Important

For resources provisioned by infrastructure as code templates (CloudFormation, CDK, Terraform, etc.), it is recommended to update the templates instead of using CLI commands.

Note

Replace 5ugbbrmu7ud3u5hsipfzug61p with your product code in the following example.

```
aws resourcegroupstaggingapi tag-resources \
    --resource-arn-list arn:aws:ec2:region:account-id:instance/i-1234567890abcdef0 \
    --tags aws-apn-id=pc:5ugbbrmu7ud3u5hsipfzug61p
```

API/SDK Bulk Tagging

Use the Resource Groups Tagging API to tag multiple resources across different service types in a single call. This is the most efficient approach for large-scale tagging.

Note

Replace 5ugbbrmu7ud3u5hsipfzug61p with your product code in the following example.

```
import boto3

tagging = boto3.client('resourcegroupstaggingapi')
tagging.tag_resources(
    ResourceARNList=[
        'arn:aws:ec2:us-east-1:123456789012:instance/i-1234567890abcdef0',
        'arn:aws:s3:::my-bucket'
    ],
    Tags={'aws-apn-id': 'pc:5ugbbrmu7ud3u5hsipfzug61p'}
)
```

Important

For resources managed by infrastructure as code (CloudFormation, CDK, Terraform), always apply tags through the IaC tool instead. SDK/CLI tagging of IaC-managed resources causes drift detection on the next IaC run.

Included AWS Services

The following AWS services are supported for resource tagging implementation. Resources must be tagged with key `aws-apn-id` and value `pc:product-code` format.

For a downloadable version of this list, download the Resource Tagging Included Services (CSV).

Resource Tagging Included Services

Service Name	Product Service Code	Notes
Amazon API Gateway	AmazonApiGateway	None
Amazon AppStream	AmazonAppStream	Excludes User Fees
AWS AppSync	AWSAppSync	None
Amazon Athena	AmazonAthena	None
Aurora DSQL	AuroraDSQL	None
AWS Backup	AWSBackup	None
Amazon Bedrock	AmazonBedrock	Excludes spend from AWS Marketplace Private Offers
Amazon Bedrock AgentCore	AmazonBedrockAgentCore	Includes AgentCore RunTime, Browser Tool, Code Interpreter, Gateway, Identity, and Memory
AWS Certificate Manager	AWSCertificateManager	Includes AWS Certificate Manager Private CA
Amazon Cloud Directory	AmazonCloudDirectory	None
AWS Cloud WAN	AWSCloudWAN	Excludes Core Network Edge Hours and data transfer

Service Name	Product Service Code	Notes
Amazon CloudFront	AmazonCloudFront	Excludes data transfer costs and Lambda@Edge
AWS CloudHSM	CloudHSM	None
Amazon CloudWatch	AmazonCloudWatch	Logs only
AWS CodeBuild	CodeBuild	None
AWS CodePipeline	AWSCodePipeline	None
AWS CodeStar	AWSCodeStar	None
Amazon Cognito	AmazonCognito	Excludes Amazon Cognito add-ons
Amazon Comprehend	comprehend	None
Amazon Connect	AmazonConnect	Includes Full Connect Unlimited AI and A La Carte; excludes Cases, Entity Resolution, Legacy Pinpoint Engagement, Meetings SDK, ContactLens, Chat, Email, Lex, Q in Connect, Tasks, Voice, Customer Profiles, Outbound Campaigns Processing, Telephony
AWS Data Pipeline	datapipeline	None
AWS Database Migration Service	AWSDatabaseMigrationSvc	None
AWS DataSync	AWSDataSync	None
AWS Deadline Cloud	AWSDeadlineCloud	Excludes AWS data transfer charges and Bring-Your-Own-License third-party creative tool software license costs

Service Name	Product Service Code	Notes
AWS Direct Connect	AWSDirectConnect	Excludes AWS Local Zones
AWS Directory Service	AWSDirectoryService	None
Amazon DocumentDB (with MongoDB compatibility)	AmazonDocDB	None
Amazon DynamoDB	AmazonDynamoDB	Excludes DAX
Amazon EC2	AmazonEC2	Includes Amazon EBS, EBS Snapshots, EC2 Mac, AWS Local Zones deployment; excludes Capacity Block for ML
Amazon ECR	AmazonECR	None
Amazon ECS	AmazonECS	Includes Fargate and AWS App Mesh
Amazon EKS	AmazonEKS	Excludes Fargate; includes AWS App Mesh
AWS Elastic Beanstalk	AWSElasticBeanstalk	None
AWS Elastic Disaster Recovery (DRS)	AWSElasticDisaster Recovery	None
Amazon Elastic File System	AmazonEFS	None
Amazon ElastiCache	AmazonElastiCache	None
AWS Elemental MediaConvert	AWSElemen talMediaConvert	None
AWS Elemental MediaLive	AWSElementalMediaL ive	None
AWS Elemental MediaPackage	AWSElemen talMediaPackage	None

Service Name	Product Service Code	Notes
Amazon EMR	ElasticMapReduce	Includes AWS Local Zones deployment
Amazon FinSpace	AmazonFinSpace	Excludes any kdb Insights software license amount
Amazon FSx	AmazonFSx	None
Amazon GameLift	AmazonGameLift	Excludes GameLift Anywhere, FleetIQ, and FlexMatch when using either EC2 for GameLift or On-premises for GameLift
AWS Glue	AWSGlue	None
AWS HealthImaging	AmazonMedicalImaging	None
AWS HealthLake	AmazonHealthLake	Excludes FHIR data export and transformation
AWS IoT Core	AWSIoT	Excludes Registry operations
AWS IoT SiteWise	AWSIoTSiteWise	Excludes Alarm, Query, Data Storage - Warm Storage, Edge - Data collection pack, Assistant - Monthly Enablement Fee, Assistant - API bundle price, Messaging - Bulk Import, Data Export - Bulk Import
Amazon Kendra	AmazonKendra	None
AWS Key Management Service	awskms	Excludes cross-account request costs
Amazon Keyspaces (for Apache Cassandra)	AmazonMCS	None

Service Name	Product Service Code	Notes
Amazon Kinesis Data Analytics	AmazonKinesisAnalytics	None
Amazon Kinesis Data Firehose	AmazonKinesisFirehose	None
Amazon Kinesis Data Streams	AmazonKinesis	None
Amazon Kinesis Video Streams	AmazonKinesisVideo	None
AWS Lambda	AWSLambda	None
Elastic Load Balancing	AWSELB	None
AWS Mainframe Modernization	AWSM2	Excludes 'M2 Custom' and Blu Age Transformation Center costs
Amazon MemoryDB for Redis	AmazonMemoryDB	Excludes Snapshot Storage
Amazon MQ	AmazonMQ	None
Amazon MSK	AmazonMSK	None
Amazon Neptune	AmazonNeptune	None
AWS Network Firewall	AWSNetworkFirewall	None
Amazon Omics	AmazonOmics	None
Amazon OpenSearch Service	AmazonES	Includes Amazon Elasticsearch Service; excludes OpenSearch Serverless and OpenSearch Ingestion

Service Name	Product Service Code	Notes
AWS Payment Cryptography	PaymentCryptography	Excludes following APIs under Requests usage: ListAliases, ListKeys, GetResourcePolicy, GetParametersForImport, GetParametersForExport, DeleteResourcePolicy, DeleteAlias, CreateKey, CreateAlias
Amazon QuickSight	AmazonQuickSight	Excludes Region fee for Q, SPICE, unused charges on subscription packs, Pro user, Pro author, and administrator
Amazon Redshift	AmazonRedshift	Amazon Redshift Provisioned and Amazon Redshift Serverless
Amazon Relational Database Service (RDS)	AmazonRDS	Includes all RDS engines, Amazon RDS Custom, AWS Local Zones deployment; excludes Db2 licensing fees
AWS Resilience Hub	AWSResilienceHub	None
Amazon Route 53	AmazonRoute53	Excludes Amazon Route 53 Resolver, Traffic Flow, and CIDR block storage
Amazon S3	AmazonS3	Includes storage cost only and all storage tiers; excludes Requests
Amazon S3 Glacier	AmazonGlacier	Excludes Glacier Deep Archive
Amazon SageMaker AI	AmazonSageMaker	Excludes Amazon SageMaker AI training plans for training jobs or HyperPod clusters
AWS Secrets Manager	AWSSecretsManager	None
AWS Security Hub	AWSSecurityHub	None

Service Name	Product Service Code	Notes
Amazon Simple Notification Service (SNS)	AmazonSNS	None
Amazon Simple Queue Service (SQS)	AWSQueueService	None
AWS Step Functions	AmazonStates	None
AWS Storage Gateway	AWSStorageGateway	None
AWS Systems Manager	AWSSystemsManager	OpsCenter only
Amazon Timestream	AmazonTimestream	None
AWS Transfer Family	AWSTransfer	Excludes AWSDataTransfer
AWS Transit Gateway	AmazonVPC	Includes Transit Gateway VPN, VPC, Peering, DirectConnect, DX; excludes data transfer costs
Amazon VPC Lattice	AmazonVPC	None
Amazon WorkSpaces	AmazonWorkSpaces	Includes WorkSpaces Core; excludes third-party VDI management software license and third-party OS or software license on virtual desktop

Note

Revenue attribution continues until the tag is removed or the resource is terminated.

Note

Partner Revenue Measurement intends to support all AWS services. We recommend that you instrument PRM on all AWS services and resources that your partner solution

interacts with to avoid on-going operational changes as service coverage expands. At this time, revenue attribution data is surfaced for the services listed above. Any partial spend captured on services not listed above is aggregated as "Other".

User Agent String

This guide provides step-by-step instructions for AWS Partners to implement User Agent strings for Partner Revenue Measurement revenue attribution using AWS APIs and SDKs.

> **Note**
>
> If you have an AMI product listed on AWS Marketplace that uses services beyond Amazon EC2, or an ML product that uses services beyond Amazon SageMaker AI, AWS recommends integrating a User Agent string to receive additional revenue attribution for your product.

> **Note**
>
> **Required Format:** `APN_1.1/pc_<YOUR-PRODUCT-CODE>$`
> The User Agent string format must be included in all regular AWS API/CLI calls made by your solution. Where `<YOUR-PRODUCT-CODE>` is your alphanumeric product code from AWS Marketplace and $ is the required end delimiter.

> **Note**
>
> For revenue attribution, your product must conduct at least one API operation on an AWS Resource Name (ARN) per month. If no API operations are performed on a resource in a given month, that resource does not contribute to revenue attribution for that month.

Method-Specific Requirements

In addition to the general prerequisites, ensure you meet the following requirements for User Agent string implementation:

- Product listing on AWS Marketplace (Public). See the AWS Marketplace Seller Guide for general information. For product-specific listing guide instructions:

Product Type	Listing Guide
Amazon Machine Image (AMI)	AMI Product Listing Guide
SaaS	SaaS Product Listing Guide
ML	ML Product Listing Guide
Professional Services	Professional Services Listing Guide

- Product that uses one or more of the supported AWS services for User Agent string
- Direct regular AWS API/CLI access from your application

Manual Implementation

Follow these steps to implement User Agent strings in your regular AWS API/CLI calls for Partner Revenue Measurement:

1. Retrieve your product code from AWS Marketplace Management Portal.
2. Create the User Agent string using the format `APN_1.1/pc_<YOUR-PRODUCT-CODE>$`. Replace `<YOUR-PRODUCT-CODE>` with your actual alphanumeric product code. The $ symbol marks the end delimiter.
3. Update your AWS SDK configuration to include the User Agent string. For complete implementation examples across different programming languages, see User Agent Implementation Samples. For automated methods, see Automated User Agent.

   Basic example for AWS CLI:

```
# Set user-agent string for AWS CLI
export AWS_SDK_UA_APP_ID="APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$"

# Example EC2 API call with user-agent
aws ec2 run-instances --image-id ami-0abcdef1234567890 --instance-type t2.micro --
region us-east-1
```

> **Note**
>
> Replace `5ugbbrmu7ud3u5hsipfzug61p` with your actual AWS Marketplace product code.

4. Test your implementation using AWS CloudTrail to verify the User Agent string is included:

```
# Check CloudTrail logs for user-agent string
aws logs filter-log-events \
  --log-group-name CloudTrail/YourLogGroup \
  --filter-pattern "APN_1.1" \
  --start-time 1640995200000
```

   Look for the `userAgent` field in CloudTrail events to confirm your User Agent string is being sent with API calls.

5. Deploy to production and contact your AWS partner management team or APN Support (Partner Central login required) with your deployment details for validation.

   Example CloudTrail event showing User Agent string:

```
{
  "eventTime": "2025-01-14T21:21:54Z",
  "eventName": "RunInstances",
  "eventSource": "ec2.amazonaws.com",
  "userAgent": "APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$ aws-cli/2.0.0",
  "sourceIPAddress": "203.0.113.12",
  "resources": [
    {
      "accountId": "123456789123",
      "type": "AWS::EC2::Instance"
    }
  ]
}
```

> **Note**
>
> User Agent strings must be implemented consistently across all regular AWS API/CLI calls to ensure accurate revenue attribution. For multi-region deployments, ensure the User

Agent string is included in regular AWS API/CLI calls across all regions where your solution operates.

Note

Partner Revenue Measurement is intended to measure production workloads. Dev/test/staging environments can be used for validating your implementation before rolling out to production.

Important

User Agent based attribution is evaluated monthly. Your partner solution must make at least one regular AWS API/CLI call per resource per month for continued attribution. If no calls are made on a resource in a given month, that resource does not contribute to revenue attribution for that month.
In scenarios where your partner solution does not make frequent regular AWS API/CLI calls, you can make non-mutating, read-only calls (such as `Describe*` operations) to demonstrate continued interaction between your partner solution and the AWS resource. Refer to the included services for supported API actions.

Infrastructure as Code Limitations

AWS CloudFormation / AWS CDK: CloudFormation does not support custom User Agent strings natively. When CloudFormation creates resources, it makes regular AWS API/CLI calls using its own service principal, and you cannot control the User Agent string in those calls. Use Resource Tagging for CloudFormation-deployed resources instead. If your CloudFormation template includes Custom Resources (Lambda-backed), the Lambda function code can include a User Agent string via the SDK approach.

Terraform: Terraform supports User Agent attribution via `provider_meta` with the `user_agent` argument (AWS provider **>= 6.27.0** or AWSCC provider **>= 1.67.0**). This is scoped to the declaring module only, ensuring correct attribution without collision across multiple partner modules. See Terraform Implementation Sample for full details.

Implementation Samples

This section provides code samples for implementing User Agent strings across different programming languages and AWS SDKs. Each sample demonstrates how to configure the User Agent string format `APN_1.1/pc_<YOUR-PRODUCT-CODE>$` for AWS services.

> **Note**
>
> Replace the example product code `5ugbbrmu7ud3u5hsipfzug61p` with your actual AWS Marketplace product code retrieved from the AWS Marketplace Management Portal.

Python (boto3)

```
import boto3
from botocore.config import Config

UA_STRING = "APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$"

# Create a config object with the custom user agent
session_config = Config(user_agent=UA_STRING)

# EC2 client
ec2 = boto3.client('ec2', config=session_config)

# S3 client
s3 = boto3.client('s3', config=session_config)
```

Java (AWS SDK v2)

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.client.config.SdkAdvancedClientOption;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.s3.S3Client;

public class AwsCustomUserAgentExample {
    private static final String UA_STRING = "APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$";
    private static final Region REGION = Region.US_WEST_2;

    public static void main(String[] args) {
```

```
        ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
            .putAdvancedOption(SdkAdvancedClientOption.USER_AGENT_PREFIX, UA_STRING)
            .build();

        Ec2Client ec2 = Ec2Client.builder()
            .region(REGION)
            .overrideConfiguration(overrideConfig)
            .build();

        S3Client s3 = S3Client.builder()
            .region(REGION)
            .overrideConfiguration(overrideConfig)
            .build();
    }
}
```

.NET (AWS SDK)

```
using Amazon;
using Amazon.S3;
using Amazon.Runtime;

class Program
{
    static void Main(string[] args)
    {
        AWSConfigs.UseAlternateUserAgentHeader = true;
        string UA_STRING = "APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$";

        var s3Config = new AmazonS3Config
        {
            RegionEndpoint = RegionEndpoint.USEast2,
        };
        var s3Client = new AmazonS3Client(s3Config);

        s3Client.BeforeRequestEvent += delegate (object sender, RequestEventArgs e)
        {
            if (e is WebServiceRequestEventArgs args)
            {
                args.Headers["User-Agent"] = UA_STRING;
            }
        };
```

```
    }
}
```

Node.js (AWS SDK v3)

```
const { EC2Client } = require("@aws-sdk/client-ec2");
const { S3Client } = require("@aws-sdk/client-s3");

const PRODUCT_CODE = "5ugbbrmu7ud3u5hsipfzug61p";
const REGION = "us-west-2";

function addPRMUserAgent(client, productCode) {
    client.middlewareStack.add(
        (next) => async (args) => {
            if (!args.request.headers) args.request.headers = {};
            const existing = args.request.headers["User-Agent"] || "";
            const prmUAString = `APN_1.1/pc_${productCode}$`;
            args.request.headers["User-Agent"] = existing
                ? `${existing} ${prmUAString}`
                : prmUAString;
            return next(args);
        },
        { step: "finalizeRequest", name: "prmUserAgent" }
    );
    return client;
}

const ec2Client = addPRMUserAgent(new EC2Client({ region: REGION }), PRODUCT_CODE);
const s3Client = addPRMUserAgent(new S3Client({ region: REGION }), PRODUCT_CODE);
```

Go (AWS SDK v2)

```
package main

import (
    "context"
    "net/http"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)
```

```
const UA_STRING = "APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$"

type PRMUserAgentTransport struct {
    Transport http.RoundTripper
}

func (t *PRMUserAgentTransport) RoundTrip(req *http.Request) (*http.Response, error) {
    req.Header.Set("User-Agent", UA_STRING)
    return t.Transport.RoundTrip(req)
}

func main() {
    cfg, _ := config.LoadDefaultConfig(context.TODO(),
        config.WithRegion("us-east-2"),
        config.WithHTTPClient(&http.Client{
            Transport: &PRMUserAgentTransport{
                Transport: http.DefaultTransport,
            },
        }),
    )
    s3Client := s3.NewFromConfig(cfg)
    _ = s3Client
}
```

Ruby (AWS SDK v3)

```
require 'aws-sdk-ec2'
require 'aws-sdk-s3'

UA_STRING = 'APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$'
REGION = 'us-west-2'

Aws.config.update({
  region: REGION,
  user_agent_suffix: UA_STRING
})

ec2_client = Aws::EC2::Client.new
s3_client = Aws::S3::Client.new
```

Terraform

The Terraform AWS provider supports three ways to inject custom User Agent information:

Method	Scope	Recommended
`provider_meta` `user_agent` argument	Declaring module only	Yes
`user_agent` provider argument	Provider block	No
`TF_APPEND_USER_AGENT` env var	Global (all API calls)	No

`provider_meta` is scoped to the declaring module only, ensuring correct attribution without collision when multiple partner modules are used in the same Terraform configuration.

Prerequisites: Terraform >= 1.0 and AWS provider **>= 6.27.0** (or AWSCC provider **>= 1.67.0**).

In your module's `terraform` block, add the `provider_meta "aws"` block with the PRM User Agent string:

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 6.27.0"
    }
  }

  provider_meta "aws" {
    user_agent = [
      "APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$",
    ]
  }
}
```

Replace `5ugbbrmu7ud3u5hsipfzug61p` with your actual product code. The `$` is a required end delimiter, do not omit it.

Important

User Agent attribution requires ongoing API or CLI interaction with AWS resources. Terraform typically interacts with resources only during `plan`, `apply`, and `destroy` operations. If the partner solution provisions static, long-running resources with limited ongoing API activity, consider using tag-based attribution with Terraform instead.

Note

Do not declare a `provider "aws" {}` block in your module. The provider configuration should be controlled by the root module (the customer). Your module should only use `provider_meta`.
`provider_meta` user-agent does **not** inherit to child modules. If your module calls other modules that also need attribution, each module must declare its own `provider_meta`.
If both provider-level `user_agent` and `provider_meta` are present, the provider-level User Agent appears first in the header, followed by `provider_meta`.

Automated User Agent

For partner solutions that make multiple regular AWS API/CLI calls, you can automate User Agent string inclusion using AWS SDK Application ID settings instead of instrumenting every call individually.

Warning

Only configure User Agent strings for regular AWS API/CLI calls that are directly made by your partner solution. Do not configure User Agent strings for customer-initiated calls that are independent of your solution.

Note

If an Application ID is already configured, you can append your Partner Revenue Measurement User Agent string using a space as a delimiter. For example:
`EXISTING_APP_ID APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$`

Shared AWS Config File

Configure the User Agent string in the AWS shared configuration file (`~/.aws/config`):

> **Note**
>
> Replace `5ugbbrmu7ud3u5hsipfzug61p` with your product code. See Product Code Retrieval. The $ is the required end delimiter.

```
[default]
sdk_ua_app_id=APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$
```

Environment Variable

Set the User Agent string using environment variables for automated deployments:

> **Note**
>
> Replace `5ugbbrmu7ud3u5hsipfzug61p` with your product code. The $ is the required end delimiter.

Linux/macOS:

```
export AWS_SDK_UA_APP_ID=APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$
```

Windows:

```
setx AWS_SDK_UA_APP_ID APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$
```

Python:

```
import os
os.environ['AWS_SDK_UA_APP_ID'] = 'APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$'
```

Node.js:

```
process.env.AWS_SDK_UA_APP_ID = 'APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$';
```

JVM System Property (Java/Kotlin)

For Java and Kotlin applications, configure the User Agent string using JVM system properties:

> **Note**
>
> Replace `5ugbbrmu7ud3u5hsipfzug61p` with your product code. The `$` is the required end delimiter.

```
java -Dsdk.ua.appId="APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$" -jar your-application.jar
```

Or set it programmatically:

```
System.setProperty("sdk.ua.appId", "APN_1.1/pc_5ugbbrmu7ud3u5hsipfzug61p$");
```

Support by AWS SDKs and Tools

The Application ID configuration methods described above are supported by most AWS SDKs and tools. For a complete list, see Application ID support by AWS SDKs and tools. JVM system property settings are only supported by the AWS SDK for Java and the AWS SDK for Kotlin.

Validation

After configuring automated User Agent settings, verify the configuration:

1. Make a test AWS API call from your application
2. Check AWS CloudTrail logs to verify the User Agent string appears in the `userAgent` field
3. Confirm the format matches exactly: `APN_1.1/pc_<YOUR-PRODUCT-CODE>$`

Included AWS Services

The following AWS services are supported for User Agent string implementation. User Agent strings must be included in all regular AWS API/CLI calls to ensure proper revenue attribution.

For detailed information including specific API actions and regions supported for each service, download the User Agent Included Services and API Actions (CSV).

User Agent String Included Services

Service Name	Product Service Code	Notes
AWS Certificate Manager	AWSCertificateManager	None
Amazon CloudFront	AmazonCloudFront	None
Amazon CloudWatch	AmazonCloudWatch	Alarms and Logs
AWS CodeBuild	CodeBuild	None
AWS Direct Connect	AWSDirectConnect	None
AWS Directory Service	AWSDirectoryService	None
Amazon DynamoDB Accelerator (DAX)	AmazonDAX	None
Amazon EC2	AmazonEC2	None
Amazon ECS	AmazonECS	None
Amazon ElastiCache	AmazonElastiCache	None
Elastic Load Balancing (ELB)	AWSELB	None
Amazon EMR	ElasticMapReduce	None
Amazon EventBridge	AmazonEventBridge	None
Amazon Kinesis	AmazonKinesis	Data Streams and Data Firehose
AWS License Manager	AWSLicenseManager	None
Amazon Lightsail	AmazonLightsail	None
Amazon MemoryDB for Redis	AmazonMemoryDB	None
Amazon OpenSearch Service	AmazonES	None

Service Name	Product Service Code	Notes
Amazon RDS	AmazonRDS	None
Amazon Route 53	AmazonRoute53	None
Amazon S3	AmazonS3	None
AWS Shield	AWSShield	None
AWS WAF	awswaf	None
Amazon WorkSpaces	AmazonWorkSpaces	None

Note

Monthly API operations on resources are required for attribution to occur.

Note

Partner Revenue Measurement intends to support all AWS services. We recommend that you instrument PRM on all AWS services and resources that your partner solution interacts with to avoid on-going operational changes as service coverage expands. At this time, revenue attribution data is surfaced for the services listed above. Any partial spend captured on services not listed above is aggregated as "Other".

Automation

Partner Revenue Measurement supports automation for each implementation method. Choose the automation approach that matches your implementation method.

AWS Marketplace Metering

AWS Marketplace Metering is a fully automated, zero-touch experience for partners. When you list an AMI or ML product on AWS Marketplace and customers purchase and use it, revenue attribution occurs automatically through existing product metadata. No additional implementation is required from partners.

For more information, see AWS Marketplace Metering.

Resource Tagging Automation

Automate resource tagging using infrastructure as code tools and bulk tagging utilities. For detailed instructions, see Automated Tagging in the Resource Tagging chapter.

Supported automation tools:

- AWS CloudFormation
- AWS Cloud Development Kit (AWS CDK)
- Terraform
- AWS Tag Editor
- AWS CLI

User Agent String Automation

Automate User Agent string inclusion across all regular AWS API/CLI calls using AWS SDK Application ID configuration. For detailed instructions, see Automated User Agent in the User Agent String chapter.

Supported automation methods:

- Shared AWS config file (`~/.aws/config`)

- Environment variable (`AWS_SDK_UA_APP_ID`)
- JVM system property (Java/Kotlin)

Included AWS services

Partner Revenue Measurement supports specific AWS services for revenue attribution measurement. The supported services vary by implementation method.

- AWS Marketplace Metering included services
- Resource Tagging included services
- User Agent String included services

Best practices

Some AWS services require service-specific implementation steps for Partner Revenue Measurement. This section provides best practices for these services.

Amazon Bedrock - Resource Tagging

Amazon Bedrock uses *application inference profiles* as the taggable resource for Partner Revenue Measurement. You must create an application inference profile, tag it with the `aws-apn-id` tag, and then use that profile for all model invocations.

Understanding inference profiles

An inference profile is an Amazon Bedrock resource that specifies a foundation model and its associated AWS Regions for model invocation. Inference profiles enable cost management and resource measurement through cost allocation tags.

Amazon Bedrock offers the following types of inference profiles:

- **Cross-region (system-defined) inference profiles** – Predefined profiles that include multiple Regions to which requests for a model can be routed.
- **Application inference profiles** – User-created profiles to measure costs and model usage. You can create a profile that routes requests to one Region or to multiple Regions.

> **Warning**
>
> **System-defined inference profiles do not support tagging.** While system-defined (cross-region) inference profiles enhance flexibility in model usage by routing requests across multiple Regions, they do not support attaching custom tags for measuring, managing, and controlling costs across workloads and tenants. Only **application inference profiles** support the `aws-apn-id` tag required for Partner Revenue Measurement attribution. You must create an application inference profile to use Resource Tagging with Amazon Bedrock.

Prerequisites

- Access to supported foundation models through the Amazon Bedrock console or API.

- Your IAM role must have access to inference profile API actions. If your role has the `AmazonBedrockFullAccess` managed policy attached, you can skip this step. Otherwise, create a policy with the following permissions:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "bedrock:InvokeModel*",
        "bedrock:CreateInferenceProfile"
      ],
      "Resource": [
        "arn:aws:bedrock:*::foundation-model/*",
        "arn:aws:bedrock:*:*:inference-profile/*",
        "arn:aws:bedrock:*:*:application-inference-profile/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "bedrock:GetInferenceProfile",
        "bedrock:ListInferenceProfiles",
        "bedrock:DeleteInferenceProfile",
        "bedrock:TagResource",
        "bedrock:UntagResource",
        "bedrock:ListTagsForResource"
      ],
      "Resource": [
        "arn:aws:bedrock:*:*:inference-profile/*",
        "arn:aws:bedrock:*:*:application-inference-profile/*"
      ]
    }
  ]
}
```

Tagging an application inference profile

Follow these steps to create and tag an application inference profile using the AWS CLI.

> **Note**
>
> The tag value must use the format `pc:product-code`, where *`product-code`* is your AWS Marketplace product code. To retrieve your product code, see Product Code Retrieval. Do **NOT** use the Product ID or UUID formatted product ID.

1. **Create an application inference profile.**

```
aws bedrock create-inference-profile \
  --inference-profile-name "my-partner-inference-profile" \
  --model-source "copyFrom=arn:aws:bedrock:us-east-1:123456789012:inference-profile/
us.amazon.nova-pro-v1:0"
```

   Replace the Region, account ID, and inference profile ID with your values. The model source uses a `copyFrom` parameter that references a system-defined inference profile.

   The output returns the inference profile ARN and status:

```
{
  "inferenceProfileArn": "arn:aws:bedrock:us-east-1:123456789012:application-
inference-profile/k1c3lwu20lem",
  "status": "ACTIVE"
}
```

2. **Tag the inference profile with your partner identifier.**

```
aws bedrock tag-resource \
  --resource-arn "arn:aws:bedrock:us-east-1:123456789012:application-inference-
profile/k1c3lwu20lem" \
  --tags Key=aws-apn-id,Value=pc:5ugbbrmu7ud3u5hsipfzug61p
```

   Use the ARN from the previous step. Replace *`5ugbbrmu7ud3u5hsipfzug61p`* with your AWS Marketplace product code. For details on the tag format, see Resource Tagging.

3. **Verify the tag.**

```
aws bedrock list-tags-for-resource \
  --resource-arn "arn:aws:bedrock:us-east-1:123456789012:application-inference-
profile/k1c3lwu20lem"
```

The output confirms the tag:

```
{
  "tags": [
    {
      "key": "aws-apn-id",
      "value": "pc:5ugbbrmu7ud3u5hsipfzug61p"
    }
  ]
}
```

> **Important**
>
> All subsequent model invocations must use the tagged application inference profile ARN to ensure proper Partner Revenue Measurement attribution.

Listing application inference profiles

To list all application inference profiles in your account:

```
aws bedrock list-inference-profiles --type-equals "APPLICATION"
```

To get details for a specific profile:

```
aws bedrock get-inference-profile \
  --inference-profile-identifier "inference-profile-id"
```

Amazon EKS - Resource Tagging

Amazon EKS clusters run containerized applications on a set of nodes. Customers pay for EKS cluster hours and the underlying AWS resources including nodes (EC2 instances), load balancers, and EBS volumes. To measure revenue with Partner Revenue Measurement, you must tag both the Kubernetes cluster and its underlying AWS resources.

> **Note**
>
> The tag value must use the format `pc:product-code`, where *`product-code`* is your AWS Marketplace product code. To retrieve your product code, see Product Code Retrieval.

Tagging the Kubernetes cluster

You can add tags to new or existing Kubernetes clusters using the Amazon EKS console, `eksctl`, AWS CLI, AWS API, or infrastructure-as-code tools like Terraform.

- **New clusters** – Apply tags during cluster creation using the `tags` parameter on the `CreateCluster` API action.
- **Existing clusters** – Apply tags using the `TagResource` API, or the **Tags** tab in the Amazon EKS console.
- **Terraform** – Use the `tags` argument when creating an EKS cluster. For more information, see Terraform EKS cluster resource.

```
aws eks tag-resource \
  --resource-arn arn:aws:eks:region:account-id:cluster/cluster-name \
  --tags aws-apn-id=pc:5ugbbrmu7ud3u5hsipfzug61p
```

Tagging nodes within a node group

Amazon EKS clusters can schedule pods on any combination of self-managed nodes and EKS managed nodes. In all cases, ensure nodes are tagged with `aws-apn-id` using the format `pc:product-code`.

- **Managed nodes** – Use a custom launch template with the `TagSpecification` parameter to specify tags to apply to nodes (EC2 instances) in the node group. For example:

```
"TagSpecifications": [
  {
    "ResourceType": "instance",
    "Tags": [
      {
        "Key": "aws-apn-id",
        "Value": "pc:5ugbbrmu7ud3u5hsipfzug61p"
```

```
      }
    ]
  }
]
```

You can launch a managed node group with a custom launch template using the EKS API, AWS CLI, CloudFormation, or the EKS console. For more information, see Launch template support.

- **Self-managed nodes** – Apply tags by creating a node group with `eksctl` using the `tags` parameter. Tags are applied to all EC2 instances created as part of the node group. You can also apply tags using the AWS Management Console. For more information, see Tagging your Amazon EC2 resources.

Tagging load balancers

The AWS Load Balancer Controller manages Elastic Load Balancers for a Kubernetes cluster.

- **Application Load Balancer (ALB)** – The controller creates an ALB when you create a Kubernetes Ingress. To tag ALBs, add the following annotation to the Ingress:

```
alb.ingress.kubernetes.io/tags: aws-apn-id=pc:5ugbbrmu7ud3u5hsipfzug61p
```

For more information, see Application load balancing on Amazon EKS.

- **Network Load Balancer (NLB)** – The controller creates an NLB when you create a Kubernetes Service of type `LoadBalancer` using IP targets. To tag NLBs, add the following annotation to the Service:

```
service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags: aws-apn-
id=pc:5ugbbrmu7ud3u5hsipfzug61p
```

For more information, see Network load balancing on Amazon EKS.

Tagging EBS volumes

The Amazon EBS Container Storage Interface (CSI) driver provides a CSI interface that allows Amazon EKS clusters to manage the lifecycle of EBS volumes. To add tags to dynamically provisioned EBS volumes, use the `--extra-tags` command option in the CSI driver. For detailed instructions, see the Amazon EBS CSI driver documentation.

Fargate on EKS

Fargate on EKS is not supported for Partner Revenue Measurement resource tagging.

Isolating partner solution resources on a shared EKS cluster

If a customer runs both partner solution workloads and other workloads in the same EKS cluster, you cannot tag the cluster itself (the control plane). Instead, tag only the load balancers and the nodes running partner solution workloads. To isolate the workloads, use Kubernetes affinities and taints.

1. **Create a partner-solution-only node group.**
   - Tag this node group following the instructions in Tagging nodes within a node group.
   - Add a taint with effect `NO_SCHEDULE`.
   - Add a label (for example, `partner-solution: "true"`).
2. **Create partner solution pods.**
   - Add a toleration to the pods with a key/value matching the taint from step 1 and an effect of `NoSchedule`.
   - Add a `requiredDuringSchedulingIgnoredDuringExecution` affinity with a `matchExpression` that matches the label from step 1.

Steps 1 and 2a prevent non-partner-solution pods from deploying on partner solution nodes. Steps 1b and 2b prevent partner solution pods from deploying on non-partner-solution nodes. No changes are required to existing non-partner-solution pods or nodes.

For more information, see:

- Taints and Tolerations
- Node Affinity
- Taints on EKS managed node groups

FAQs

1. General Questions

1.1 What is Partner Revenue Measurement?

Partner Revenue Measurement is a set of capabilities that enables AWS Partners to measure the AWS service consumption driven by their solutions and quantify their impact on overall AWS revenue. These capabilities empower AWS Partners to better understand their AWS revenue impact and product consumption patterns. Partner Revenue Measurement offers three implementation options: AWS Marketplace Metering, Resource Tagging, and User Agent String.

1.2 Which AWS services are supported by Partner Revenue Measurement?

Partner Revenue Measurement supports AWS services across implementation methods. The supported services vary by method. See AWS Marketplace Metering included services, Resource Tagging included services, and User Agent String included services for complete lists.

1.3 Where do I find my AWS Marketplace product code?

Log in to **AWS Marketplace Management Portal**, navigate to your **Products** page, select your product, and find the product code in the **Product Summary** section. The product code format is typically a long alphanumeric string like: **5ugbbrmu7ud3u5hsipfzug61p**. Please do **NOT** use the Product ID or the UUID formatted product ID from the AWS Marketplace listing. For further information, refer to Retrieve your product code.

1.4 What architecture patterns does Partner Revenue Measurement support?

Partner Revenue Measurement supports three architecture patterns: 1) Partner Account - all components in partner's AWS account/VPC, 2) Customer Account - all components in customer's AWS account/VPC, 3) Hybrid - components distributed across both partner and customer accounts/VPCs.

1.5 How do I get support for Partner Revenue Measurement implementation?

Contact your AWS partner management team or APN Support (Partner Central login required) for validation assistance and support with your Partner Revenue Measurement implementation.

1.6 How should I handle if there is another tag on the AWS resource?

Since an AWS resource can only have one tag with the `aws-apn-id` key, only one partner identifier is allowed per resource. For multi-partner scenarios where multiple partners operate on the same AWS resource, consider using the User Agent String method instead. If you must use resource tagging, coordinate with the other partner and the customer to determine tag ownership before making changes.

1.7 Who can remove tags?

Any user that has access to the account can remove a tag. Both customers and partners (if they have access to the customer's account) can remove tags.

1.8 Which regions are currently supported?

Partner Revenue Measurement currently supports only Commercial regions, not European Sovereign Cloud (ESC) or US GovCloud / Amazon Dedicated Cloud (ADC).

1.9 How often does my partner solution need to make regular AWS API/CLI calls for User Agent based attribution?

Your partner solution must make at least one regular AWS API/CLI call per resource per month. Attribution is evaluated on a monthly billing cycle. If no calls are made on a resource in a given month, that resource does not contribute to revenue attribution for that month. Attribution resumes the next month a qualifying call is made. In scenarios where your partner solution does not make frequent calls, you can use non-mutating, read-only calls (such as `Describe*` operations) to demonstrate continued interaction. Refer to the included services for supported API actions.

1.10 What are my options if the customer's environment does not permit additional resource tags?

Use the User Agent String method instead. User Agent strings do not require adding user-defined tags to any AWS resource, do not consume the customer's tag quota, and do not interfere with existing tag policies. The User Agent string is captured in AWS CloudTrail logs, which also provides the customer with operational visibility into partner solution activity for auditing and operational excellence purposes.

2. Troubleshooting

For troubleshooting guidance across all implementation methods, see Troubleshooting Partner Revenue Measurement.

3. Additional FAQs

For additional FAQs, see the Partner Revenue Measurement FAQs on AWS Partner Central (login required).

Troubleshooting Partner Revenue Measurement

This section provides troubleshooting guidance for each Partner Revenue Measurement implementation method:

- AWS Marketplace Metering Issues
- Resource Tagging Issues
- User Agent String Issues

AWS Marketplace Metering Issues

AWS Marketplace Metering is a fully automated, zero-touch experience. Revenue attribution occurs automatically when customers purchase and use your AMI or ML product through AWS Marketplace. No additional implementation is required from partners.

If you are experiencing issues with revenue attribution for your AWS Marketplace Metering products, open a support ticket through APN Support (Partner Central login required). Include the following details:

- AWS account ID
- AWS Marketplace product code
- Product type (AMI or ML)
- Description of the issue and expected behavior

Resource Tagging Issues

Tags not providing revenue attribution

If resource tags are not generating revenue attribution:

Verify tag implementation

1. Check tag key is exactly: **aws-apn-id** (lowercase)
2. Verify tag value format: **pc:product-code**

3. Confirm product code matches AWS Marketplace listing (see Product Code Retrieval)
4. Ensure resources are in supported services
5. Check that resources are actively consuming AWS services and incurring spend. For example, IAM is a no-cost AWS service, so tagging IAM resources does not generate revenue attribution. Focus on tagging resources that incur charges such as EC2 instances, S3 buckets with storage, RDS databases, or Lambda functions with invocations
6. Verify tags are applied correctly using AWS Tag Editor or reach out to your AWS partner management team or APN Support (Partner Central login required) for assistance

Tag conflicts with other partners

Since an AWS resource can only have one tag with the `aws-apn-id` key, only one partner identifier is allowed per resource. If another partner's tag exists on a resource, resource tagging creates a conflict.

For multi-partner scenarios where multiple partners operate on the same AWS resource, consider using the User Agent String method instead. Each partner can independently use their own identifier within their respective regular AWS API/CLI calls without conflicts.

If you must use resource tagging, coordinate with the other partner and the customer to determine tag ownership before making changes.

User Agent String Issues

User Agent string not appearing in CloudTrail

If your User Agent string is not visible in CloudTrail logs:

Verify User Agent implementation

1. Confirm the format is exactly: `APN_1.1/pc_<YOUR-PRODUCT-CODE>$`
2. Verify the $ end delimiter is present and not stripped by your shell or runtime
3. Confirm product code matches AWS Marketplace listing (see Product Code Retrieval)
4. Ensure the User Agent string is applied to the correct AWS SDK client configuration, not just one service client

5. Check CloudTrail is enabled and logging the relevant API calls in the correct region
6. Verify the `userAgent` field in CloudTrail events contains your string

Common format errors

Common User Agent String Format Issues

Issue	Cause	Solution
Missing end delimiter	The $ character was omitted or escaped by the shell	Ensure the string ends with $. Use single quotes or escape appropriately for your shell
Wrong prefix	Using incorrect prefix format	Use exactly `APN_1.1/pc_` as the prefix
Product code mismatch	Using Product ID or UUID instead of product code	Retrieve the alphanumeric product code from AWS Marketplace Management Portal (see Product Code Retrieval)
No API operations on resources	Resources not receiving API calls in a given month	Ensure your product performs at least one API operation on an AWS resource per month for attribution

CloudTrail Logs Verification

Use the following command to verify your User Agent string appears in CloudTrail logs:

```
aws logs filter-log-events \
  --log-group-name CloudTrail/YourLogGroup \
  --filter-pattern "APN_1.1" \
  --start-time 1640995200000
```

Look for the `userAgent` field in CloudTrail events. The format should match: `APN_1.1/pc_<YOUR-PRODUCT-CODE>$`

Validation and Testing

AWS Partner Team validation

For official validation, contact your AWS partner management team or APN Support (Partner Central login required):

- Include: AWS account ID, region, product code, sample resource ARN
- Provide: Tag screenshots or CloudTrail log excerpts and test timestamps
- Allow: 3-5 business days for validation response

Common Implementation Errors

Common Partner Revenue Measurement Implementation Issues

Method	Issue	Cause	Solution
Resource Tagging	Tags not working	Wrong tag format	Use `aws-apn-id` key with `pc:product-code` value
Resource Tagging	No revenue attribution	Resources not incurring spend	Ensure resources are actively consuming AWS services and incurring charges
Resource Tagging	Product code mismatch	Incorrect product code	Verify code in AWS Marketplace Management Portal (see Product Code Retrieval)
User Agent String	String not in CloudTrail	SDK not configured correctly	Verify SDK client configuration includes User Agent string for all service clients
User Agent String	Missing delimiter	$ stripped by shell	Use single quotes or escape the $ character appropriately
Marketplace Metering	No attribution	Product not purchased via Marketplace	Ensure customers purchase and use the product through AWS Marketplace. Open

Method	Issue	Cause	Solution
			a support ticket via APN Support

Document history for the Partner Revenue Measurement User Guide

The following table describes the documentation releases for Partner Revenue Measurement.

Change	Description	Date
Added best practices chapter and additional FAQs; updated Terraform User Agent guidance	Added best practices chapter covering Bedrock and EKS integration. Added additional FAQs section.	April 13, 2026
Updated Terraform User Agent implementation guidance	Updated Terraform section to use `provider_meta` with the `user_agent` argument (AWS provider >= 6.27.0).	April 8, 2026
Added User Agent String and AWS Marketplace Metering implementation methods; restructured documentation	Added User Agent String and AWS Marketplace Metering as new implementation methods. Restructured documentation to organize content by implementation method: AWS Marketplace Metering, Resource Tagging, and User Agent String. Updated Automation and Included AWS Services as hub pages linking to method-sp ecific content.	April 2, 2026
Initial release of Partner Revenue Measurement with ResourceTagging capability	Initial release of the PartnerRevenueMeasurement User Guide	January 29, 2026