aws

Developer Guide

AWS Partner Central

AWS Partner Central: Developer Guide

Table of Contents

The AWS Partner Central API Reference was restructured. For more information about the supported API operations, see the AWS Partner Central API Reference.

Welcome to the AWS Partner Central Developer Guide

This developer guide describes how AWS Partners can use service APIs to integrate and manage build, market, sell, and grow activities with AWS.

Topics

Using AWS Partner Central API with an AWS SDK

AWS software development kits (SDKs) are available for many popular programming languages. Each SDK provides an API, code examples, and documentation that make it easier for developers to build applications in their preferred language.

SDK documentation	Code examples
AWS SDK for C++	AWS SDK for C++ code examples
AWS CLI	AWS CLI code examples
AWS SDK for Go	AWS SDK for Go code examples
AWS SDK for Java	AWS SDK for Java code examples
AWS SDK for JavaScript	AWS SDK for JavaScript code examples
AWS SDK for Kotlin	AWS SDK for Kotlin code examples
AWS SDK for .NET	AWS SDK for .NET code examples
AWS SDK for PHP	AWS SDK for PHP code examples
AWS Tools for PowerShell	AWS Tools for PowerShell code examples
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) code examples
AWS SDK for Ruby	AWS SDK for Ruby code examples

SDK documentation	Code examples
AWS SDK for Rust	AWS SDK for Rust code examples
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP code examples
AWS SDK for Swift	AWS SDK for Swift code examples

Example availability

Can't find what you need? Request a code example by using the **Provide feedback** link at the bottom of this page.

Setup and authentication

Setting up and authenticating with the AWS Partner Central API involves three steps. Here's an overview of the process:

1. Link your AWS Marketplace Seller account to Partner Central.
2. Set up permissions using IAM.
3. Authenticate your API calls using Signature Version 4 (SigV4).

Linking your AWS account to Partner Central

Linking your AWS account to Partner Central is a prerequisite for using the API. For more information, see Linking AWS Partner Central accounts with AWS Marketplace seller accounts. You must sign in to Partner Central with an account that has alliance-lead or cloud-administrator permissions, navigate to the **`Account Linking`** section, and follow the prompts.

Setting up IAM

To use the AWS Partner Central API, you will need an AWS Identity and Access Management (IAM) role or an IAM user to start making calls. For more information, see When do I use IAM?. Follow the steps for Creating IAM roles and Creating an IAM user in your AWS account guides for this. You must create this IAM Role/User in your Partner Central-linked AWS Marketplace Seller account. IAM role/user creation does not incur any costs.

1. Create an IAM Role/User

   Sign in to the AWS Management Console, navigate to the IAM service, and follow the steps to create an IAM role or an IAM user.

2. Assign Policies:

   Attach managed policies or create custom policies as needed. To modify or expand permissions, apply additional policies to the IAM Role instead of copying and combining the content from `AWSPartnerCentralOpportunityManagement` with other permissions. Avoid duplicating managed policies, as doing so will prevent you from automatically gaining access to new features as they're released, and you'll have to manually update your policies in the future. For more details about access policies, see the Access Control documentation.

Managing AWS Marketplace offers

For managing AWS Marketplace offers and linking them to opportunities, partners must give the IAM role permission to access Catalog APIs. Ensure the role/user has permissions, such as `aws-marketplace:ListEntities` and `aws-marketplace:SearchAgreements`.

Authenticating API calls

AWS Partner Central API uses Signature Version 4 (SigV4) for authentication. Here's how to implement it:

Using the AWS SDK

AWS SDKs automatically handle request signing. Provide your AWS credentials, and the SDK does the rest.

1. For Java, see Provide temporary credentials to the AWS SDK for Java.
2. For Python (Boto3), see Credentials.
3. For JavaScript (Node.js), see Setting credentials in Node.js.
4. For .NET, see Credential and profile resolution.
5. For other programming languages and more examples, see the Tools to Build on AWS.

Authentication without using the AWS SDK

If an AWS SDK is not available for your chosen programming language, authentication involves manually creating a canonical request, signing the request, and handling the session tokens. AWS offers comprehensive guidance for using SigV4 signing. However, please note that using the AWS SDK is recommended as manual request signing increases the complexity and requires careful management of security tokens.

Every request for the awsJson1_0 protocol MUST be sent to the root URL (/) using the HTTP "POST" method using following headers.

Required Headers

Every API request for the awsJson1_0 protocol MUST be sent to the root URL (/) using the HTTP "POST" method using following headers.

Header	Required	Description
Content-Type	true	This header has a static value of `application/x-amz-json-1.0` .
Content-Length	true	The standard Content-Length header defined by RFC 9110#section-8.6.
X-Amz-Target	true for requests	For example, the value for the operation `CreatePartner` of the service `PartnerCentralAccount` is `PartnerCentralAccount.CreatePartner` .

Signing your calls with custom user-agent

When making API requests to AWS Partner Central, we recommends including the `X-Amzn-User-Agent` header to help AWS identify the source of the client application, monitor usage, and audit performance. AWS uses this header to distinguish the type of client application making the call and to gather insights about the success rate of different client implementations.

Custom user-agent header

Header Name: `X-Amzn-User-Agent`

Purpose: Distinguishes the type of client making the API request, categorizing the source of the interaction.

Format: `{Integrator}|{Product}`

Example Value: `AWS|AWS Partner CRM Connector`

Including this header in every request enables AWS to analyze request patterns, track integrations, and improve the API experience for different client systems.

Using custom headers in SDKs

To include the `X-Amzn-User-Agent` header in SDK calls, you can modify the client request behavior before making the API call. Below is a selling API example using the AWS SDK for Python (Boto3):

```
import boto3

# Define service and endpoint details
service_name = "partnercentral-selling"
endpoint_url = "https://partnercentral-selling.us-east-1.api.aws"

# Create a boto3 client for Partner Central
partner_central_client = boto3.client(
service_name=service_name,
='us-east-1',
endpoint_url=endpoint_url
)

# Function to add the custom User-Agent header

def add_version_header(params, **kwargs):
params["headers"]['X-Amzn-User-Agent'] = 'AWS|AWS Partner CRM Connector'

# Register the event to modify the request before the call is made
partner_central_client.meta.events.register(
f'before-call.{service_name}.*', add_version_header
)

# Now, whenever an API call is made using this client, the custom User-Agent header
 will be included
```

This example demonstrates how to register an event in the Boto3 SDK to automatically append the `X-Amzn-User-Agent` header to every API request. The same approach can be applied to other AWS SDKs by modifying their respective request-interception mechanisms.

> **Note**
>
> The `X-Amzn-User-Agent` header format has been simplified. We recommend to use the new format. The previous format (`CompanyName | ProductName | CRMName | ProductVersion`) remains supported for backward compatibility. Existing integrations will continue to work without modification.

Partner Central agents MCP Server

The Partner Central agents MCP Server provides Partner Central tools through the Model Context Protocol (MCP), enabling your AI agents and tools to discover and interact with opportunity management, customer insights, and funding programs through natural language.

The server handles authentication, session management, and human-in-the-loop approval for write operations maintaining control while streamlining workflows.

Overview

The Partner Central agents MCP Server is a fully managed, AWS-hosted service that connects MCP-compatible AI clients to the Partner Central agents. It uses JSON-RPC 2.0 over HTTPS with SigV4 authentication and supports Server-Sent Events (SSE) for real-time streaming responses.

The server supports multi-turn conversations, file attachments for document analysis, and a built-in approval workflow that requires your explicit consent before any write operation executes.

Agent capabilities

- **Pipeline insights** — Get conversational intelligence about your sales pipeline, including at-risk opportunities, stage progression, and closed-lost analysis
- **Opportunity creation** — Create new opportunities through natural language conversation, by uploading meeting notes, proposals, or call transcripts, or by cloning an existing opportunity
- **Opportunity summary** — Generate concise, at-a-glance summaries of any deal covering stage, spend, close date, and more
- **Sales play generation** — Build customized sales strategies combining opportunity details, industry context, and AWS solution recommendations
- **Customer profile creation** — Generate company profiles using publicly available information covering industry, business model, geography, and recent developments
- **Solution recommendation** — Cross-reference your registered solutions against opportunity requirements to find the best match
- **Funding recommendation** — Evaluate opportunities against available AWS funding programs, estimate amounts, and create fund requests

- **Next step recommendations** — Get prioritized action plans grounded in AWS co-sell standards and stage progression guidance
- **Opportunity progression** — Upload supporting documents, extract relevant data, and progress opportunities through pipeline stages
- **AI-assisted product listing** — Generate high-quality AWS Marketplace product listings from your existing digital assets, score listing strength against AWS Marketplace standards, and receive field-level recommendations to improve discoverability. For more information, see AI-assisted product listing in the *AWS Marketplace Seller Guide*.

Key benefits

- **Conversational access to Partner Central** — Ask questions in natural language instead of navigating complex console workflows
- **More time selling** — Create opportunities through a short conversation instead of completing a multi-step form, which reduces data entry and lets partner sales teams spend more time selling, with the agent recommending improvements so partners submit higher-quality opportunities and improve pipeline hygiene
- **Human-in-the-loop safety** — All write operations require your explicit approval before execution
- **Multi-turn conversations** — Refine your queries within a session without repeating context
- **File analysis** — Attach documents (PDF, DOCX, XLSX, CSV, images) for the agent to analyze alongside your questions
- **Centralized access management** — Control access through IAM policies with fine-grained permissions
- **Sandbox testing** — Test workflows in an isolated sandbox environment before touching production data
- **Streaming responses** — Get feedback via SSE as the agent processes your request

Usage examples

All AI-generated insights include a Session ID for traceability. Data is isolated to the logged-in partner's own opportunities. All content carries clear disclosure labels and is governed by the AWS Responsible AI Policy.

Opportunity creation

The agent creates a new opportunity from a natural language description or an uploaded document (PDF, DOCX, XLSX, TXT). It extracts customer name, project scope, expected close date, and other required fields, enriches customer details from publicly available web data, validates the draft against AWS readiness requirements, and creates the opportunity through the Partner Central Selling API after you approve.

- "Create an opportunity for Acme Corp — they want to migrate their data warehouse to Redshift, target close date end of Q3, expected $40K monthly AWS spend"
- "Here are my call notes from yesterday's meeting with GlobalTech — create an opportunity from this transcript"
- "Use this proposal PDF to create an opportunity for the customer's SAP migration"

Opportunity cloning

The agent creates a new opportunity from an existing one, merges in the new customer or project details you provide, and validates that at least one differentiating field has changed before submission so duplicates are not rejected by AWS review.

- "Clone opportunity O1234567890 for a new customer — Globex, same workload, target close date end of Q4"
- "Create an opportunity like O9876543210 but for the customer's production environment instead of the sandbox"

Pipeline insights

Get conversational intelligence about your sales pipeline. The agent analyzes stage progression, deadlines, stalled deals, and closed-lost patterns to surface what matters most.

- "Which opportunities need my attention this week?"
- "How many opportunities are closing next month?"
- "What are the top reasons we've lost opportunities in the last 6 months?"

Opportunity summary

The agent synthesizes company name, industry, stage, expected monthly AWS spend, target close date, and other key details into a concise summary — no need to scan individual form fields.

- "Give me a summary of opportunity O1234567890"
- "What's the current status and key details for the Acme Corp deal?"

Sales play generation

The agent builds a customized sales strategy by combining the opportunity's details, the customer's industry context, and relevant AWS solution recommendations into a ready-to-use sales play.

- "Generate a sales play for opportunity O1234567890"
- "What's the best approach to sell cloud migration to this financial services customer?"
- "Build me a sales strategy for the GlobalTech data analytics deal"

Customer profile creation

The agent generates a company profile using publicly available information — covering industry classification, business model (B2B/B2C/hybrid), geographic presence, company size, market focus, and recent business developments. Profiles are labeled "Generated with publicly available data and AWS AI insights."

- "Create a customer profile for Acme Corp"
- "What do we know about this customer's industry and business model?"
- "Pull together a company overview for my upcoming meeting with GlobalTech"

Solution recommendation

The agent cross-references your registered solutions against opportunity requirements, showing solution name, description, and whether it's already attached to the opportunity.

- "Which of our solutions best match opportunity O1234567890?"
- "Is our data analytics solution already attached to this deal?"

- "Recommend solutions for a customer looking to migrate their SAP workloads"

Funding recommendation

The agent evaluates each opportunity against available AWS funding programs based on opportunity details and program eligibility criteria. When a match is found, it displays program name, description, and detailed reasoning. You can then estimate funding amounts, create auto-populated fund request drafts, or learn more about programs conversationally. SCA (Strategic Collaboration Agreement) budget availability is surfaced when relevant, and all actions respect IAM permissions.

- "Am I eligible for any funding programs on opportunity O6789012345?"
- "Estimate the funding amount for a POC with this customer"
- "Create a MAP benefit application for this opportunity"

Next step recommendations

The agent evaluates the opportunity against AWS's stage progression guidance, compares current data against criteria for well-qualified opportunities, and identifies exactly what information you still need to collect. The result is a prioritized action plan grounded in AWS co-sell standards.

- "What do I need to do next to advance opportunity O1234567890?"
- "Is this opportunity ready for submission? What fields are missing?"
- "What are the requirements to move this deal from Prospect to Qualified?"

Opportunity progression

The agent accepts supporting documents (meeting transcripts, call notes, email summaries), extracts relevant information, maps it to opportunity fields, evaluates stage requirements, and updates the opportunity. If gaps remain, it returns a breakdown of satisfied vs. unsatisfied requirements.

- "Here are my call notes — update opportunity O1234567890 with the relevant details"
- "I'm attaching the meeting transcript. Progress this opportunity based on what we discussed."
- "Review this email summary and tell me which opportunity fields it satisfies"

Agentic experience for partner onboarding

The agent automates partner onboarding to Partner Central and provides guided assistance for Marketplace seller setup and PRM compliance — all through natural language.

Agent capabilities:

- **Partner profile automation** — Scan your website, auto-populate your partner profile, and manage visibility, alliance lead contact, training domains, and account connections
- **Seller setup guidance** — Step-by-step guidance for Marketplace seller registration including tax forms, banking, compliance (KYC/BAV/SU), ESC catalog, and service-linked roles
- **PRM compliance guidance** — Assess PRM readiness, retrieve product codes for revenue tagging, and verify subsidiary account connections for consolidated revenue attribution

Partner profile automation

The agent scans your company website to extract and auto-populate your partner profile, then guides you through any remaining gaps. It can update profile fields, change visibility, manage your alliance lead contact, link training certification domains, and handle account connection invitations.

- "Can you look at my website and fill in my profile?"
- "Make our profile public so AWS customers can find us"
- "Update our alliance lead contact to [name] at [email]"
- "Connect our EMEA subsidiary account"

Seller setup guidance

The agent walks through Marketplace seller registration end-to-end, determining the right tax form based on your country and entity type, explaining banking and disbursement setup by region, and identifying compliance requirements (KYC, Bank Account Verification, Secondary User verification) that apply to you.

- "I want to start selling on Marketplace — where do I begin?"
- "What tax form do I need? I'm a company in Germany"
- "Do I need KYC? I'm selling to customers in France"

- "How do I set up payments if I'm outside the US?"

PRM compliance guidance

The agent checks whether your account is PRM-ready — verifying that your account is connected, a listing exists, and your product code is available for tagging. For partners with subsidiary accounts, it verifies that all seller accounts are properly linked for consolidated revenue attribution.

- "Am I PRM compliant?"
- "Where do I find my product code for tagging?"
- "Tell me all the steps I need to perform to be PRM compliant"

Get started

Ready to set up the Partner Central agents MCP Server? Head to the Getting started guide for step-by-step setup instructions.

Getting Started with the Partner Central Agent MCP Server

This guide walks you through setting up programmatic access to the Partner Central Agent MCP Server using a custom MCP client. The server uses direct HTTPS with SigV4 authentication — no proxy or IDE plugin required.

Prerequisites

Before you begin, make sure you have:

- An active Partner Central account (migrated to the AWS console)
- An AWS account with IAM permissions for Partner Central
- AWS CLI installed and configured with credentials
- Access to the us-east-1 (N. Virginia) region
- HTTPS connectivity to `partnercentral-agents-mcp.us-east-1.api.aws`
- TLS 1.2+ support in your HTTP client
- An MCP-compatible client that supports JSON-RPC 2.0 and SigV4 request signing

Step 1: Set up IAM permissions

The Partner Central Agent MCP Server requires IAM permissions at two levels: protocol access (to communicate with the MCP endpoint) and data access (to perform Partner Central operations).

Attaching IAM policies

To attach a policy to your IAM identity using the AWS Management Console:

1. Open the IAM console.
2. In the left navigation pane, choose **Users**, **User groups**, or **Roles** depending on the identity you want to attach the policy to, then choose the name of the specific user, group, or role.
3. Choose the **Permissions** tab.
4. Choose **Attach policies** (or **Add permissions** if it's the first time).
5. In the policy list, search for and select the managed policy you want to attach (for example, a custom policy you created from the JSON blocks below).
6. Choose **Attach policies** (or **Next** and then **Add permissions**) to confirm.

The permissions take effect immediately. You can attach multiple policies to the same identity.

Recommended: Use the managed policy

The simplest way to grant MCP protocol access is to attach the `AWSMcpServiceActionsFullAccess` managed policy to your IAM identity. This policy includes all permissions needed to interact with the MCP server.

For fine-grained control, you can use the `aws:IsMcpServiceAction` condition key in your IAM policies to scope permissions specifically to MCP service actions.

Minimum permissions for MCP protocol access

At minimum, your IAM identity needs this action to interact with the MCP server:

Action	Description
`partnercentral:UseSession`	Required to create, update, and retrieve conversation sessions

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "partnercentral:UseSession"
            ],
            "Resource": "*",
   "Condition": {
                "Bool": {
                    "aws:IsMcpServiceAction": "true"
                }
            }
        }
    ]


}
```

Data access permissions

To actually perform Partner Central operations through the agent, you need additional permissions based on your use case.

Opportunity management:

```
{
    "Effect": "Allow",
    "Action": [
        "partnercentral:List*",
        "partnercentral:Get*",
        "partnercentral:CreateOpportunity",
        "partnercentral:UpdateOpportunity",
        "partnercentral:SubmitOpportunity",
        "partnercentral:AssignOpportunity",
        "partnercentral:AssociateOpportunity",
        "partnercentral:DisassociateOpportunity"
    ],
    "Resource": "*"
}
```

Funding programs:

```
{
    "Effect": "Allow",
    "Action": [
        "partnercentral:ListBenefitAllocations",
        "partnercentral:ListBenefitApplications",
        "partnercentral:CreateBenefitApplication",
        "partnercentral:GetBenefitApplication",
        "partnercentral:UpdateBenefitApplication",
        "partnercentral:SubmitBenefitApplication",
        "partnercentral:AmendBenefitApplication",
        "partnercentral:CancelBenefitApplication",
        "partnercentral:RecallBenefitApplication",
        "partnercentral:AssociateBenefitApplicationResource",
        "partnercentral:DisassociateBenefitApplicationResource"
    ],
    "Resource": "*"
}
```

Marketplace access:

```
{
    "Effect": "Allow",
    "Action": [
        "aws-marketplace:DescribeEntity",
        "aws-marketplace:DescribeAgreement",
        "aws-marketplace:SearchAgreements",
        "aws-marketplace:ListEntities"
    ],
    "Resource": "*"
}
```

Partner Central Onboarding:

```
{
    "Effect": "Allow",
    "Action": [
        "partnercentral:ListPartners",
        "partnercentral:GetPartner",
        "partnercentral:GetProfileVisibility",
        "partnercentral:GetAllianceLeadContact",
```

```
        "partnercentral:GetProfileUpdateTask",
        "partnercentral:StartProfileUpdateTask",
        "partnercentral:CancelProfileUpdateTask",
        "partnercentral:PutProfileVisibility",
        "partnercentral:PutAllianceLeadContact",
        "partnercentral:SendEmailVerificationCode",
        "partnercentral:AssociateAwsTrainingCertificationEmailDomain",
        "partnercentral:DisassociateAwsTrainingCertificationEmailDomain",
        "partnercentral:GetAccountConnections",
        "partnercentral:GetConnectionInvitations",
        "partnercentral:CreateConnectionInvitation",
        "partnercentral:CancelConnectionInvitation",
        "partnercentral:RespondConnectionInvitation",
        "partnercentral:CancelConnection",
        "partnercentral:ManageConnectionPreferences"
    ],
    "Resource": "*"
}
```

For Marketplace seller setup, also add:

```
{
    "Effect": "Allow",
    "Action": [
        "aws-marketplace:ListEntities",
        "aws-marketplace:DescribeEntity",
        "aws-marketplace:DescribeChangeSet",
        "aws-marketplace:StartChangeSet"
    ],
    "Resource": "*"
}
```

For service-linked role management (Resale Authorizations/CPPO):

```
{
    "Effect": "Allow",
    "Action": [
        "iam:GetRole",
        "iam:CreateServiceLinkedRole"
    ],
    "Resource": "*"
}
```

Full access policy

For development and testing, you can combine all permissions into a single policy:

```
aws iam create-policy \
    --policy-name PartnerCentralAgentsFullAccess \
    --policy-document '{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "partnercentral:UseSession",
                    "partnercentral:List*",
                    "partnercentral:Get*",
                    "partnercentral:CreateOpportunity",
                    "partnercentral:UpdateOpportunity",
                    "partnercentral:SubmitOpportunity",
                    "partnercentral:AssignOpportunity",
                    "partnercentral:AssociateOpportunity",
                    "partnercentral:DisassociateOpportunity",
                    "partnercentral:CreateResourceSnapshot",
                    "partnercentral:CreateResourceSnapshotJob",
                    "partnercentral:StartResourceSnapshotJob",
                    "partnercentral:CreateEngagement",
                    "partnercentral:CreateEngagementInvitation",
                    "partnercentral:RejectEngagementInvitation",
                    "partnercentral:StartEngagementByAcceptingInvitationTask",
                    "partnercentral:StartEngagementFromOpportunityTask",
                    "partnercentral:CreateBenefitApplication",
                    "partnercentral:UpdateBenefitApplication",
                    "partnercentral:SubmitBenefitApplication",
                    "partnercentral:AmendBenefitApplication",
                    "partnercentral:CancelBenefitApplication",
                    "partnercentral:RecallBenefitApplication",
                    "partnercentral:AssociateBenefitApplicationResource",
                    "partnercentral:DisassociateBenefitApplicationResource",
                    "partnercentral:ListPartners",
                    "partnercentral:StartProfileUpdateTask",
                    "partnercentral:CancelProfileUpdateTask",
                    "partnercentral:PutProfileVisibility",
                    "partnercentral:PutAllianceLeadContact",
                    "partnercentral:SendEmailVerificationCode",
                    "partnercentral:AssociateAwsTrainingCertificationEmailDomain",
```

```
                    "partnercentral:DisassociateAwsTrainingCertificationEmailDomain",
                    "partnercentral:CreateConnectionInvitation",
                    "partnercentral:CancelConnectionInvitation",
                    "partnercentral:RespondConnectionInvitation",
                    "partnercentral:CancelConnection",
                    "partnercentral:ManageConnectionPreferences"
                ],
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "aws-marketplace:DescribeEntity",
                    "aws-marketplace:DescribeAgreement",
                    "aws-marketplace:SearchAgreements",
                    "aws-marketplace:ListEntities",
                    "aws-marketplace:DescribeChangeSet",
                    "aws-marketplace:StartChangeSet"
                ],
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "iam:GetRole",
                    "iam:CreateServiceLinkedRole"
                ],
                "Resource": "*"
            }
        ]
    }'
```

Read-only policy

For production environments or read-only use cases, restrict permissions to read operations:

```
aws iam create-policy \
    --policy-name PartnerCentralAgentReadOnly \
    --policy-document '{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
```

```
                "Action": [
                    "partnercentral:UseSession",
                    "partnercentral:List*",
                    "partnercentral:Get*",
                    "partnercentral:ListPartners",
                    "partnercentral:GetPartner",
                    "partnercentral:GetProfileVisibility",
                    "partnercentral:GetAllianceLeadContact",
                    "partnercentral:GetProfileUpdateTask",
                    "partnercentral:GetAccountConnections",
                    "partnercentral:GetConnectionInvitations"
                ],
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "aws-marketplace:DescribeEntity",
                    "aws-marketplace:DescribeAgreement",
                    "aws-marketplace:SearchAgreements",
                    "aws-marketplace:ListEntities",
                    "aws-marketplace:DescribeChangeSet"
                ],
                "Resource": "*"
            },
            {
                "Effect": "Allow",
                "Action": [
                    "iam:GetRole"
                ],
                "Resource": "*"
            }
        ]
    }'
```

Step 2: Connect your MCP client

The Partner Central Agent MCP Server uses direct HTTPS with SigV4 request signing. There is no proxy layer — your MCP client sends JSON-RPC 2.0 requests directly to the endpoint.

Endpoint

```
https://partnercentral-agents-mcp.us-east-1.api.aws/mcp
```

Authentication

All requests must be signed with AWS Signature Version 4 using:

- Service name: `partnercentral-agents-mcp`
- Region: `us-east-1`

Initialize the MCP connection

Send an `initialize` request to establish the protocol:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "initialize",
    "params": {
        "protocolVersion": "2025-03-26",
        "capabilities": {},
        "clientInfo": {
            "name": "my-partner-client",
            "version": "1.0.0"
        }
    }
}
```

Expected response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": {
        "protocolVersion": "2025-03-26",
        "capabilities": {
            "tools": {
                "listChanged": false
            }
```

```
        },
        "serverInfo": {
            "name": "PartnerCentralAgentMCPServer",
            "version": "1.0.0"
        }
    }
}
```

List available tools

```
{
    "jsonrpc": "2.0",
    "id": 2,
    "method": "tools/list",
    "params": {}
}
```

Signing your calls with MCP header

When making requests to Partner Central agents MCP, we recommend including the custom MCP header using the following methods to help AWS identify the source of the client application, monitor usage, and audit performance. AWS uses this header to distinguish the type of client application making the call and to gather insights about the success rate of different client implementations.

Method 1: _meta field (programmatic/stateless MCP)

For code that directly constructs MCP `tools/call` requests, provide the `_meta` field on requests.

```
{
    "method": "tools/call",
    "params": {
        "name": "sendMessage",
        "arguments": {
            "content": [
                {
                    "type": "text",
                    "text": "List my open opportunities with expected close date in Q1
 2026"
                }
            ],
```

```
            "catalog": "AWS"
        },
        "_meta": {
            "integrator": "<Integrator's Company Name / Direct>",
            "sourceProduct": "<Integrator's Application Name>"
        }
    }
}
```

Method 2: clientInfo (session-based custom agents)

For custom MCP clients establishing sessions, provide MCP header info inside the `clientInfo` field:

```
{
    "method": "initialize",
    "params": {
        "protocolVersion": "2024-11-05",
        "clientInfo": {
            "integrator": "<Integrator's Company Name / Direct>",
            "sourceProduct": "<Integrator's Application Name>"
        }
    }
}
```

Fields in `clientInfo`:

- `integrator` — Company name or "Direct" for partners

  Example: `AWS`
- `sourceProduct` — Product/agent name

  Example: `AWS CRM Connector`

Method 3: URL parameter (hosted MCP only)

Only for hosted MCP clients where the integrator cannot modify protocol fields. Use the URL parameter:

Server URL: `https://mcp.partnercentral.aws?appId=<Integrator's Company Name / Direct>`

Step 3: Verify your setup

Send a simple message to confirm everything is working. Use the `Sandbox` catalog for testing:

```
{
    "jsonrpc": "2.0",
    "id": 3,
    "method": "tools/call",
    "params": {
        "name": "sendMessage",
        "arguments": {
            "content": [
                {
                    "type": "text",
                    "text": "Hello, what can you help me with?"
                }
            ],
            "catalog": "Sandbox"
        }
    }
}
```

If you receive a response with `"status": "complete"` and a text reply from the agent, your setup is working correctly. The response will also include a `sessionId` that you can use for follow-up messages.

Step 4: Run your first tasks

Query your opportunities

```
{
    "jsonrpc": "2.0",
    "id": 4,
    "method": "tools/call",
    "params": {
        "name": "sendMessage",
        "arguments": {
            "sessionId": "session-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
            "content": [
                {
                    "type": "text",
```

```
                    "text": "List my open opportunities with expected revenue over
 $50K"
                }
            ],
            "catalog": "AWS"
        }
    }
}
```

Check funding eligibility

```
{
    "jsonrpc": "2.0",
    "id": 5,
    "method": "tools/call",
    "params": {
        "name": "sendMessage",
        "arguments": {
            "sessionId": "session-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
            "content": [
                {
                    "type": "text",
                    "text": "Am I eligible for MAP funding for opportunity
 O1234567890?"
                }
            ],
            "catalog": "AWS"
        }
    }
}
```

Retrieve session history

```
{
    "jsonrpc": "2.0",
    "id": 6,
    "method": "tools/call",
    "params": {
        "name": "getSession",
        "arguments": {
            "sessionId": "session-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
            "catalog": "AWS"
        }
```

```
    }
}
```

Partner onboarding

```
{
    "jsonrpc": "2.0",
    "id": 7,
    "method": "tools/call",
    "params": {
        "name": "sendMessage",
        "arguments": {
            "sessionId": "session-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
            "content": [
                {
                    "type": "text",
                    "text": "Help me onboard to Partner Central"
                }
            ],
            "catalog": "AWS"
        }
    }
}
```

Other onboarding tasks to try:

- "Guide me through the tax interview process"
- "Can you look at my website and fill in my partner profile?"
- "What do I still need to do to be ready to sell on Marketplace?"

Security considerations

- Do not pass AWS credentials through MCP tool parameters. Authentication is handled by SigV4 request signing at the transport layer.
- Use the Sandbox catalog for testing and development. The `"Sandbox"` catalog provides an isolated environment that does not affect production partner data.
- Apply least-privilege IAM policies in production. Use the read-only policy for monitoring and reporting use cases. Only grant write permissions when the user needs to update opportunities or submit funding applications.

- Review write operations carefully. The server uses human-in-the-loop approval for all write operations. When a write action is proposed, review the parameters before approving.
- Session data is transient. Sessions expire 48 hours after creation. Do not rely on sessions for long-term data storage.
- File uploads go to an ephemeral S3 bucket. Uploaded files are stored temporarily and are not retained permanently. Do not upload files containing credentials, secrets, or other sensitive information.

Next steps

- **Configuration Reference** — Full reference for endpoint, IAM actions, session management, and error codes
- **Tools Reference** — Detailed documentation for `sendMessage` and `getSession` tools

Configuration Reference

This page provides the complete reference for connecting to and configuring the Partner Central Agent MCP Server.

Endpoint

Property	Value
URL	`https://partnercentral-agents-mcp.us-east-1.api.aws/mcp`
Region	`us-east-1` (N. Virginia) only
Protocol	JSON-RPC 2.0 over HTTPS
Streaming	Server-Sent Events (SSE)
Authentication	AWS Signature Version 4
SigV4 service name	`partnercentral-agents-mcp`
TLS requirement	TLS 1.2 or higher

IAM permissions

All IAM actions use the `partnercentral:` prefix unless otherwise noted.

MCP protocol access

The simplest way to grant MCP protocol access is to attach the `AWSMcpServiceActionsFullAccess` managed policy to your IAM identity. This policy includes all permissions needed to interact with the MCP server. For fine-grained control, you can use the `aws:IsMcpServiceAction` condition key in your IAM policies to scope permissions specifically to MCP service actions.

Action	Description
`partnercentral:UseSession`	Create, update, and retrieve conversation sessions

Opportunity management

Action	Description
`partnercentral:List*`	List opportunities, solutions, and other Partner Central resources
`partnercentral:Get*`	Retrieve details for individual opportunities and other resources
`partnercentral:CreateOpportunity`	Create an opportunity
`partnercentral:UpdateOpportunity`	Modify opportunity fields (stage, close date, revenue, etc.)
`partnercentral:SubmitOpportunity`	Submit an opportunity for AWS review
`partnercentral:AssignOpportunity`	Assign an opportunity to a partner represent ative

Action	Description
partnercentral:AssociateOpportunity	Link an opportunity to another resource (solution, etc.)
partnercentral:DisassociateOpportunity	Remove a link between an opportunity and another resource
partnercentral:StartEngagementFromOpportunityTask	Submit opportunities to start an engagement from an opportunity

Funding programs

Action	Description
partnercentral:ListBenefitAllocations	List available benefit allocations for your account
partnercentral:ListBenefitApplications	List submitted benefit applications
partnercentral:CreateBenefitApplication	Create a new benefit application (MAP, POC, WMP)
partnercentral:GetBenefitApplication	Retrieve details of a specific benefit application
partnercentral:UpdateBenefitApplication	Update a pending benefit application
partnercentral:AssociateBenefitApplicationResource	Link a resource to a benefit application
partnercentral:DisassociateBenefitApplicationResource	Remove a resource link from a benefit application

Marketplace access

Action	Description
`aws-marketplace:DescribeEntity`	Retrieve details of a marketplace entity
`aws-marketplace:SearchAgreements`	Search marketplace agreements
`aws-marketplace:ListEntities`	List marketplace entities

Onboarding Agent

Partner account management (`partnercentral`):

Action	Description
`partnercentral:ListPartners`	List partner registrations for the account
`partnercentral:GetPartner`	Retrieve partner registration and profile details
`partnercentral:GetProfileVisibility`	Retrieve current profile visibility (PUBLIC/PRIVATE)
`partnercentral:GetAllianceLeadContact`	Retrieve alliance lead contact information
`partnercentral:GetProfileUpdateTask`	Check status of a pending async profile update
`partnercentral:StartProfileUpdateTask`	Submit a partner profile update (async)
`partnercentral:CancelProfileUpdateTask`	Cancel a pending profile update task
`partnercentral:PutProfileVisibility`	Change profile visibility to PUBLIC or PRIVATE

Action	Description
partnercentral:PutAllianceLeadContact	Update alliance lead contact (name, title, email)
partnercentral:SendEmailVerificationCode	Send a verification code for email-based domain/contact changes
partnercentral:AssociateAwsTrainingCertificationEmailDomain	Link an email domain for training certification tracking
partnercentral:DisassociateAwsTrainingCertificationEmailDomain	Remove a linked training certification email domain
partnercentral:GetAccountConnections	List active account connections (e.g., subsidiary links)
partnercentral:GetConnectionInvitations	List pending sent and received connection invitations
partnercentral:CreateConnectionInvitation	Send a connection invitation to another AWS account
partnercentral:CancelConnectionInvitation	Cancel a pending outgoing connection invitation
partnercentral:RespondConnectionInvitation	Accept or reject an incoming connection invitation
partnercentral:CancelConnection	Cancel an active account connection
partnercentral:ManageConnectionPreferences	Get or update sharing preferences between connected accounts

Marketplace seller setup (`aws-marketplace`):

Action	Description
`aws-marketplace:ListEntities`	List Marketplace entities (e.g., seller entities)
`aws-marketplace:DescribeEntity`	Retrieve full details of a Marketplace entity
`aws-marketplace:DescribeChangeSet`	Check status of a submitted change set
`aws-marketplace:StartChangeSet`	Submit a Marketplace change set (e.g., seller profile update)

IAM — service-linked role (`iam`):

Action	Description
`iam:GetRole`	Check whether the Marketplace Resale Authorization service-linked role exists
`iam:CreateServiceLinkedRole`	Create the `AWSServiceRoleForMarketplaceResaleAuthorization` role

Catalog environments

Catalog	Description
`"AWS"`	Production environment. All operations affect live partner data.
`"Sandbox"`	Isolated testing environment. No impact on production data. Use for development and validation.

Session management

Property	Value
Session ID format	UUID v4 with `session-` prefix (e.g., `session-550e8400-e29b-41d4-a716-446655440000`)
Session creation	Automatic on first `sendMessage` call without a `sessionId`
Session expiry	48 hours from session creation (absolute expiry, not inactivity-based)

File upload

Property	Value
Max files per message	3
Image size limit	3.75 MB
Document size limit	4.5 MB
Allowed extensions	doc, `docx`, `pdf`, png, `jpeg`, `xlsx`, `csv`, `txt`
S3 bucket (production)	aws-partner-central-marketplace-ephemeral-writeonly-files
Upload path	s3://{bucket}/{aws-account-id}/
S3 URI requirement	Must include `versionId` query parameter

Upload workflow

1. Upload your file to the appropriate S3 bucket under your AWS account ID prefix.
2. Note the S3 URI including the `versionId` returned by the upload.

3. Include the file as a `document` content block in your `sendMessage` call.

Example S3 URI format:

```
s3://aws-partner-central-marketplace-ephemeral-writeonly-files/123456789012/my-document.pdf?versionId=abc123
```

Error codes

Code	Name	Description
-32001	AUTHENTICATION_FAILURE	SigV4 signature is invalid or credentials have expired
-31004	TOOL_PERMISSION_DENIED	IAM identity lacks the required `partnercentral:` action for this operation
-32002	ACCESS_DENIED	General access denied (account not enrolled, region mismatch, etc.)
-32004	LIMIT_EXCEEDED	Rate limit or quota exceeded. Retry with exponential backoff.
-30001	RESOURCE_NOT_FOUND	Requested resource (session, opportunity, etc.) does not exist
-32600	INVALID_REQUEST	Malformed JSON-RPC request or invalid parameters
-32603	INTERNAL_ERROR	Server-side error. Retry the request.

Rate limits

The server enforces per-account rate limits:

Operation	Sustained rate	Burst
`sendMessage`	2 requests per minute	10
All other operations	10 requests per minute	20

If you exceed these limits, the server returns error code `-32004` (LIMIT_EXCEEDED). Implement exponential backoff with jitter in your client to handle rate limiting gracefully.

Streaming (SSE) event types

When `stream` is set to `true` in a `sendMessage` call, the server returns Server-Sent Events with these event types:

Event Type	Description
`stream_start`	Stream connection established
`assistant-response.start`	Agent has begun generating a response
`assistant-response.delta`	Incremental text chunk of the agent's response
`assistant-response.completed`	Agent has finished generating the response
`server-tool-use`	Agent is invoking an internal tool (read operation)
`server-tool-response`	Result of an internal tool invocation
`tool_approval_request`	Agent is requesting human approval for a write operation
`stream_end`	Stream connection closing

Event Type	Description
done	Final event indicating the SSE stream is complete

Next steps

- **Tools Reference** — Detailed documentation for `sendMessage` and `getSession` tools

Tools Reference

The Partner Central Agent MCP Server exposes two MCP tools: `sendMessage` for all agent interactions, and `getSession` for retrieving session state. All Partner Central operations — opportunity queries, funding applications, document analysis — are handled through natural language via `sendMessage`.

Tools overview

Tool	Description	Category
`sendMessage`	Send messages to the Partner Central AI agent. Supports text, file attachments, and human-in-the-loop approval responses.	Read / Write
`getSession`	Retrieve session state including conversation history, events, and metadata.	Read-only

`sendMessage`

Primary tool for all Partner Central AI agent interactions. Use this tool to ask questions, request actions, attach documents for analysis, and respond to approval requests for write operations.

The agent maintains conversation context within a session, so you can ask follow-up questions without repeating prior context.

Parameters

- `content` (required) — Array of content blocks. Each block must include a `type` field that determines the block structure. You can include multiple blocks in a single message (e.g., text + document attachment).

  Content block types:

Type	Fields	Description
`text`	`type` (required), `text` (required)	User message text sent to the agent
`document`	`type` (required), `filename` (required), `s3Uri` (required)	File attachment for the agent to analyze. The `s3Uri` must include a `versionId` parameter.
`tool_approval_response`	`type` (required), `toolUseId` (required), `decision` (required), `message` (optional)	Response to a human-in-the-loop approval request

- `catalog` (required) — Target environment for the operation.

  Valid values: `"AWS"` (production), `"Sandbox"` (testing)

- `sessionId` (optional) — UUID v4 identifying an existing session to continue. Omit to create a new session. Format: `session-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.

  Default: A new session is created automatically.

- `stream` (optional) — Enable Server-Sent Events (SSE) streaming for real-time response delivery.

  Valid values: `true`, `false`

  Default: `false`

Response

The response includes:

Field	Description
`sessionId`	Session identifier for follow-up messages
`status`	Response status: `"complete"` , `"requires_approval"` , or `"error"`
`content`	Array of response content blocks from the agent

Examples

Basic text message (new session)

Request:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "tools/call",
    "params": {
        "name": "sendMessage",
        "arguments": {
            "content": [
                {
                    "type": "text",
                    "text": "List my open opportunities with expected close date in Q1
 2026"
                }
            ],
            "catalog": "AWS"
        }
    }
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": {
        "content": [
            {
                "type": "text",
                "text": "I found 12 open opportunities with expected close dates in Q1
 2026. Here's a summary:\n\n1. **O1234567890** - Acme Corp Cloud Migration - $250,000 -
 Qualified stage\n2. **O1234567891** - GlobalTech Data Analytics - $180,000 - Prospect
 stage\n..."
            }
        ],
        "sessionId": "session-550e8400-e29b-41d4-a716-446655440000",
        "status": "complete"
    }
}
```

Follow-up message (existing session)

Request:

```
{
    "jsonrpc": "2.0",
    "id": 2,
    "method": "tools/call",
    "params": {
        "name": "sendMessage",
        "arguments": {
            "sessionId": "session-550e8400-e29b-41d4-a716-446655440000",
            "content": [
                {
                    "type": "text",
                    "text": "Tell me more about O1234567890. Is it ready for
 submission?"
                }
            ],
            "catalog": "AWS"
        }
    }
}
```

File attachment

Upload a document to S3 first, then reference it in the message:

```
{
    "jsonrpc": "2.0",
    "id": 3,
    "method": "tools/call",
    "params": {
        "name": "sendMessage",
        "arguments": {
            "sessionId": "session-550e8400-e29b-41d4-a716-446655440000",
            "content": [
                {
                    "type": "text",
                    "text": "Review this customer proposal and suggest which
 opportunity it aligns with"
                },
                {
                    "type": "document",
                    "filename": "acme-proposal.pdf",
                    "s3Uri": "s3://aws-partner-central-marketplace-ephemeral-writeonly-
files/123456789012/acme-proposal.pdf?versionId=abc123def456"
                }
            ],
            "catalog": "AWS"
        }
    }
}
```

File upload constraints:

- Maximum 3 files per message
- Image size limit: 3.75 MB
- Document size limit: 4.5 MB
- Allowed extensions: `doc`, `docx`, `pdf`, `png`, `jpeg`, `xlsx`, `csv`, `txt`
- Files must be uploaded to `s3://{bucket}/{your-aws-account-id}/`
- The S3 URI must include the `versionId` query parameter

Human-in-the-loop approval workflow

When the agent needs to perform a write operation (e.g., update an opportunity, submit a funding application), it returns a `"requires_approval"` status with the proposed action details. You must respond with a `tool_approval_response` content block.

Step 1 — Agent requests approval:

```
{
    "jsonrpc": "2.0",
    "id": 4,
    "result": {
        "content": [
            {
                "type": "text",
                "text": "I'd like to update opportunity O1234567890 with the following
 changes:\n- Target close date: 2026-03-31\n- Expected revenue: $300,000\n- Stage:
 Qualified\n\nPlease approve, reject, or override this action."
            },
            {
                "type": "tool_approval_request",
                "toolUseId": "tool-use-98765",
                "toolName": "update_opportunity_enhanced",
                "parameters": {
                    "opportunityId": "O1234567890",
                    "targetCloseDate": "2026-03-31",
                    "expectedRevenue": 300000,
                    "stage": "Qualified"
                }
            }
        ],
        "sessionId": "session-550e8400-e29b-41d4-a716-446655440000",
        "status": "requires_approval"
    }
}
```

Step 2 — Approve the action:

```
{
    "jsonrpc": "2.0",
    "id": 5,
    "method": "tools/call",
    "params": {
```

```
        "name": "sendMessage",
        "arguments": {
            "sessionId": "session-550e8400-e29b-41d4-a716-446655440000",
            "content": [
                {
                    "type": "tool_approval_response",
                    "toolUseId": "tool-use-98765",
                    "decision": "approve"
                }
            ],
            "catalog": "AWS"
        }
    }
}
```

Step 2 (alternative) — Reject the action:

```
{
    "jsonrpc": "2.0",
    "id": 5,
    "method": "tools/call",
    "params": {
        "name": "sendMessage",
        "arguments": {
            "sessionId": "session-550e8400-e29b-41d4-a716-446655440000",
            "content": [
                {
                    "type": "tool_approval_response",
                    "toolUseId": "tool-use-98765",
                    "decision": "reject",
                    "message": "The expected revenue should be $250,000, not $300,000"
                }
            ],
            "catalog": "AWS"
        }
    }
}
```

Step 2 (alternative) — Override with custom response:

```
{
    "jsonrpc": "2.0",
    "id": 5,
```

```
    "method": "tools/call",
    "params": {
        "name": "sendMessage",
        "arguments": {
            "sessionId": "session-550e8400-e29b-41d4-a716-446655440000",
            "content": [
                {
                    "type": "tool_approval_response",
                    "toolUseId": "tool-use-98765",
                    "decision": "override",
                    "message": "Use expected revenue of $250,000 and keep the stage as
 Prospect instead"
                }
            ],
            "catalog": "AWS"
        }
    }
}
```

Approval decision values:

Decision	Behavior
`"approve"`	Execute the tool with the proposed parameter s
`"reject"`	Do not execute the tool. Optional `message` explains why.
`"override"`	Provide a custom response or modified instructions via `message`

Streaming with SSE

Enable streaming to receive incremental response chunks as the agent processes your request:

Request:

```
{
    "jsonrpc": "2.0",
    "id": 6,
```

```
    "method": "tools/call",
    "params": {
        "name": "sendMessage",
        "arguments": {
            "sessionId": "session-550e8400-e29b-41d4-a716-446655440000",
            "content": [
                {
                    "type": "text",
                    "text": "Analyze my pipeline and identify opportunities at risk"
                }
            ],
            "catalog": "AWS",
            "stream": true
        }
    }
}
```

The server responds with a stream of SSE events:

```
event: stream_start
data: {"sessionId": "session-550e8400-e29b-41d4-a716-446655440000"}

event: assistant-response.start
data: {}

event: server-tool-use
data: {"toolName": "analyze_pipeline", "parameters": {}}

event: server-tool-response
data: {"toolName": "analyze_pipeline", "result": {"opportunitiesAnalyzed": 47,
 "atRisk": 5}}

event: assistant-response.delta
data: {"text": "I analyzed your pipeline of 47 opportunities and identified "}

event: assistant-response.delta
data: {"text": "5 that are at risk of slipping:\n\n"}

event: assistant-response.delta
data: {"text": "1. **O2345678901** - Close date is past due by 15 days\n"}

event: assistant-response.completed
data: {"status": "complete"}
```

```
event: stream_end
data: {}
```

getSession

Retrieve the current state of a conversation session, including full conversation history, events, and metadata. Use this to inspect session state, review past interactions, or resume a conversation.

Parameters

- `sessionId` (required) — UUID of the session to retrieve. Format: `session-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.
- `catalog` (required) — Environment the session belongs to.

  Valid values: `"AWS"`, `"Sandbox"`

Response

Field	Type	Description
`sessionId`	string	Session identifier
`createdAt`	string	ISO 8601 timestamp of session creation
`lastActivity`	string	ISO 8601 timestamp of last activity
`sequenceNumber`	integer	Current event sequence number
`stateType`	string	Current session state
`events`	array	Full conversation history (user messages, agent responses, tool uses)

Field	Type	Description
variables	object	Session variables and metadata
eventCount	integer	Total number of events in the session

Example

Request:

```
{
    "jsonrpc": "2.0",
    "id": 7,
    "method": "tools/call",
    "params": {
        "name": "getSession",
        "arguments": {
            "sessionId": "session-550e8400-e29b-41d4-a716-446655440000",
            "catalog": "AWS"
        }
    }
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "id": 7,
    "result": {
        "content": [
            {
                "type": "text",
                "text": "{\"sessionId\":\"session-550e8400-e29b-41d4-
a716-446655440000\",\"createdAt\":\"2026-01-15T10:30:00Z\",\"lastActivity\":
\"2026-01-15T11:45:00Z\",\"sequenceNumber\":8,\"stateType\":\"END_TURN\",\"eventCount
\":8,\"events\":[...],\"variables\":{}}"
            }
        ]
    }
```

```
}
```

Error handling

All errors follow the JSON-RPC 2.0 error format:

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "error": {
        "code": -32001,
        "message": "Authentication failed. Verify your SigV4 credentials and ensure
 they have not expired."
    }
}
```

See the section called “Error codes” for the complete list of error codes and their meanings.

Recommended retry strategy

- For `-32004` (LIMIT_EXCEEDED): Retry with exponential backoff starting at 1 second
- For `-32603` (INTERNAL_ERROR): Retry up to 3 times with exponential backoff
- For `-32001` (AUTHENTICATION_FAILURE): Refresh credentials and retry
- For all other errors: Do not retry automatically — inspect the error message and correct the request

About the AWS Partner Central APIs

The following sections provide general information on the APIs.

Topics

Using the AWS Partner Central API

The following sections provide information about using the AWS Partner Central APIs.

Topics

Using the AWS Partner Central Account API

Managing partner account

Partner accounts are registered and linked with an existing AWS account to provide a full IAM-based experience. This approach eliminates the need for user-level credentials, ensuring that all registrations and operations are managed through IAM entity within the AWS account. AWS Partner Central enables seamless integration with AWS services, and this model provides partner

entity management, supports multiple catalogs, and establishes an AWS account-level entitlement system.

Partners can utilize the available partner account APIs to register, manage, and update their accounts:

Partner Registration API

Partners can register their account using their AWS account for Partner Engagement. Using the Create API, partners will provide required alliance lead information, accept APN terms, and provide a unique legal name.

Partner Profile API

Partners manage profiles containing company details (name, description, website, logo) with public/private visibility control, asynchronous validation, and status tracking allowing one update at a time.

Domain Management API

Partners can register and verify business domains through email validation to establish organizational identity and associate employees' training and certifications.

Partner Connection API

Partners connect their AWS accounts with other AWS accounts for various purposes, such as collaborating on co-selling with other partners, sharing AWS Marketplace Offers data between accounts, authorizing distributors/channel partners to distribute/re-sell their products, and more.

Partner Verification API

Partner verification is a mandatory prerequisite for partner account registration. Before partners can create a partner account, they must complete business verification and identity verification processes. These verifications validate that the business is legally registered and confirm the identity of the person registering the AWS account.

Topics

- Working with Partner Verification

Working with Partner Registration

Partner Registration

Partners can register their account using their AWS account for Partner Engagement. Using the Create API, partners will provide required alliance lead information, accept APN terms, and provide a unique legal name.

During Registration

1. **Synchronous Partner Entity Creation** – Customers initiate the partner registration process by calling the Create API, which performs input validation and creates the Partner Entity within the same request.
2. **Alliance Lead Information Requirement** – During partner registration, customer must provide alliance lead contact information for AWS Partner Network (APN) related communication.
3. **Terms & Conditions Enforcement** – Customers must accept the APN Terms and Conditions during registration. Acceptance is required to complete the registration and access any features. The terms apply to all APN program benefits and functionalities.
4. **Tag-On-Create Support** – As part of AWS Consistent Authorization Experience (CAE) and Tag-Based Authorization, customers are able to add tags at the time of partner entity creation.
5. **Legal Name-Based Validation** – Registration will enforce uniqueness based on partner legal name, ensuring no duplicate registrations under the same entity.

Post Registration

1. **Tag Management Support** – After registration, partners are able to tag, un-tag, and list tags for an existing partner entity.
2. **Fetch Partner Entity** - After a partner entity is created, customers can use List API to retrieve the Partner ID and then use the Get API to fetch the details of a specific partner entity.

API Summary

1. **CreatePartner API:** Customers initiate the partner registration process by calling the CreatePartner API, which performs input validation and creates the Partner Entity within the same request.

2. **ListPartners API:** After registration, customers can use the List API to retrieve list of partner entities.
3. **GetPartner API:** After registration, customers can use the Get API to retrieve details of a specific partner entity.
4. **PutAllianceLeadContact API:** Updates the primary alliance lead contact information. This operation allows partners to transfer primary alliance lead designation or modify contact details while maintaining organizational continuity.
5. **GetAllianceLeadContact API:** Updates the primary alliance lead contact information. This operation allows partners to transfer primary alliance lead designation or modify contact details while maintaining organizational continuity.

Working with Partner Profile

Profile management

Partners manage profiles containing company details (name, description, website, logo) with public/private visibility control, asynchronous validation, and status tracking allowing one update at a time.

1. **Profile Information** – Partner profiles must include essential company details such as display name, description, website URL, logo, IndustrySegments and PrimarySolutionType.
2. **Multilingual Support** – Partner profiles support multiple languages. Each localized version of the profile contains an attribute for locale.
3. **Profile Visibility Control** – Partners can configure their profile visibility (public or private).
4. **Profile Storage** – Profile information will be stored within the partner account object.
5. **Profile Status Management** – Profile updates are validated asynchronously before publishing. Partners can check the latest profile updating status (IN_PROGRESS, SUCCEED, FAILED and CANCELLED). Only approved profiles become publicly visible.
6. **Request Concurrency Control** – Only one profile update request is allowed at a time. If a profile update is already in progress, customers must explicitly cancel the ongoing update via the CancelPartnerProfileUpdateTask API before initiating a new one.
7. **Cancellation of Update Tasks** – Partners can cancel an in-progress profile update by calling the CancelPartnerProfileUpdateTask API. Cancellation is only allowed while the update is in the IN_PROGRESS status and must be completed before starting a new update.

API Summary

1. **StartPutProfileTask API:** This API accepts partner profile updates and performs basic synchronous input validation.
2. **GetProfileUpdateTask API:** This API allows Partners to fetch the current profile update status. It is intended to be used after customer start a profile update via StartPutPartnerProfile API, enabling Partners to check whether their request is IN_PROGRESS, FAILED, SUCCEEDED or CANCELLED.
3. **CancelProfileTask API:** Allows a partner to explicitly cancel a profile update task that is currently in the IN_PROGRESS status. This includes both auto-vetting and manual review phases. Canceling a request enables the partner to initiate a new profile update without waiting for the current vetting process to complete.
4. **PutProfileVisibility API:** By calling this API, partners can control the visibility of a profile regardless of locale. This allows partners to make profiles public or private without modifying the content.
5. **GetProfileVisibility API:** By calling the UpdatePartnerProfileVisibility API, partners can control the visibility of a profile regardless of locale. This allows partners to make profiles public or private without modifying the content.

Working with Domain Management

Domain Management

Partners can register and verify business domains through email validation to establish organizational identity and associate employees' training and certifications.

API Summary

1. **SendEmailVerificationCode API:** Initiates email verification process by sending a verification code to the specified email address. Used for both new contact creation and email address updates.
2. **AssociateAwsTrainingCertificationEmailDomain API:** Associates an email domain with AWS training and certification for the partner account, enabling automatic verification of employee certifications.
3. **DisassociateAwsTrainingCertificationEmailDomain API:** Removes the association between an email domain and AWS training and certification for the partner account.

Working with Partner Account Connections

Partners can manage connections between their own accounts or connections with other partners' AWS accounts using the AWS Partner Connection API. The process involves two phases:

1. **Connection Invitation Phase:** Initiating and responding to connection requests
2. **Active Connection Phase:** Managing established connections and their associated business activities

Creating and Sending a Connection Invitation

Step 1: Create a Connection Invitation

The first step in establishing a partner connection is creating and sending a connection invitation using the `CreateConnectionInvitation` action. When creating an invitation, you must specify:

- **Catalog:** The AWS catalog (AWS or Sandbox) where the connection will be managed
- **ConnectionType:** The type of relationship you want to establish. Choose one of the following:
  - **`OPPORTUNITY_COLLABORATION`:** Use this type when you want to send a connection request to another partner's account so that you can send them opportunity engagement invitations for opportunity collaboration
  - **`SUBSIDIARY`:** Use this type when you want to connect multiple AWS Marketplace seller accounts that you own to your "primary" account that you have linked with APN or where you are registered as a Partner
- **ReceiverIdentifier:** The receiver's public partner profile ID (for Opportunity Collaboration connection type) or AWS account ID (for Subsidiary connection type)
- **Email:** Your contact email address for communication
- **Name:** Your name as the sender
- **Message:** A description explaining the purpose of the connection request

When created, the invitation enters the Pending state, awaiting action from the receiver.

> **Important**
>
> By sending a connection invitation, you consent to revealing your AWS account ID to the receiver if they accept the invitation. This disclosure is necessary to establish the account-level connection.

Step 2: Monitor Your Sent Invitations

You can track the status of invitations you've sent using the `ListConnectionInvitations` action with the `ParticipantType` parameter set to `SENDER`. This allows you to:

- View all outgoing invitations
- Filter by status (`PENDING`, `ACCEPTED`, `REJECTED`, `EXPIRED`)
- Filter by connection type
- Check specific invitations to other partners

Step 3: Cancel an Invitation (Optional)

If you need to withdraw a pending invitation before it's been accepted, you can use the `CancelConnectionInvitation` action. Note that:

- You can only cancel invitations in the Pending state
- Once an invitation is accepted and a connection is established, it cannot be canceled
- To end an established connection, you must use the `CancelConnection` action instead

Receiving and Responding to Connection Invitations

Step 1: List Incoming Invitations

To view connection invitations you've received, use the `ListConnectionInvitations` action with the ParticipantType parameter set to `RECEIVER`. You can filter by:

- Connection type
- Status (`PENDING`, `ACCEPTED`, `REJECTED`)
- Sender's identifier

Step 2: Review Invitation Details

Use the `GetConnectionInvitation` action to view complete details of a specific invitation, including:

- Sender's name and contact email
- Invitation message explaining the purpose
- Connection type being requested
- Expiration date
- Sender's public profile information

Step 3: Accept or Reject the Invitation

After reviewing the invitation details, you have two options:

Option A: Accept the Invitation

Use the `AcceptConnectionInvitation` action to establish the connection. When you accept:

- A new Connection is created in the Active state
- The invitation's status changes to Accepted
- Your AWS account ID is revealed to the sender
- You can begin conducting business activities enabled by the connection type

> **Important**
>
> **Important Consent:** By accepting a connection invitation, you consent to revealing your AWS account ID to the sender. This disclosure is necessary to establish the account-level connection.

Option B: Reject the Invitation

Use the `RejectConnectionInvitation` action if you don't wish to establish the connection. You can optionally provide a reason for rejection. Once rejected:

- The invitation's status changes to Rejected

- No connection is established
- The invitation will no longer appear in your list of pending invitations

Automatic Expiration

If no action is taken within the expiration period, the system automatically transitions the invitation to the Expired state. Expired invitations cannot be accepted or rejected.

Managing Active Connections

Viewing Your Connections

Once a connection is established, both parties can view and manage it using the following actions:

`ListConnections`: Retrieve a list of your connections with filtering options:

- Filter by connection type and status (e.g., `OPPORTUNITY_COLLABORATION:ACTIVE`)
- Filter by other party's identifier
- View summary information for each connection

`GetConnection`: Retrieve complete details of a specific connection, including:

- All connection types associated with this partner relationship
- Status of each connection type (Active or Terminated)
- Contact information from the original invitation
- AWS account IDs and public profile IDs of both parties
- Creation and termination timestamps for each connection type

Adding Connection Types to an Existing Connection

If you already have an active connection with a partner and want to establish a new type of relationship, you can send a new connection invitation with a different connection type. Upon acceptance:

- The new connection type is added to the existing connection
- Both connection types remain independently manageable
- The same connection ID is maintained for the relationship

> **Note**
>
> There can only be one active connection of any given type between two accounts in a given catalog at any time.

Terminating Connection Types

Either party can terminate a specific connection type using the `CancelConnection` action. When terminating:

- Specify the connection ID and the connection type to terminate
- Provide a reason for the termination
- The connection type's status changes to `CANCELED`
- Other connection types within the same connection remain unaffected

Complete Connection Termination

When all connection types within a connection are terminated, the connection itself transitions to the Terminated state. A completely terminated connection:

- Cannot be reactivated, but you can send a new connection request to continue business with that account
- Remains in the system for historical record-keeping
- Can be viewed using the `GetConnection` API
- Requires a new invitation to re-establish the relationship

Monitoring Connection Events

Partners should monitor connection-related events using Amazon EventBridge to stay informed of:

- New incoming connection invitations
- Status changes to sent invitations (`ACCEPTED`, `REJECTED`, `EXPIRED`)
- Connection terminations initiated by the other party

Upon receiving an event, use the appropriate Get API action to fetch the latest details:

- `GetConnectionInvitation` for invitation updates
- `GetConnection` for connection updates

Possible Events

- **Connection Invitation Created:** Notifies the recipient account of incoming connection invitations.
- **Connection Invitation Cancelled:** Notifies the recipient account when a sender cancels an incoming connection invitation.
- **Connection Invitation Expired:** Notifies both sender and recipient accounts when a connection invitation expires without action.
- **Connection Invitation Rejected:** Notifies the sender account when their connection invitation is rejected by the recipient.
- **Connection Invitation Accepted:** Notifies the sender account when their connection invitation is accepted and a connection is established.
- **Connection Cancelled:** Notifies both connected parties when all a connection is fully terminated.

Understanding Connection States

Connection Invitation States

State	Description	Can Transition To
`PENDING`	Initial state when created; awaiting receiver action	`ACCEPTED`, `REJECTED`, `EXPIRED`, `CANCELED`
`ACCEPTED`	Receiver accepted; connection created	None (terminal state)
`REJECTED`	Receiver declined; contains optional rejection reason	None (terminal state)
`EXPIRED`	System-expired due to no action within expiration period	None (terminal state)

State	Description	Can Transition To
CANCELED	Sender withdrew the invitation before acceptance	None (terminal state)

Connection Type States

State	Description
ACCEPTED	Connection type is operational; associated business activities enabled
CANCELED	Connection type ended by either party

Best Practices

1. **Clear Communication:** Provide detailed, clear messages in your connection invitations explaining the purpose and expected collaboration.
2. **Timely Responses:** Monitor and respond to connection invitations promptly to avoid automatic expiration.
3. **Regular Review:** Periodically review your active connections to ensure they remain relevant to your current business needs.
4. **Proper Termination:** When ending a business relationship, use the CancelConnection action with a clear reason to maintain professional communication.
5. **Event Monitoring:** Set up EventBridge rules to automatically receive notifications of connection status changes, to help stay informed of important updates.
6. **Connection Type Management:** Consider which connection types are needed before establishing connections, as each type enables different business capabilities.
7. **Security Awareness:** Remember that establishing a connection reveals AWS account IDs between parties. Only connect with trusted partners.

Connection Types and Their Purposes

Connection Type	Purpose	Use Case
`OPPORTUNITY_COLLABORATION`	Enables collaboration on co-selling opportunities with other partners	When you want other partner to have access to your opportunity or vice versa, use this connection type. This connection allows you to send opportunity engagement invitations to connected partners.
`SUBSIDIARY`	Establishes connections between your multiple AWS Marketplace seller accounts and your primary Partner account	Use when you have multiple AWS Marketplace Seller Accounts you want to connect to your primary account where you are registered as a partner and seller.

Working with Partner Verification

Managing Partner Verification

Partner verification is a mandatory prerequisite for partner account registration. Before partners can create a partner account, they must complete business verification and identity verification processes. These verifications validate that the business is legally registered and confirm the identity of the person registering the AWS account.

Partners can utilize the available verification APIs to initiate and monitor verification processes:

During Verification

1. **Business Verification Initiation** – Partners initiate business verification by calling the `StartVerification` API with business registration details including legal name, registration ID, country code, and jurisdiction of incorporation.
2. **Identity Verification Initiation** – Partners initiate identity verification by calling the `StartVerification` API with registrant information. The API returns a secure completion

URL where the registrant completes the identity verification workflow using government-issued identification.

3. **Verification Status Monitoring** – Partners use the `GetVerification` API to retrieve the current status and details of verification processes. The API returns verification type, status, timestamps, and verification-specific details.
4. **Verification Completion** – Both business verification and identity verification must reach `SUCCEEDED` status before partners can proceed to partner account creation. Failed or expired verifications must be resolved before account registration.

Post Verification

1. **Partner Account Creation** – After successful verification, partners proceed to create their partner account using the `CreatePartner` API described in the Partner Account documentation.
2. **Verification Record Retrieval** – Partners can retrieve verification details at any time using the `GetVerification` API with the verification ID returned during initiation.

API Summary

1. **`StartVerification` API:** Partners initiate verification processes by calling the `StartVerification` API. For business verification, the API validates business registration details. For identity verification, the API generates a time-limited completion URL for the registrant to complete identity verification.
2. **`GetVerification` API:** After initiating verification, partners use the `GetVerification` API to retrieve verification status and details. The API returns current status, timestamps, and verification-specific information.

Using the AWS Partner Central Selling API

This AWS Partner Central API reference is designed to help AWS Partners integrate customer relationship management (CRM) systems with Partner Central. The API automates interactions with Partner Central, which helps to ensure effective engagements in joint business activities.

The API provides standard AWS API functionality. Access it by either using API Actions or by using an AWS SDK that's tailored to your programming language or platform. For more information, see Getting Started with AWS and Tools to Build on AWS.

Features offered by AWS Partner Central API

1. **Opportunity management:** Facilitates the management of coselling opportunities with AWS using API actions such as `CreateOpportunity`, `UpdateOpportunity`, `ListOpportunities`, `GetOpportunity`, and `AssignOpportunity`.
2. **AWS referral management:** Facilitates receiving referrals shared by AWS using actions like `ListEngagementInvitations`, `GetEngagementInvitation`, `StartEngagementByAcceptingInvitation`, and `RejectEngagementInvitation`.
3. **Entity association:** Associate related entities such as *AWS Products*, *Partner Solutions*, *AWS Marketplace Solutions*, *AWS Marketplace Products*, and *AWS Marketplace Private Offers* with opportunities using the actions `AssociateOpportunity` and `DisassociateOpportunity`.
4. **View AWS opportunity details:** Use the `GetAWSOpportunitySummary` action to retrieve real-time summaries of AWS opportunities that are linked to your opportunities.
5. **List solutions:** Provides list APIs for listing solutions partners offer using `ListSolutions`.
6. **Event subscription:** Partners can subscribe to real-time updates on opportunities by listening to events such as *Opportunity Created*, *Opportunity Updated*, *Engagement Invitation Accepted*, *Engagement Invitation Rejected* and *Engagement Invitation Created* using Amazon EventBridge.

Supported Regions and endpoints

The AWS Partner Central API is available in the US East (N. Virginia) Region.

Region	Endpoint
us-east-1	partnercentral-selling.us-east-1.api.aws

Partners can test and validate API integrations in a secure sandbox environment. This allows you to test API actions without affecting live data. For more information, see Testing in a sandbox for the AWS Partner Central Selling API.

Selling API entities

AWS Partner Central entities represent key business components used in coselling engagements between partners and AWS. These entities encapsulate information related to opportunities, solutions, products, and offers, enabling smooth collaboration and management of joint sales

activities. The following sections provide descriptions of the core entities in the Partner Central Selling API.

Opportunity

An opportunity is a potential sale or deal that a business identifies and actively pursues. It's a qualified prospect or lead with a specific need that the company's products or services can address. Opportunities are typically tracked in a sales pipeline or CRM system and form the foundation of future revenue. Effective opportunity management involves nurturing leads through the sales process, from the initial qualification to closing. For more information, see Working with your opportunities and Data types

Partner Solutions

Represents a Partner Solution (referred to as offering on AWS Partner Central), which is a software product or consulting practice created and delivered by AWS Partners. Partner Solutions help customers address specific business challenges or achieve particular goals using AWS services. For more information, see What is a solution?

AWS product

Represents a specific AWS service or product. AWS offers a wide range of products and services designed to provide scalable, reliable, and cost-effective infrastructure solutions. Partners can obtain the latest list of AWS Products from the bulk import page on Partner Central (Start Import > AWS Products and Solutions). For more information, you can learn about AWS Products or view the list of all AWS Products.

AWS Marketplace private offer

AWS Marketplace private offer is a feature that allows AWS Marketplace sellers to offer specific pricing and terms to individual AWS customers. Through private offers, sellers can negotiate custom prices, payment schedules, and end user license terms. AWS customers can obtain software solutions that meet their specific requirements, while also possibly benefiting from more favorable terms or pricing compared to standard offerings. The private offer process involves the seller creating a unique offer with tailored terms, which is then shared privately with the designated AWS customer for their review and acceptance. For more information, see Private offers in AWS Marketplace.

Engagement invitation

Engagement Invitation refers to a formal request from AWS for partners to collaborate on a specific referral. This allows AWS and the partner to work together to drive the opportunity forward. The invitation can be accepted or rejected by the partner.

Topics

Working with your opportunities

What is an Opportunity?

During the initial stages of the sales process, a sales representative assesses whether an interested individual (called *lead*) has the potential to become a customer. This assessment and validation phase is referred to as *Qualification*. Once a lead is deemed qualified and is considered to have a higher probability of converting to a customer, it is then classified as an *Opportunity*.

Working with Your Opportunities

Partners can manage opportunities created within their CRM systems and synchronize them with AWS Partner Central using the AWS Partner Central Selling API. This allows partners to track and manage opportunities from initiation to closure.

Creating an Opportunity

The first step in managing opportunities is creating an opportunity using the `CreateOpportunity` action. This creates an opportunity with the `Lifecycle.ReviewStatus` set to `Pending Submission`. However, the opportunity is not yet submitted to AWS for validation.

After the opportunity is created, you must associate at least one Partner Solution, AWS Marketplace Solution, or AWS Marketplace Product with the opportunity using the `AssociateOpportunity` action. This action clearly defines what the opportunity is attempting to sell. You can view the complete list of available solutions in your account using the `ListSolutions` API, and you can associate between one and ten solutions.

Optionally, you can also associate relevant AWS Products with the opportunity using the `AssociateOpportunity` action. This step helps AWS sales teams understand what AWS products are expected to be sold in conjunction with the opportunity.

After the required solution or product is associated and, optionally, AWS Products are linked, you can start engagement on the opportunity by using the `StartEngagementFromOpportunityTask` action. This is when you submit the opportunity to start an engagement.

Review Process

After starting the engagement on the opportunity using the `StartEngagementFromOpportunityTask` action, the opportunity enters the AWS validation phase, and its `Lifecycle.ReviewStatus` is set to `Submitted`. No changes can be made to the opportunity until the review process is complete.

During this validation phase, AWS ensures that the opportunity details are accurate and complete. While the validation is in progress, the `Lifecycle.ReviewStatus` is set to `In-Review`.

If there are changes or additional details required from the partner, AWS sets the `Lifecycle.ReviewStatus` to `Action Required`, and any required updates are communicated via the `Lifecycle.ReviewComments` field.

Once the opportunity passes validation, the `Lifecycle.ReviewStatus` changes to `Approved`, making the opportunity ready for co-selling activities.

Partners should monitor the `Opportunity Updated` event using Amazon EventBridge. This will notify them of any status changes or feedback from AWS. Upon receiving the event, partners can use the `GetOpportunity` API action to fetch the latest opportunity details and verify the `Lifecycle.ReviewStatus` field.

Resolving Validation Issues

If the `Lifecycle.ReviewStatus` is set to `Action Required`, partners need to address the issues highlighted by AWS. To resolve these, partners can update the opportunity using the `UpdateOpportunity` API action.

During the `Action Required` state, only certain fields are editable. These fields include:

1. `Customer.Account.Address.City`
2. `Customer.Account.Address.Country`
3. `Customer.Account.Address.PostalCode`
4. `Customer.Account.Address.StateOrRegion`
5. `Customer.Account.Address.StreetAddress`
6. `Customer.Account.WebsiteUrl`
7. `LifeCycle.TargetCloseDate`
8. `Project.ExpectedCustomerSpend.Amount`
9. `Project.ExpectedCustomerSpend.Currency`
10. `Project.CustomerBusinessProblem`
11. `PartnerOpportunityIdentifier`

After making the necessary changes, the opportunity re-enters the validation phase, and the process repeats until the opportunity's `Lifecycle.ReviewStatus` is set to `Approved` or `Disqualified`.

Post-Approval Updates

Once the opportunity is `Approved`, partners can continue to update the opportunity as needed using the `UpdateOpportunity` action, facilitating seamless co-selling activities.

Partners should continue monitoring the `Opportunity Updated` events through Amazon EventBridge to remain updated on any changes. For more information on tracking AWS updates, refer to the "Working with opportunity updates" section. Partners can also update select fields based on the business validation rules.

Working with opportunities from AWS

1. Receiving the AWS Opportunity

Opportunities are shared with partners when an AWS sales executive attaches a partner to an opportunity in AWS's CRM system. These are referred to as AWS Opportunities, distinct from opportunities created in the partner's own account.

When an AWS Opportunity has a partner attached, AWS creates an Engagement Invitation containing a subset of data from the AWS Opportunity. Partners will receive an *Engagement Invitation Created* event.

2. Reviewing the Engagement Invitation

The Engagement Invitation contains essential information such as `Project.Title`, `Project.CustomerUseCase`, `Lifecycle.Stage`, `Project.CustomerBusinessProblem`, and a few additional fields. Partners can use this data to decide whether to pursue the opportunity.

However, the following fields from the AWS Opportunity are not included in the Engagement Invitation:

```
Customer.Account.Address.StreetAddress
Customer.Account.Address.City
Customer.Account.Address.PostalCode
Customer.Contact
Customer.Account.AWSAccountId
Project.OtherSolutionDescription
RelatedEntityIdentifiers
```

3. Handling the Engagement Invitation

Upon receiving an *Engagement Invitation Created* event, partners can use the `GetEngagementInvitation` action to retrieve details of the AWS Opportunity.

If the partner decides not to pursue the opportunity, they can reject the invitation using the `RejectEngagementInvitation` action, along with the required `RejectionReason`. Once rejected, access to the opportunity is lost.

To accept the invitation and proceed, partners should use the `StartEngagementByAcceptingInvitationTask` action. This is an asynchronous action that sequentially performs the following tasks:

1. Accepts the Engagement Invitation.
2. Creates a new opportunity in the partner's account using data from the AWS Opportunity.
3. Includes additional details required to identify the customer in the partner's account.

Upon completion, an *Opportunity Created* event is triggered with the corresponding Opportunity ID. Partners can then use this ID in the `GetOpportunity` action to retrieve full details of the opportunity.

If the partner is not using events, they can call `StartEngagementByAcceptingInvitationTask` again with the same payload to check the latest status.

4. Managing the AWS Opportunity Post-Acceptance

Once the opportunity is in the partner's account, it can be managed and updated like any other opportunity in the system. Partners can use actions like `UpdateOpportunity` to make any necessary changes.

For more details on how to track AWS updates and manage opportunity updates, refer to the Working with opportunity updates section.

Working with opportunity updates

Updating opportunities

Partners can use the `UpdateOpportunity` action to update opportunities, with specific rules governing which fields are updated, and when they're updated:

1. Updates cannot be made if the `Lifecycle.ReviewStatus` is `Submitted` or `In-Review`.
2. Before submission, partners can make updates, but AWS will not process them unless the `StartEngagementFromOpportunityTask` action is invoked.
3. When the opportunity is in `Submitted` or `In-review` status, all updates are blocked.
4. If the opportunity is in `Action Required` status, AWS opens select fields for updates. These fields include:
   - `Customer.Account.Address.City`
   - `Customer.Account.Address.Country`
   - `Customer.Account.Address.PostalCode`
   - `Customer.Account.Address.StateOrRegion`

- `Customer.Account.Address.StreetAddress`
- `Customer.Account.WebsiteUrl`
- `LifeCycle.TargetCloseDate`
- `Project.ExpectedCustomerSpend.Amount`
- `Project.ExpectedCustomerSpend.CurrencyCode`
- `Project.ExpectedCustomerSpend.EstimationURL`
- `Project.ExpectedCustomerSpend.Frequency`
- `Project.ExpectedCustomerSpend.TargetCompany`
- `Project.CustomerBusinessProblem`
- `PartnerOpportunityIdentifier`

5. After the review process (i.e., when `Lifecycle.ReviewStatus` is set to `Approved`), the following fields cannot be updated:
   - `Customer.Account.Address.Country`
   - `Customer.Account.Address.PostalCode`
   - `Customer.Account.Industry`
   - `Customer.Account.WebsiteUrl`
   - `Project.CustomerBusinessProblem`
   - `PartnerOpportunityIdentifier`
   - `Project.Title`
6. For all other fields, updates can be made using the `UpdateOpportunity` action. However, additional restrictions may apply based on business rules for the specific program or opportunity type. For more details, refer to field-level validations.
7. For all updates made through both the UI and API, the *Opportunity Updated* event is generated.

Receiving updates from AWS on opportunities

AWS typically updates AWS opportunities, and each time an update is made, an *Opportunity Updated* event is generated.

To retrieve the latest updates from AWS, partners need to invoke two separate actions

1. `GetOpportunity` to retrieve details of the partner's opportunity.
2. `GetAWSOpportunitySummary` to retrieve real-time summaries of the AWS opportunity data.

Most regular updates from AWS will be available through the `GetAWSOpportunitySummary` response. However, AWS may occasionally update attributes in the partner's opportunity directly.

To consume updates from AWS

1. Invoke the `GetAWSOpportunitySummary` action to retrieve the latest details of the AWS Opportunity.
2. If changes need to be reflected in the partner's opportunity, use the `UpdateOpportunity` action to copy the relevant data onto the partner's opportunity.

Partners can choose to automate this process as a direct update mechanism or implement a manual review process to validate and update the data.

Working with multipartner opportunities

AWS Partners can work together on opportunities with AWS as an active participant.

Topics

Engagements and snapshots

An *engagement* is a resource owned by AWS for collaboration between multiple AWS Partner accounts and AWS on a customer opportunity. Engagements facilitate secure information sharing among partners and collaboration while maintaining individual opportunity ownership and control. Engagements encompass opportunities that originate from both AWS and partners, with AWS as an active participant. Partners can invite AWS or other partners to join an engagement using

`EngagementInvitations`. When invited partners accept, they receive their own opportunity within the same engagement.

Engagement members share progress through snapshots—point in time, immutable copies of specific fields from an underlying resource such as an opportunity. Snapshots are created within the engagement and shared with members. When the underlying resource changes, the owner can create a new snapshot revision to reflect updates. AWS provides snapshot jobs—customer-owned jobs that automatically create new revisions when the resource changes. Once shared, snapshots remain permanently accessible to engagement members.

Creating a custom policy for ResourceSnapshotJobRole

Collaborating on multi-partner opportunities requires sharing snapshots of your opportunities with other partners in an engagement. To maintain access to the latest opportunity details, you need to create a custom role called `ResourceSnapshotJobRole`. This role allows the system to create snapshots of your opportunities on your behalf and retrieve snapshots from other partners in the same engagement. Without this role, any updates you make to your opportunity will not be visible to other partners in the engagement, and they will continue to see outdated snapshots of your opportunity data.

> **Note**
>
> You can use a name other than `ResourceSnapshotJobRole`. If you use a different name, replace all instances of `ResourceSnapshotJobRole` in the following policies with your policy name.

To create this role, go to your IAM console and create the `ResourceSnapshotJobRole` role with the following custom trust policy:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "resource-snapshot-job.partnercentral-selling.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

After creating this role, you need to attach the AWSPartnerCentralSellingResourceSnapshotJobExecutionRolePolicy AWS managed policy, or attach the following permissions policy:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "partnercentral:CreateResourceSnapshot"
            ],
            "Resource": [
                "arn:aws:partnercentral:*::catalog/AWS/engagement/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "partnercentral:GetOpportunity"
            ],
            "Resource": [
                "arn:aws:partnercentral:*:111122223333:catalog/AWS/opportunity/*"
            ]
        }
    ]
}
```

Replace `{account}` with your AWS account ID.

After you create the `ResourceSnapshotJobRole` role and attach the permissions, you need to set the `ResourceSnapshotJobRole` for your organization. Use the PutSellingSystemSettings

action to set the role you created as the `ResourceSnapshotJobRole` for your company (identified by your AWS account):

```
aws-cli/2.13.5 Python/3.11.4 Linux/4.14.255-314-253.539.amzn2.x86_64
Host: partner-central.aws.amazon.com
Content-Type: application/json

{
  "ResourceSnapshotJobRoleIdentifier": "arn:aws:iam::{account}:role/
ResourceSnapshotJobRole"
}
```

Replace `{account}` with your AWS account ID, and `ResourceSnapshotJobRole` with the name of the role you created. This role will be implicitly assumed by the resource snapshot jobs to access your opportunities and create snapshots for sharing with other partners in the engagement.

Inviting partners to an opportunity

Partners can collaborate on opportunities that originate from both partners and AWS by initiating an engagement invitation. You can invite a maximum of nine partners to an opportunity.

To invite partners to an opportunity

1. Follow the steps in Finding and connecting with partners in the *AWS Partner Central Sales Guide*. You must connect with partners before you can invite them to opportunities. This connection grants mutual access to each other's account IDs, ensuring invitations reach the intended partner account.
2. Use the StartEngagementFromOpportunityTask action to create an engagement and associate an opportunity with it. This asynchronous action performs the following tasks sequentially:
    1. Creates an engagement.
    2. Associates the opportunity with the engagement.
    3. Invokes the CreateEngagementInvitation action to AWS.
    4. Submits the opportunity to AWS.
    5. Starts the `ResourceSnapshotJob` to create snapshots.
3. Invite other partners to join.

a. To get the engagement ID associated with your opportunity, use the ListEngagementFromOpportunityTasks action, filtered by the `TaskIdentifier` returned in the previous step.
b. Use the receiving partner's engagement ID and account ID to send an invitation with `CreateEngagementInvitation`.

A successful invitation sends an engagement invitation created event to the receiving partner.

Retrieving engagement invitation details

When you receive an engagement invitation created event, use the GetEngagementInvitation action to retrieve the invitation details. The response includes essential information about the customer opportunity associated with the engagement.

Review the invitation details, particularly the `InvitationMessage`, `Project.Title`, `Project.CustomerUseCase`, and `Project.CustomerBusinessProblem` fields. This information provides context about the customer opportunity and the inviting partner's expectations for collaboration.

Use these details to evaluate if you want to pursue the opportunity by accepting or declining the engagement invitation.

Responding to an engagement invitation

When a partner sends you an engagement invitation, it creates an engagement invitation created event that notifies you of the invitation. You have the option to accept or reject the invitation. This decision determines your involvement in the multipartner opportunity.

To reject an engagement invitation:

Use the RejectEngagementInvitation action to reject the invitation. You must provide a `RejectionReason` parameter explaining your decision. Once rejected, you lose access to the invitation details, and an engagement invitation rejected event notifies the sending partner that you have rejected their invitation.

To accept an engagement invitation:

To accept the invitation and proceed with the multipartner opportunity, use the StartEngagementByAcceptingInvitationTask action. This asynchronous action performs the following tasks sequentially:

1. Accepts the engagement invitation.
2. Creates a new opportunity in your partner account using data from the sending partner's opportunity. You will receive an opportunity created event.
3. Includes additional details required to identify the customer in your account.
4. Publishes an engagement invitation accepted event which notifies the partner that you have accepted the invitation and have been added to the engagement.

Expired engagement invitation

If you do not respond to an engagement invitation within fifteen days, it expires, and you cannot participate in the multipartner opportunity. An engagement invitation expired event notifies the sending partner that the invitation expired.

> **Note**
>
> If you still want to collaborate, the sending partner must initiate a new engagement invitation.

Reviewing and updating multipartner opportunities

When a partner initiates an engagement on an opportunity, the review status of the newly created opportunity in the receiving partner's account is determined by the sending partner's opportunity status.

Opportunity with submitted review status:

If the sending partner's opportunity has `Lifecycle.ReviewStatus` set to `Submitted`, the following process occurs:

1. The new opportunity created in the receiving partner's account will have `Lifecycle.ReviewStatus` set to `Submitted`.
2. The `Submitted` status remains until the sending partner's opportunity is `Approved`.
3. Once the sending partner's opportunity is validated and the `Lifecycle.ReviewStatus` is set to `Approved`, the other partners' opportunities will automatically inherit the `Approved` status.

This process ensures consistent opportunity details across multipartner opportunities, eliminating the need for multiple reviews.

Once complete, an opportunity created event is triggered with the corresponding opportunity ID. Partners can use this ID with the `GetOpportunity` action to retrieve the full opportunity details.

Opportunity with approved review status:

If the sending partner initiates an engagement on an opportunity with `Lifecycle.ReviewStatus` set to `Approved`, the following process occurs:

1. The new opportunity created in the receiving partner's account will have `Lifecycle.ReviewStatus` set to `Approved`.
2. An opportunity created event is triggered with the corresponding opportunity ID.
3. Partners can use the GetOpportunity action with the opportunity ID to retrieve the full opportunity details.

However, the approved opportunity may not have some partner-specific details that were not copied from the sending partner's opportunity. The receiving partner should update these details using the UpdateOpportunity action when the opportunity stage is updated to `Qualified` or a later stage:

- `Project.CustomerUseCase`
- `Project.DeliveryModels`
- `Solution`

- `PrimaryNeedsFromAws`
- `LifeCycle.TargetCloseDate`
- `Project.ExpectedCustomerSpend.Amount`
- `Project.ExpectedCustomerSpend.Currency`
- `Marketing.source`
- `Marketing.AwsFundingUsed`

Once the opportunity is created in the receiving partner's account, it can be managed and updated like any other opportunity within the partner's system. Partners can use the UpdateOpportunity action to make changes or provide more information about their involvement in the opportunity.

Using snapshots to receive partner updates

Within an engagement, partners maintain and update their opportunities independently. When someone revises an engagement's resources, such as adding an opportunity, an engagement resource snapshot created event is published.

To stay informed about changes and access the most current information from other partners' opportunities, you can use the following actions:

- ListEngagementResourceAssociations – Use this action to retrieve the engagement ID associated with the opportunity.
- ListResourceSnapshots – Use this action to retrieve a comprehensive list of all opportunity snapshots associated with the engagement, providing an overview of available snapshots across all partners.
- GetResourceSnapshot – Use this action to obtain real-time summaries of specific opportunity snapshots, allowing you to view the most up-to-date information without directly accessing another partner's opportunity.

If you identify relevant changes or updates from another partner's opportunity that should be reflected in your own, use the UpdateOpportunity action. This action facilitates selectively incorporating pertinent data into your opportunity, ensuring alignment and consistency across the engagement.

Viewing engagement members

An engagement on a multipartner opportunity can have up to ten partners. Whenever a new partner accepts the engagement invitation, an engagement member added event is published, notifying all current members about the change.

To view all members collaborating within an engagement, follow these steps:

1. Use the ListEngagementResourceAssociations action to retrieve the engagement ID associated with the opportunity.
2. Provide the ID from step 1 to the ListEngagementMembers action to fetch the partner details of engagement members.

> **Note**
>
> Only members of an engagement can invoke the `ListEngagementMembers` action.

Monitoring resource snapshot job status

When working with multipartner opportunities, you must create a role that allows you to create snapshots of your opportunities for other partners and retrieve opportunity snapshots from other partners in an engagement. Without this role, any updates you make to your opportunity will not be available to other partners in the engagement, and they will continue to see outdated snapshots of your opportunity.

To track the status of resource snapshot jobs associated with a multipartner opportunity in an engagement, follow these steps:

1. Use the ListEngagementResourceAssociations action to retrieve the engagement ID associated with the opportunity.
2. Provide the ID from step 1 to the ListResourceSnapshotJobs action to generate a list of all snapshot jobs owned by the caller in the engagement. Retrieve the job ID from the response.
3. Provide the job ID to the GetResourceSnapshotJob action to track the job status and see if it's running.

The `ResourceSnapshotJob` operation publishes metrics to Amazon CloudWatch for its asynchronous operations. CloudWatch processes the data into readable, near real-time metrics to help you monitor the performance and health of the service.

The following metrics are available:

- **Faults** – Counts internal issues during job execution. Use this metric to monitor service stability and set alarms for fault thresholds.
- **Errors** – Tracks customer-related issues during job processing. This metric helps identify problems with customer inputs or configurations.
- **Invocations** – Represents the total number of job invocations, including both successful and failed attempts. Use this to track service usage trends over time.

> **Note**
>
> Metrics are reported to CloudWatch only when there is activity in the `ResourceSnapshotJob` service. If there are no jobs or no data for a specific metric, that metric isn't reported.

To ensure smooth operation of the `ResourceSnapshotJob` operation:

- Use the metrics to verify that the service performs as expected.
- Create CloudWatch alarms to monitor metrics and trigger actions (such as sending notifications) when metrics exceed acceptable ranges.
- Set up proactive monitoring to Identify and resolve issues.

For more information about using CloudWatch metrics, see the Amazon CloudWatch User Guide.

Associating, disassociating and assigning opportunities

Opportunities can be associated or disassociated with Partner Solutions, AWS Products, AWS Marketplace Solutions, AWS Marketplace Products, AWS Marketplace Offers, and AWS Marketplace Offer Sets at any stage of the opportunity lifecycle.

Associating opportunities with other entities

The associated entities are retrieved from the `GetOpportunity` method within the `RelatedEntityIdentifiers` object. The `RelatedEntityIdentifiers` can be updated using `AssociateOpportunity`. Note that this field cannot be updated using the `UpdateOpportunity` or `CreateOpportunity` method.

Solutions

Before an engagement with AWS is started using the `StartEngagementFromOpportunityTask` action, it is mandatory to associate at least one and up to ten Partner Solutions. An AWS Referral may or may not contain a Partner Solution.

To view your existing solutions, use the `ListSolutions` action.

Partners can create, update, and manage their solutions in the `Build` section on AWS Partner Central.

Important

To progress the opportunity stage to *Committed* or *Launched*, you must associate a valid solution or AWS Marketplace product listing with the opportunity.

Note

Legacy Partner Solutions (`S-XXXX` identifiers) using the `Solutions` entity type are still supported. However, we recommend associating `AwsMarketplaceSolutions` entity type with the opportunities as the `Solution` entity type will be deprecated in the future.

AWS products

Up to 20 AWS Products can be associated with an opportunity. To view a list of available AWS Products, use the list of AWS Products hosted on GitHub. Association with AWS Products is exclusively done using the `AssociateOpportunity` action. To replace or remove a Product, use the `DisassociateOpportunity` action.

Solutions

Opportunities can be associated with solutions from your seller account. Use the `AssociateOpportunity` action with `RelatedEntityType` set to `AwsMarketplaceSolutions`.

The entity must be in *Public* or *Limited* status. The service rejects draft or inactive entities.

For associating a solution, an ARN is required. The following example shows the ARN format for a solution:

```
arn:aws:aws-marketplace:us-east-1:123456789012:AWSMarketplace/Solution/soln-
ab1c2de3fgh4i
```

You can also associate solutions from a subsidiary AWS Marketplace account connected to your primary account through Partner Account Connection. Enter the full ARN, including the subsidiary account ID.

The `ListSolutions` action returns `AwsMarketplaceSolutionArn` on each solution summary, and supports filtering by ARN.

AWS Marketplace Products

Opportunities can be associated with AWS Marketplace Products from your seller account. Use the `AssociateOpportunity` action with `RelatedEntityType` set to `AwsMarketplaceProducts`.

The entity must be in *Public* or *Limited* status. The service rejects draft or inactive entities.

For associating a product, an ARN is required. The ARN includes the product type (for example, `AmiProduct`, `SaaSProduct`, `ContainerProduct`). The following example shows the ARN format for an AMI product:

```
arn:aws:aws-marketplace:us-east-1:123456789012:AWSMarketplace/AmiProduct/prod-
ab1c2de3fgh4i
```

You can also associate products from a subsidiary AWS Marketplace account connected to your primary account through Partner Account Connection. Enter the full ARN, including the subsidiary account ID.

AWS Marketplace offers and offer sets

AWS Marketplace offers

Opportunities can be tied to an AWS Marketplace Private Offer. To view available offers, use the ListEntities from the AWS Marketplace Catalog API. Currently, you can only associate offers from the AWS Marketplace Seller account linked to AWS Partner Central.

For associating a private offer ARN is required. Sample:

```
arn:aws:aws-marketplace:us-east-1:123123123123:AWSMarketplace/Offer/offer-
dtn3example1tg
```

AWS Marketplace offer sets

Opportunities can also be associated with AWS Marketplace Offer Sets. Offer sets enable the grouping of multiple related marketplace offers together for comprehensive solution packaging. To view available offer sets, use the ListEntities from the AWS Marketplace Catalog API.

For associating an offer set, an ARN is required. Sample:

```
arn:aws:aws-marketplace:us-east-1:123123123123:AWSMarketplace/OfferSet/offerset-
dtn3example1tg
```

Important: An opportunity can be associated with either **one Offer** OR **one Offer Set**, but not both. Only one of these entity types can be associated with an opportunity at a time.

Disassociating opportunities from other entities

Use the `DisassociateOpportunity` action to unlink the opportunity from the following entity types:

- Solutions
- AWS Products
- AWS Marketplace Solutions
- AWS Marketplace Products
- AWS Marketplace Offers
- AWS Marketplace Offer Sets

Depending on the state of the opportunity, different validation rules apply when you unlink these related objects.

Assigning opportunities

Use `AssignOpportunity` to change the opportunity's owner. You can set any of your Partner Central users to be the opportunity owner.

Working with your leads

What is a Lead?

In business-to-business (B2B) sales, a lead represents a potential customer who has expressed interest in a company's products or services but has not yet been fully qualified as a sales opportunity. Leads are the initial stage of the sales pipeline and require nurturing and qualification before they can be converted into opportunities for active co-selling with AWS.

Leads shared by AWS with partners are based on customer engagement signals such as webinar attendance, campaign participation, or partner solution inquiries. These leads include valuable context like customer company information, business problems, and engagement details to help partners prioritize and qualify potential opportunities.

Working with Lead Invitations

Partners receive lead invitations from AWS when potential customers express interest in partner solutions or services. The lead invitation process allows partners to evaluate leads before committing to engagement, ensuring efficient resource allocation and higher conversion rates.

Receiving Lead Invitations

AWS creates lead invitations and shares them with partners through the Selling API. When a new lead invitation is available, AWS sends an Engagement Invitation with the Lead context to eligible partners.

Partners should monitor the `Engagement Invitation Created` event using Amazon EventBridge. This event notification includes the `payloadType` field set to `LeadInvitation`, allowing partners to distinguish lead invitations from opportunity invitations. Upon receiving the event, partners can retrieve invitation details to evaluate the lead.

Listing Lead Invitations

Partners can view all their lead invitations using the `ListEngagementInvitations` API action. This action retrieves invitations where the calling principal is either a sender or receiver.

To filter specifically for lead invitations, partners should use the `payloadType` filter with the value `LeadInvitation`. This filter ensures that only lead invitations are returned, excluding opportunity invitations from the results.

Lead invitations can be in the following states:

- **Pending**: Awaiting partner acceptance or rejection
- **Accepted**: Partner has accepted the invitation and gained access to full lead details
- **Rejected**: Partner has declined the invitation
- **Expired**: Invitation has passed its expiration date without action

Evaluating Lead Invitations

Before accepting a lead invitation, partners should retrieve detailed invitation information using the `GetEngagementInvitation` API action. This provides essential context for making informed accept or reject decisions.

The lead invitation payload includes:

Customer Information (available pre-acceptance):

- Company name, industry, and country
- Website URL
- Market segment (Enterprise, Large, Medium, Small, Micro)
- AWS maturity level (Evaluating, Single-Account, Multi-Account)

Customer Interactions:

- Source type and campaign information
- Customer action taken (form completion, webinar attendance, etc.)
- Business problem description provided by the customer
- Use case category
- Contact title (provides Authority context for BANT qualification)

> **Important**
>
> Customer contact information (name, email, phone number) is not visible at the invitation stage. Contact details become available only after accepting the invitation, ensuring customer privacy and preventing lead farming behavior.

Accepting Lead Invitations

When a partner decides to pursue a lead, they must accept the invitation using the `AcceptEngagementInvitation` API action. This action adds the partner to the engagement and grants access to full lead details, including customer contact information.

After acceptance:

- The partner is added as a member of the engagement
- Customer contact information becomes visible through the engagement Lead context
- The lead is classified as a Partner Qualified Lead (PQL) in AWS systems
- The lead appears in the partner's active leads list
- Partners can begin nurturing the lead and establishing customer contact

Partners can accept multiple lead invitations programmatically by calling `AcceptEngagementInvitation` for each invitation, enabling efficient bulk processing of high-quality leads.

Rejecting Lead Invitations

If a lead does not align with the partner's capabilities, capacity, or business focus, partners can decline the invitation using the `RejectEngagementInvitation` API action. When rejecting a lead invitation, partners should provide a rejection reason to help AWS improve lead routing and matching. Common rejection reasons include:

- Lack of geographic coverage
- Insufficient technical expertise for the use case
- Current capacity constraints
- Customer industry mismatch
- Budget misalignment

After rejection:

- The lead is removed from the partner's pending invitations queue
- Customer contact information is never shared with the partner
- The lead is classified as a Partner Rejected Lead (PRL) in AWS systems
- The invitation remains visible in the partner's rejected invitations list for reference

Rejected leads may be eligible for reassignment to other partners, provided customer consent requirements are met.

Managing Accepted Leads

After accepting a lead invitation, partners can access complete lead information, nurture customer relationships, and track the lead through qualification stages.

Viewing Lead Details

Partners access full lead details using the `GetEngagement` API action. The engagement contains a Lead context with comprehensive information about the customer and their interactions with AWS.

The Lead context includes:

Complete Customer Profile:

- All company information from the original invitation
- Full contact details for all customer interactions (name, email, phone, business title)
- AWS maturity level and market segment

Interaction History:

- Multiple customer touch-points consolidated into a single engagement
- Each interaction includes source information, customer action, business problem, and contact details
- Interactions are listed chronologically to provide a complete customer journey view

Qualification Status:

- Current state of lead qualification (Unqualified, Qualified, Disqualified, or custom status)
- Partners can update this status as they progress through their qualification process

This comprehensive view enables partners to understand the customer's complete engagement history across multiple AWS campaigns and touchpoints, rather than treating each interaction as a separate lead.

Listing Active Leads

Partners can retrieve all their active leads using the `ListEngagements` API action with the `contextType` filter set to `Lead`. This returns engagements where the partner is a member and the engagement contains a Lead context.

The list response includes key information such as:

- Engagement ID and title
- Customer company name
- Current engagement score
- Qualification status
- Number of interactions
- Creation and modification timestamps

Partners can use this list to build dashboards, track lead pipeline health, and identify leads requiring attention.

Enriching Leads with Prospecting Insights

Partners can enrich accepted leads with AWS insights to better prioritize and qualify them before investing in outreach. Enrichment is performed asynchronously through prospecting tasks, which augment a lead engagement with AWS-derived signals to support qualification decisions.

Starting a prospecting task

Partners initiate enrichment using the `StartProspectingFromEngagementTask` API action. This action accepts up to 100 engagement identifiers in a single request, allowing partners to enrich leads in batches. The task runs asynchronously and immediately returns a `TaskId` and `TaskArn` that partners use to track progress. The initial `TaskStatus` is one of `PENDING`, `IN_PROGRESS`, `COMPLETED`, or `FAILED`.

Partners can optionally provide a `TaskName` to label the task and a `ClientToken` to ensure idempotency across retries.

Polling for results

Because prospecting is asynchronous, partners poll for completion using the `GetProspectingFromEngagementTask` API action with the `TaskId` returned at start. The response reports the overall task status and a per-engagement result for each identifier submitted.

Each engagement is processed independently, so individual engagements can succeed or fail without affecting the others in the same task. For each engagement, the result includes:

- **Engagement identifier**: The engagement that was processed.
- **Status**: `PENDING`, `IN_PROGRESS`, `COMPLETED`, or `FAILED`.
- **Prospecting context identifier**: Populated when the engagement is processed successfully. This identifier references the enriched prospecting context added to the engagement and can be used in subsequent operations.
- **Reason code and message**: Populated only when an engagement fails, providing an enumerated failure code and a human-readable description with suggested recovery steps.

Monitoring multiple tasks

To track enrichment across many leads, partners use the `ListProspectingFromEngagementTasks` API action. This action lists all prospecting tasks initiated by the caller's account and supports optional filters by task identifier, task name, or start time range, with configurable sorting. The response is paginated; use the `NextToken` value from each response to retrieve subsequent pages.

When enrichment completes successfully, the AWS insights are added to the engagement as a prospecting context. Partners can retrieve the enriched details with `GetEngagement` and use them to set the lead's qualification status and decide which leads to advance toward conversion.

Updating Lead Information

As partners nurture leads and gather additional information, they can update lead details using the `UpdateEngagementContext` API action. This action allows partners to modify the Lead context within the engagement.

Partners can update:

Qualification Status: Partners should update the qualification status as they progress through their internal qualification process:

- **Unqualified**: Default status after accepting a lead; indicates the lead requires evaluation
- **Qualified**: Lead meets qualification criteria and shows high potential for conversion to an opportunity
- **Disqualified**: Lead does not meet criteria or is not a good fit for the partner's solutions
- **Custom States**: Partners can define their own qualification states to match their internal processes

Lead Metadata: Partners can add or update additional information relevant to their qualification and nurturing processes.

Updating qualification status helps partners track lead progression, generate accurate pipeline reports, and identify leads ready for conversion to opportunities.

Monitoring AWS Updates

AWS may update lead information based on new customer engagement signals or refined engagement scoring. When AWS updates the engagement, partners receive an `Engagement Updated` event via Amazon EventBridge.

These updates may include:

- Refined engagement scores based on new customer activity
- Additional customer interactions from new campaigns or touch-points
- Updated customer information

Partners should monitor these events and call `GetEngagement` to retrieve the latest lead details. This ensures partners have the most current information for prioritizing and nurturing leads.

The `Engagement Updated` event includes the `contextTypes` field, allowing partners to filter specifically for leads (engagements with Lead context).

Converting a Lead to an Opportunity

When a lead has been qualified and demonstrates serious buying intent, partners can convert it into an opportunity for formal co-selling collaboration with AWS. This conversion marks the transition from lead nurturing (primarily partner-driven) to opportunity management (collaborative with AWS).

Recommended Approach: Convenience API

Partners can streamline the opportunity creation and linking process by using the `StartOpportunityFromEngagementTask` convenience API action. This task API orchestrates multiple actions automatically:

1. Creates a draft opportunity with preliminary information from the lead context
2. Links the opportunity to the source engagement via resource snapshot
3. Adds the CustomerProject context to the engagement

This convenience method reduces the number of API calls required and ensures proper attribution tracking. Partners using this approach still need to:

- Complete opportunity details using `UpdateOpportunity`
- Associate Partner Solutions using `AssociateOpportunity`
- Submit the opportunity using `StartEngagementFromOpportunityTask` when ready

The convenience API is particularly useful for partners with automated lead-to-opportunity workflows who want to minimize integration complexity.

Alternative Approach: Step-by-step API

Creating a Draft Opportunity

The first step in converting a lead is creating a draft opportunity using the `CreateOpportunity` API action. This creates an opportunity with the `Lifecycle.ReviewStatus` set to `Pending Submission`. At this stage, the opportunity is not yet submitted to AWS for validation.

Partners should populate the opportunity with information gathered during lead nurturing, including:

- Customer account details
- Project information and business problem
- Expected customer spend
- Target close date
- Any other relevant details collected during qualification

The draft opportunity allows partners to prepare complete information before submitting to AWS, ensuring higher approval rates and faster validation.

Linking Opportunity to Lead Engagement

After creating the draft opportunity, partners must link it to the source lead engagement using the `CreateResourceSnapshot` API action. This step is critical for:

- **Attribution Tracking**: Establishes the provenance chain from lead invitation through opportunity closure, enabling accurate ROI calculation for marketing campaigns and lead sources.
- **Conversion Metrics**: Allows AWS and partners to measure lead-to-opportunity conversion rates and identify successful lead sources.
- **Context Preservation**: Maintains the complete customer journey history, including all original interactions and engagement signals.

When creating the resource snapshot, partners specify:

- The engagement ID (source lead engagement)
- The opportunity identifier
- The resource type (Opportunity)

This action automatically adds a CustomerProject context to the engagement, signaling that the lead has been converted to an active opportunity. The engagement now contains both the original Lead context (preserving history) and the new CustomerProject context (indicating active opportunity).

Completing Opportunity Details

Once the opportunity is created and linked, partners must complete additional opportunity requirements before submission. See the `AssociateOpportunity` API action for details.

Updating Project Details: Partners can use the `UpdateOpportunity` API action to refine project information, customer business problems, expected spend, and other relevant details gathered during lead qualification.

Submitting the Opportunity

Once all required information is complete, partners submit the opportunity for AWS validation using the `StartEngagementFromOpportunityTask` convenience API. This transitions the opportunity from draft status to the AWS review process.

Validation Requirement: The engagement must have a CustomerProject context before submission. This context is automatically created when using `CreateResourceSnapshot` to link the opportunity to the engagement. If this context is missing, the submission will fail.

After submission:

- The opportunity's `Lifecycle.ReviewStatus` changes to `Submitted`
- The lead is classified as a Partner Sales Qualified Lead (PSQL) in AWS systems
- AWS begins validation to ensure opportunity details are accurate and complete
- No changes can be made to the opportunity until the review process is complete

The opportunity then follows the standard validation workflow described in the "Working with your opportunities" documentation, progressing through states such as In-Review, Action Required, Approved, or Disqualified.

Working with deal sizing insights

What is Deal Sizing?

Deal Sizing is a capability within the APN Customer Engagements (ACE) Opportunities in AWS Partner Central that uses AI to provide automated deal size estimates and AWS service recommendations when Partners create or update opportunities.

AI forecasted MRR estimates

Deal sizing insights provide the median of the forecasted monthly recurring revenue (MRR) ranges based on Partners' historical AWS opportunities, use cases, and business descriptions. Partners should review and update the total MRR as Partners gather more information about the deal and progress through the sales cycle.

AI-Recommended AWS Products

AI-recommended products are identified based on customer business problem descriptions and opportunity details. The AI recommends relevant AWS products aligned with technical requirements and use cases. Partners should review these suggestions and customize the product selection to match the customer's specific needs.

Enhanced Insights with Pricing Calculator URLs

Partners can also import AWS Pricing Calculator URLs to automatically populate service selections and receive enhanced insights including Migration Acceleration Program (MAP) eligibility indicators, optimization recommendations, and potential cost savings analysis.

Understanding Source-Separated Data

Deal Sizing provides insights from two separate sources to give you comprehensive visibility into opportunity value:

- **AWS Source:** AI-recommended products based on patterns from similar historical opportunities
- **Partner Source:** Your own estimates imported from AWS Pricing Calculator

Handling Variance Between Sources

When partner-provided estimates differ significantly from AWS recommendations, Partners should:

1. Review opportunity details to ensure all relevant information is captured

2. Verify Pricing Calculator URL configurations match actual requirements
3. Consider AWS recommendations as data points informed by similar historical opportunities
4. Make informed decisions based on specific customer context and requirements

Accessing Deal Sizing Insights

Partners access Deal Sizing data through the `GetAwsOpportunitySummary` API action, which returns an extended `AwsOpportunitySummary` object containing deal sizing forecasts, product recommendations, and optimization insights.

When Deal Sizing Data Becomes Available

Deal Sizing insights become available after an opportunity is created with sufficient customer and project details. The system analyzes the opportunity attributes and generates:

1. **AI forecasted MRR:** Automatically calculated based on opportunity characteristics
2. **AWS product recommendations:** Services relevant to the customer's business problem and technical requirements
3. **Enhanced insights (when Pricing Calculator URL provided):** MAP eligibility, optimization recommendations, and cost savings analysis

> **Note**
>
> Deal Sizing insights are not available for all opportunities. Eligibility depends on factors such as partner history and the completeness of opportunity details provided.

Providing Deal Sizing Inputs

Partners provide inputs for Deal Sizing through the `CreateOpportunity` and `UpdateOpportunity` API actions. The system uses opportunity attributes to generate recommendations.

Providing Total Contract Value (TCV)

Partners who estimate in Total Contract Value rather than Monthly Recurring Revenue can provide their deal value directly. AWS automatically converts TCV to AWS MRR using AI-powered conversion models.

For deal sizing in TCV, provide `ExpectedContractDuration` as `Months` (valid values range from 1 to 144 months) and `TargetCompany` as `Self` and `Frequency` as None.

```
{
  "Project": {
    "ExpectedContractDuration": { "Term": "Months", "Value": "36" },
    "ExpectedCustomerSpend": [
      {
        "Amount": "1200000.00",
        "CurrencyCode": "USD",
        "Frequency": "None",
        "TargetCompany": "Self"
      }
    ]
  }
}
```

AWS derives the AWS MRR estimate and returns both entries on `GetOpportunity`:

```
{
  "Project": {
    "ExpectedContractDuration": { "Term": "Months", "Value": "36" },
    "ExpectedCustomerSpend": [
      {
        "Amount": "5600.00",
        "CurrencyCode": "USD",
        "Frequency": "Monthly",
        "TargetCompany": "AWS"
      },
      {
        "Amount": "1200000.00",
        "CurrencyCode": "USD",
        "Frequency": "None",
        "TargetCompany": "Self"
      }
    ]
  }
}
```

In above example the `ExpectedCustomerSpend` first array contains the AWS MRR estimate (`TargetCompany: "AWS"`, `Frequency: "Monthly"`) and second array is the partner's Total Contract Value (`TargetCompany: "Self"`, `Frequency: "None"`).

TCV Validation Rules

All TCV validation errors return error code `INVALID_VALUE` with `Reason: "BUSINESS_VALIDATION_FAILED"`. The distinguishing factor is the `FieldName` and `Message`:

Condition	Field	Message
Self entry without `ExpectedContractDuration`	`project.expectedContractDuration`	ExpectedContractDuration is required when a Self entry is present
`ExpectedContractDuration` without Self entry (Create only)	`project.expectedCustomerSpend`	A Self entry is required when ExpectedContractDuration is present
Multiple Self entries	`project.expectedCustomerSpend`	Only one Self entry is allowed
Invalid TargetCompany when duration present	`project.expectedCustomerSpend.targetCompany`	TargetCompany must be 'AWS' or 'Self'
Currency mismatch between AWS and Self	`project.expectedCustomerSpend.currencyCode`	CurrencyCode must match between entries
Self entry + EstimationUrl together	`project.expectedCustomerSpend.estimationUrl`	Self entry and EstimationUrl cannot coexist

Note

On `UpdateOpportunity`, if `ExpectedContractDuration` is present without a Self entry but partner deal value exists in storage, the system reconstructs the Self entry automatically — no error is thrown.

Switching Between TCV and Manual MRR

To switch from TCV mode back to manual MRR, explicitly set `ExpectedContractDuration` to `null` and provide only an `AWS` entry:

```
{
  "Project": {
    "ExpectedContractDuration": null,
    "ExpectedCustomerSpend": [
      {
        "Amount": "8000.00",
        "CurrencyCode": "USD",
        "Frequency": "Monthly",
        "TargetCompany": "AWS"
      }
    ]
  }
}
```

> **Note**
>
> Omitting `ExpectedContractDuration` from the payload (not sending the field) means "no change" — the existing TCV data is preserved. Explicitly sending `null` means "clear TCV mode."

TCV and Pricing Calculator URLs

If a partner provides both a `Self` entry (TCV mode) and an `EstimationUrl`, the API returns a validation error. Self entry and EstimationUrl cannot coexist in the same request.

Providing Manual MRR Estimates

Partners can manually enter MRR estimates using the `Project.ExpectedCustomerSpend[].Amount` field:

> **Note**
>
> The `CurrencyCode` field accepts USD and EUR. If the AWS Partition is `aws-eusc` (AWS European Sovereign Cloud), the currency code must be EUR.

```
{
  "Project": {
    "ExpectedCustomerSpend": [
      {
        "Amount": "25000.00",
        "CurrencyCode": "USD",
        "Frequency": "Monthly",
        "TargetCompany": "AWS"
      }
    ]
  }
}
```

This field is available to all Partners and can be used to accept AI forecasts by setting the same value or to override with Partner-provided values.

Providing Pricing Calculator URLs

Partners can optionally provide AWS Pricing Calculator URLs via the `Project.ExpectedCustomerSpend[].EstimationUrl` field to automatically import service configurations and receive enhanced insights including MAP eligibility indicators, optimization recommendations, and cost savings analysis.

```
{
  "Project": {
    "ExpectedCustomerSpend": [
      {
        "Amount": null,
        "CurrencyCode": "USD",
        "EstimationUrl": "https://calculator.aws/#/estimate?id=abc123",
        "Frequency": "Monthly",
        "TargetCompany": "AWS"
      }
    ]
  }
}
```

When a valid Pricing Calculator URL is provided, the system automatically calculates the MRR amount from the calculator configuration and populates both the Amount field and detailed product-level spend data in the Partner source of `AwsProductsSpendInsightsBySource`.

Partners should set Amount to null when supplying an EstimationUrl - the system will calculate it automatically.

URLs in invalid formats and URLs that cannot be resolved will be rejected with a `ValidationException`.

How Amount and EstimationUrl Work Together

You have flexibility in how you provide monthly recurring revenue (MRR) estimates for opportunities. The API intelligently handles various combinations of manual amounts and AWS Pricing Calculator URLs to ensure that your opportunities can always be created successfully.

Using only a manual amount

When you provide only the Amount field, the API saves your manual estimate and sets EstimationUrl to null. This approach is useful when you have a specific MRR estimate from customer conversations or your own calculations.

Example request:

```
{
    "Project": {
        "ExpectedCustomerSpend": [{
            "Amount": "25000.00",
            "CurrencyCode": "USD",
            "Frequency": "Monthly",
            "TargetCompany": "AWS"
        }]
    }
}
```

Using only an AWS Pricing Calculator URL

When you provide only the EstimationUrl field:

- **Valid URL** – The API calculates the MRR amount from your calculator configuration and saves both the URL and calculated amount.
- **Invalid URL** – The API returns a ValidationException with details about why the URL is invalid.

Example request:

```
{
    "Project": {
        "ExpectedCustomerSpend": [{
            "EstimationUrl": "https://calculator.aws/#/estimate?id=abc123",
            "CurrencyCode": "USD",
            "Frequency": "Monthly",
            "TargetCompany": "AWS"
        }]
    }
}
```

Example error response (invalid URL):

```
{
    "ErrorList": [{
        "Code": "INVALID_VALUE",
        "FieldName": "project.expectedCustomerSpend.estimationUrl",
        "Message": "Invalid Estimation URL: https://calculator.aws/#/estimate?
id=invalid"
    }],
    "Message": "The provided Estimation URL is invalid",
    "Reason": "INVALID_VALUE"
}
```

Providing both Amount and EstimationUrl

When you provide both fields, the API prioritizes ensuring that your opportunity is created successfully:

1. If both values are unchanged – no update is performed on these fields
2. URL is invalid – The API saves your provided Amount and sets EstimationUrl to null, allowing your opportunity creation to proceed.
3. URL is valid and calculated amount matches your provided Amount – Both values are saved.
4. URL is valid but calculated amount doesn't match your provided Amount – The API saves your provided Amount and sets EstimationUrl to null without returning an error.

> **Note**
>
> When both values don't match existing ones, the API first attempts to calculate the amount from the URL. Your manually provided Amount always takes precedence when there's a mismatch or validation issue.

Key benefit: Invalid URLs never block opportunity creation

This approach ensures that invalid or inaccessible AWS Pricing Calculator URLs never prevent you from creating or updating opportunities. Your manually provided Amount always takes precedence when there's a conflict or validation issue, giving you full control over your opportunity data.

Understanding the Extended AwsOpportunitySummary Data Model

This section describes the response structure returned by the `GetAwsOpportunitySummary` API action.

> **Note**
>
> AI-forecasted MRR is returned in `Project.ExpectedCustomerSpend[].Amount`

The `GetAwsOpportunitySummary` API action returns deal sizing insights through the new `Insights.AwsProductsSpendInsightsBySource` structure that provides comprehensive spend analysis with source attribution.

Source Separation: AWS vs Partner Data

Deal Sizing separates insights by source to provide transparency and prevent double-counting:

- **AWS Source:** AI product recommendations from AWS
- **Partner Source:** Spend data and product details imported from partner-provided Pricing Calculator URLs

This separation enables Partners to:

1. Compare their estimates against AWS recommendations

2. Validate their sizing assumptions
3. Avoid inflating totals by accidentally combining both sources
4. Maintain visibility into data provenance

Structure Overview

```
{
  "Insights": {
    "AwsProductsSpendInsightsBySource": {
      "<AWS | Partner>": {
        "CurrencyCode": "string",
        "Frequency": "string",
        "TotalAmount": "string",
        "TotalOptimizedAmount": "string",
        "TotalPotentialSavingsAmount": "string",
        "TotalAmountByCategory": { },
        "AwsProducts": [ ]
      }
    }
  },
  "Project": {
    "ExpectedCustomerSpend": [ ]
  },
  "RelatedEntityIds": {
    "AwsProducts": [ ]
  }
}
```

Field Reference

AwsProductsSpendInsightsBySource Map

Source-separated spend insights that provide independent analysis for AWS recommendations and partner estimates.

The `Insights.AwsProductsSpendInsightsBySource` field is a map containing up to two keys:

Key	Type	Description
AWS	AwsProductsSpendInsights	AI-generated insights including recommended products from AWS
Partner	AwsProductsSpendInsights	Partner-sourced insights derived from Pricing Calculator URLs including detailed service costs and optimizations

Note

Only sources with available data are included in the response. New opportunities typically start with only the AWS source populated.

AwsProductInsights Object

Comprehensive spend analysis for a single source (AWS or Partner) including total amounts, optimization savings, program category breakdowns, and detailed product-level insights.

Each source object contains the following fields:

Field	Type	Description
CurrencyCode	string	ISO 4217 currency code. Supported values are "USD" and "EUR".
Frequency	string	Indicates how frequently the customer is expected to spend the projected amount. Only the value Monthly is allowed for the Frequency

Field	Type	Description
		field, representing recurring monthly spend.
TotalAmount	string	Total estimated spend for this source before optimizations
TotalOptimizedAmount	string	Total estimated spend after applying recommended optimizations
TotalPotentialSavingsAmount	string	Quantified savings achievable through implementing optimizations
TotalAmountByCategory	object	Spend amounts mapped to AWS programs and modernization pathways
AwsProducts	array	Product-level details including costs and optimization recommendations

Current State Limitation: For the AWS source, TotalAmount, TotalOptimizedAmount, and TotalPotentialSavingsAmount may be omitted when per-product spend recommendations are not available. Partners should reference `Project.ExpectedCustomerSpend[].Amount` for the total AWS MRR estimate in these cases.

TotalAmountByCategory Object

Maps spend amounts to AWS programs and strategic initiatives:

Category Key	Description
MAP Eligible	Spend on services eligible for Migration Acceleration Program funding

Category Key	Description
CAT Eligible	Services supporting cost allocation and tracking
Pathway - Move to Cloud Native	Services aligned with cloud-native moderniza tion
Pathway - Move to Managed Services	Services supporting managed service adoption
Pathway - Move to Open Source	Open source service alternatives
Pathway - Move to Containers	Container-based service options
Pathway - Move to Serverless	Serverless computing services
Pathway - Move to Databases	Database modernization services
Pathway - Move to Analytics	Analytics and data processing services

Note

Total category amounts may exceed TotalAmount because individual products can belong to multiple categories.

AwsProducts Array

List of AWS services with program eligibility indicators (MAP, modernization pathways), cost estimates, and optimization recommendations.

Each product object contains:

Field	Type	Required	Description
ProductCode	string	Yes	AWS Partner Central product identifier used for opportunity association

Field	Type	Required	Description
ServiceCode	string	No	Pricing Calculator service code (links to original calculator URL)
Categories	array	Yes	List of program and pathway categorie s this product is eligible for
Amount	string	No	Baseline service cost before optimizations (may be null for AWS-sourced recommend ations)
OptimizedAmount	string	No	Service cost after applying optimizat ions (may be null for AWS-sourced recommendations)
PotentialSavingsAm ount	string	No	Service-specific cost reduction through optimizations (may be null for AWS-sourc ed recommendations)
Optimizations	array	No	List of specific optimization recommendations for this product

Null Handling: For AWS-sourced products, Amount, OptimizedAmount, and PotentialSavingsAmount fields may be null when per-product spend recommendations are not

available. The system provides total MRR via `Project.ExpectedCustomerSpend[0].Amount` instead.

Optimization Object

Each optimization recommendation contains:

Field	Type	Description
Description	string	Human-readable explanation of the optimization strategy
SavingsAmount	string	Quantified cost savings achievable by implementing this optimization

Examples

Example 1: AWS recommendations Only (Current State)

This example shows the typical response when only AI recommendations are available and per-product spend amounts cannot yet be recommended.

```
{
  "Catalog": "AWS",
  "Insights": {
    "AwsProductsSpendInsightsBySource": {
      "AWS": {
        "CurrencyCode": "USD",
        "Frequency": "Monthly",
        "AwsProducts": [
          {
            "ProductCode": "AmazonEC2",
            "Categories": ["MAP Eligible", "Cost Allocation Tagging"],
            "Amount": null,
            "OptimizedAmount": null,
            "PotentialSavingsAmount": null,
            "Optimizations": []
          },
          {
```

```
            "ProductCode": "AWSLambda",
            "Categories": ["Cost Allocation Tagging", "Pathway - Move to Cloud
 Native"],
            "Amount": null,
            "OptimizedAmount": null,
            "PotentialSavingsAmount": null,
            "Optimizations": []
          },
          {
            "ProductCode": "AmazonRDS",
            "Categories": ["MAP Eligible", "Cost Allocation Tagging", "Pathway - Move
 to Managed Services"],
            "Amount": null,
            "OptimizedAmount": null,
            "PotentialSavingsAmount": null,
            "Optimizations": []
          }
        ]
      }
    }
  },
  "Project": {
    "ExpectedCustomerSpend": [
      {
        "Amount": "28000.00",
        "CurrencyCode": "USD",
        "EstimationUrl": null,
        "Frequency": "Monthly",
        "TargetCompany": "AWS"
      }
    ]
  },
  "RelatedEntityIds": {
    "AwsProducts": [
      "AmazonEC2",
      "AWSLambda",
      "AmazonRDS"
    ]
  }
}
```


Key Points: AWS source shows recommended products with categories but no per-product amounts. Total MRR estimate available via `Project.ExpectedCustomerSpend[].Amount`.

Example 2: Partner Pricing Calculator Import

This example shows enhanced insights when a partner has provided a valid Pricing Calculator URL.


```
{
  "Catalog": "AWS",
  "Insights": {
    "AwsProductsSpendInsightsBySource": {
      "Partner": {
        "CurrencyCode": "USD",
        "Frequency": "Monthly",
        "TotalAmount": "32500.00",
        "TotalOptimizedAmount": "30000.00",
        "TotalPotentialSavingsAmount": "2500.00",
        "TotalAmountByCategory": {
          "MAP Eligible": "22500.00",
          "Cost Allocation Tagging": "32500.00",
          "Pathway - Move to Cloud Native": "5000.00",
          "Pathway - Move to Managed Services": "7500.00"
        },
        "AwsProducts": [
          {
            "ServiceCode": "ec2",
            "Categories": ["MAP Eligible", "Cost Allocation Tagging"],
            "Amount": "15000.00",
            "OptimizedAmount": "13500.00",
            "PotentialSavingsAmount": "1500.00",
            "Optimizations": [
              {
                "Description": "Convert to 3-year reserved instances",
                "SavingsAmount": "1000.00"
              },
              {
                "Description": "Migrate to Graviton-based instances",
                "SavingsAmount": "500.00"
              }
            ]
          },
          {
            "ServiceCode": "lambda",
            "Categories": ["Cost Allocation Tagging", "Pathway - Move to Cloud
 Native"],
            "Amount": "5000.00",
            "OptimizedAmount": "5000.00",
```

```
            "PotentialSavingsAmount": "0.00",
            "Optimizations": []
          },
          {
            "ServiceCode": "AmazonRDS",
            "Categories": ["MAP Eligible", "Cost Allocation Tagging", "Pathway - Move
 to Managed Services"],
            "Amount": "7500.00",
            "OptimizedAmount": "6500.00",
            "PotentialSavingsAmount": "1000.00",
            "Optimizations": [
              {
                "Description": "Optimize Multi-AZ for dev/test environments",
                "SavingsAmount": "1000.00"
              }
            ]
          },
          {
            "ServiceCode": "AmazonS3",
            "Categories": ["MAP Eligible", "Cost Allocation Tagging"],
            "Amount": "5000.00",
            "OptimizedAmount": "5000.00",
            "PotentialSavingsAmount": "0.00",
            "Optimizations": []
          }
        ]
      }
    }
  },
  "Project": {
    "ExpectedCustomerSpend": [
      {
        "Amount": "32500.00",
        "CurrencyCode": "USD",
        "EstimationUrl": "https://calculator.aws/#/estimate?id=abc123",
        "Frequency": "Monthly",
        "TargetCompany": "AWS"
      }
    ]
  },
  "RelatedEntityIds": {
    "AwsProducts": []
  }
```

```
}
```

Key Points: Partner source contains complete spend details with per-product amounts. Optimization recommendations provide actionable cost reduction strategies. Category breakdowns show MAP eligibility ($22,500 of $32,500 total). Products include ServiceCode linking back to Pricing Calculator configuration.

Example 3: Both Sources Present (Comparison Scenario)

This example demonstrates how both AWS recommendations and partner estimates appear together, enabling comparison.

```
{
  "Catalog": "AWS",
  "Insights": {
    "AwsProductsSpendInsightsBySource": {
      "AWS": {
        "CurrencyCode": "USD",
        "Frequency": "Monthly",
        "AwsProducts": [
          {
            "ProductCode": "AmazonEC2",
            "Categories": ["MAP Eligible", "Cost Allocation Tagging"],
            "Amount": null,
            "OptimizedAmount": null,
            "PotentialSavingsAmount": null,
            "Optimizations": []
          },
          {
            "ProductCode": "AWSLambda",
            "Categories": ["Cost Allocation Tagging", "Pathway - Move to Cloud
 Native"],
            "Amount": null,
            "OptimizedAmount": null,
            "PotentialSavingsAmount": null,
            "Optimizations": []
          },
          {
            "ProductCode": "EBS",
            "Categories": ["MAP Eligible", "Cost Allocation Tagging"],
            "Amount": null,
            "OptimizedAmount": null,
            "PotentialSavingsAmount": null,
```

```
            "Optimizations": []
          },
          {
            "ProductCode": "RDS",
            "Categories": ["MAP Eligible", "Cost Allocation Tagging", "Pathway - Move
to Managed Services"],
            "Amount": null,
            "OptimizedAmount": null,
            "PotentialSavingsAmount": null,
            "Optimizations": []
          }
        ]
      },
      "Partner": {
        "CurrencyCode": "USD",
        "Frequency": "Monthly",
        "TotalAmount": "32500.00",
        "TotalOptimizedAmount": "30000.00",
        "TotalPotentialSavingsAmount": "2500.00",
        "TotalAmountByCategory": {
          "MAP Eligible": "22500.00",
          "Cost Allocation Tagging": "32500.00",
          "Pathway - Move to Cloud Native": "5000.00",
          "Pathway - Move to Managed Services": "7500.00"
        },
        "AwsProducts": [
          {
            "ServiceCode": "ec2",
            "Categories": ["MAP Eligible", "Cost Allocation Tagging"],
            "Amount": "15000.00",
            "OptimizedAmount": "13500.00",
            "PotentialSavingsAmount": "1500.00",
            "Optimizations": [
              {
                "Description": "Convert to 3-year reserved instances",
                "SavingsAmount": "1000.00"
              }
            ]
          },
          {
            "ServiceCode": "lambda",
            "Categories": ["Cost Allocation Tagging", "Pathway - Move to Cloud
Native"],
            "Amount": "5000.00",
```

```
            "OptimizedAmount": "5000.00",
            "PotentialSavingsAmount": "0.00",
            "Optimizations": []
          },
          {
            "ServiceCode": "amazonRDS",
            "Categories": ["MAP Eligible", "Cost Allocation Tagging", "Pathway - Move
 to Managed Services"],
            "Amount": "7500.00",
            "OptimizedAmount": "6500.00",
            "PotentialSavingsAmount": "1000.00",
            "Optimizations": [
              {
                "Description": "Optimize Multi-AZ for dev/test environments",
                "SavingsAmount": "1000.00"
              }
            ]
          },
          {
            "ServiceCode": "amazonS3",
            "Categories": ["MAP Eligible", "Cost Allocation Tagging"],
            "Amount": "5000.00",
            "OptimizedAmount": "5000.00",
            "PotentialSavingsAmount": "0.00",
            "Optimizations": []
          }
        ]
      }
    }
  },
  "Project": {
    "ExpectedCustomerSpend": [
      {
        "Amount": "28000.00",
        "CurrencyCode": "USD",
        "EstimationUrl": "https://calculator.aws/#/estimate?id=abc123",
        "Frequency": "Monthly",
        "TargetCompany": "AWS"
      }
    ]
  },
  "RelatedEntityIds": {
    "AwsProducts": [
      "AmazonEC2",
```

```
      "AWSLambda",
      "EBS",
      "RDS"
    ]
  }
}
```

Key Points: Both sources present: AWS recommendations ($28K total) and Partner calculator ($32.5K total). AWS recommendations show 4 products; Partner calculator shows 4 products (3 overlapping). Partners can compare AWS recommendations against their detailed estimates. AWS total available via `Project.ExpectedCustomerSpend[].Amount`.

Best Practices

Handling Null Values

1. **EstimationUrl Field:** Expect EstimationUrl to be null for most Partners. This is standard behavior and should not be treated as an error. Display manual MRR entry options as the primary path.
2. **Missing Categories:** If TotalAmountByCategory is not present for the AWS source, this indicates per-product recommendations are not available. Category breakdowns are available for Partner-sourced data when Pricing Calculator URLs are provided.

Optimization Recommendations

1. **Display Actionable Insights:** Show optimization descriptions and savings amounts prominently to help Partners understand cost reduction opportunities.
2. **Aggregate Savings:** Sum PotentialSavingsAmount across products to show total optimization potential.
3. **Prioritize by Impact:** Sort optimizations by SavingsAmount to help Partners focus on highest-impact recommendations.

Monitoring for Updates

1. **Poll Periodically:** Poll `GetAwsOpportunitySummary` periodically (recommended: every 5 minutes) during opportunity creation workflows.
2. **Handle Async Generation:** Deal Sizing insights may not be immediately available after opportunity creation. Implement appropriate loading states and retry logic.

API Integration Patterns

1. **Leverage Existing Integrations:** Partners already calling `GetAwsOpportunitySummary` only need to consume additional fields in the response—no new API calls required.
2. **Graceful Degradation:** Design UIs to handle scenarios where Deal Sizing data is not yet available or incomplete.

Data Quality and Accuracy

1. **Validate Pricing Calculator URLs:** Ensure URLs are valid before submission to avoid null EstimationUrl responses.
2. **Provide Rich Opportunity Details:** More complete opportunity information (customer details, project description, technical requirements) yields more accurate AI recommendations.
3. **Review Variance:** When partner estimates differ significantly from AWS recommendations, review opportunity details to ensure all relevant context is captured.

Best practices

Reacting to events

When handling events from AWS Partner Central API, ensure that your processing logic is idempotent to handle duplicate events. Instead of making immediate GetOpportunity calls for each event, consider batching or selectively fetching details based on your application's needs. For uninterrupted operations, beware of Quotas.

Implementing optimistic locking

Optimistic locking prevents unintended data overrides during concurrent updates. Here's a typical scenario:

1. Partner retrieves an opportunity from their CRM system.
2. User A updates the opportunity on AWS Partner Central.
3. User B updates the same opportunity at the same time through the CRM integration.
4. If the data changes, the CRM system attempts to upload the data but returns a `ConflictException`.
5. User reviews the error and manually resolves conflicting data.

To avoid this scenario, all UpdateOpportunity requests must include the `LastModifiedDate` parameter, which you can obtain from previous CreateOpportunity, UpdateOpportunity, and GetOpportunity actions. The update succeeds only if `LastModifiedDate` matches our system. If it doesn't, you must fetch the latest `LastModifiedDate` using GetOpportunity and reattempt the update.

Synchronizing data between CRM and AWS Partner Central

It is essential to keep your system synced with the latest data from Partner Central. The following are two strategies to ensure your system reflects the latest data:

Using events (recommended)

1. Load data using ListOpportunities.
2. Subscribe to opportunity events.
3. Respond to new opportunities or changes.
   - When you receive `Opportunity Created`, `Opportunity Updated`, or `Engagement Invitation Accepted` events, use `GetOpportunity` to fetch the latest data.
   - When you receive `Engagement Invitation Rejected` events, remove the corresponding opportunities.

Using ListOpportunities polling

1. Load data using ListOpportunities.
2. Choose a polling frequency, ensuring it is not too frequent to avoid exhausting your daily read quota.
3. Identify the latest `LastModifiedDate` from your stored data, ensuring it originates from AWS.
4. Use the timestamp in the `AfterLastModifiedDate` filter when calling ListOpportunities.

```
{
   "FilterList": [
       {
           "Name": "AfterLastModifiedDate",
           "ValueList": [ "2023-05-01T20:37:46Z" ] // Replace with actual timestamp
 of your last synced data
        }
   ]
}
```

5. AWS will return opportunities created or updated after the value indicated on the timestamp.
6. Iterate over all returned pages using `NextToken`, and update your system's data using GetOpportunity.

```
{
   "NextToken": "AAMA-EFRSN...PZa942D",
   "FilterList": [
       {
           "Name": "AfterLastModifiedDate",
           "ValueList": [ "2023-05-01T20:37:46Z" ] // Replace with actual timestamp
 of your last synced data
        }
   ]
}
```

Using the AWS Partner Central Benefits API

The AWS Partner Central Benefits API streamlines how partners discover, apply for, and manage benefits across all AWS Partner Programs. By centralizing benefit management into a single, intuitive interface, the service creates a transparent meritocracy that rewards partners while simplifying operations.

The API provides standard AWS API functionality. Access it by either using API Actions or by using an AWS SDK that's tailored to your programming language or platform. For more information, see Getting Started with AWS and Tools to Build on AWS.

What is a Benefit?

In the AWS Partner Network (APN), a Benefit represents an advantage or capability that a partner may obtain from AWS to support their business growth and customer success. Benefits are designed to reward partners based on their qualifications and contributions while providing tangible value that helps partners scale their AWS practices.

Benefits can take many forms, including:

- **Financial incentives** - Credits, cash disbursements, and funding programs like Migration Acceleration Program (MAP) or Marketing Development Funds (MDF)
- **Access benefits** - Preview access to new services, tools, or specialized resources

- **Recognition** - Digital badges, certifications, competency designations, and partner tier status
- **Technical resources** - Solution architecture assistance, technical support, and ProServe engagements
- **Marketing support** - Co-marketing opportunities, featured partner status, and success story publications

Discovering Available Benefits

Partners begin their benefit journey by discovering what benefits are available to them and understanding eligibility requirements. The Benefits API provides comprehensive discovery capabilities through the Benefit Discovery APIs.

Listing Available Benefits

Partners can view all available benefits using the `ListBenefits` API action. This action retrieves a catalog of benefits with optional filtering capabilities to help partners find relevant opportunities quickly.

To efficiently discover benefits, partners can filter by:

- **Program** - Filter by specific AWS partner programs such as MAP (Migration Acceleration Program), MDF (Marketing Development Funds), ISV Workload Migration, Innovation Sandbox, Proof of Concept, or Partner Initiative Funding
- **Fulfillment Type** - Filter by how benefits are delivered: `CREDITS`, `CASH`, `DISCOUNT`, `ACCESS`, `RECOGNITION`, or `RESOURCE`
- **Status** - Filter by benefit status: `ACTIVE` or `DEPRECATED`

The `ListBenefits` API returns benefit summaries containing essential information such as benefit ID, name, description, associated programs, and fulfillment types. This summary view allows partners to quickly scan available benefits and identify those most relevant to their business objectives.

Example filtering approach:

Partners focused on financial incentives might filter by fulfillment types `CREDITS` and `CASH` to see funding opportunities. Partners seeking technical enablement might filter by fulfillment type `RESOURCE` or `ACCESS` to find technical support benefits and tool access.

Viewing Detailed Benefit Information

Once a partner identifies a benefit of interest, they can retrieve complete details using the `GetBenefit` API action. This provides comprehensive information including:

- **Eligibility criteria** - Detailed requirements that partners must meet to qualify for the benefit
- **Benefit request schema** - The specific information and documentation partners need to provide when applying
- **Grant details** - What partners will receive upon approval
- **Associated programs** - Which AWS partner programs the benefit supports

The benefit request schema is particularly important as it defines the structure and required fields for benefit applications. Partners should review this schema carefully before creating a benefit application to ensure they gather all necessary information upfront.

> **Important**
>
> Eligibility determination for each benefit is handled by the client or UI layer. For funding-related benefits, eligibility is typically based on the partner's scorecard performance and program participation.

Topics

Working with Benefit Applications

A Benefit Application models a partner's request for a specific benefit. It captures all information needed to evaluate and process the request according to benefit-specific conditions. The benefit application lifecycle progresses through multiple states from draft creation to final approval or rejection.

Creating Benefit Applications

Partners initiate the benefit request process by creating a benefit application using the `CreateBenefitApplication` API action. At creation, the application enters

PENDING_SUBMISSION status, allowing partners to prepare complete information before submitting for review.

When creating a benefit application, partners must provide:

- **Benefit Identifier** - The ID or ARN of the benefit being requested
- **Fulfillment Types** - How the partner expects to receive the benefit (CREDITS, CASH, DISCOUNT, ACCESS, RECOGNITION, or RESOURCE)
- **Benefit Application Details** - A JSON document containing benefit-specific information as defined by the benefit's request schema
- **Client Token** - A unique idempotency token to prevent duplicate submissions

Partners can optionally provide:

- **Name and Description** - Human-readable identifiers for the application
- **Partner Contacts** - Contact information for the person managing this benefit request (maximum 1 contact)
- **File Attachments** - Supporting documentation such as project plans, SOWs, proof of cost, or customer satisfaction surveys (maximum 10 files)
- **Associated Resources** - Links to related opportunities or existing benefit allocations (maximum 10 resources)
- **Tags** - Key-value pairs for resource organization and tracking (maximum 200 tags)

The CreateBenefitApplication API performs soft validation, checking only basic field types and patterns. Full business logic validation occurs during submission, allowing partners to save incomplete applications and return later to complete them.

Best Practice: Partners should use the benefit request schema from GetBenefit to understand exactly what information is required before creating an application. This reduces the likelihood of submission failures due to missing or invalid data.

Updating Benefit Applications

Partners can modify draft benefit applications using the UpdateBenefitApplication API action. This allows partners to refine information, add documentation, or correct errors before submission.

When updating a benefit application, partners must provide:

- **Identifier** - The ID or ARN of the benefit application to update
- **Revision** - The current revision number for optimistic locking
- **Benefit Application Details** - The complete, updated JSON document

Partners must submit the complete benefit application object, even if only specific fields are changing. The best practice is to first retrieve the latest application details using `GetBenefitApplication`, modify the necessary fields, then submit the full updated payload to `UpdateBenefitApplication`.

Optimistic Locking: The revision field ensures that updates are applied only if the application hasn't changed since it was last retrieved. If the revision doesn't match the current database value, the update is rejected with a conflict error. This prevents partners from accidentally overwriting changes made by other processes or systems.

Updates can only be performed while the application is in `PENDING_SUBMISSION` status. Once submitted, applications enter the review process and can no longer be updated through this API.

Associating Resources with Benefit Applications

Partners can link benefit applications to related resources using the `AssociateBenefitApplicationResource` API action. This creates valuable connections between benefits and the business context in which they're being used.

Supported resource types include:

- **OPPORTUNITY** - Link the benefit application to a specific customer opportunity in . This is particularly useful for opportunity-specific benefits like MAP funding or POC credits tied to customer engagements.
- **BENEFIT_ALLOCATION** - Link to existing benefit allocations, enabling partners to chain benefits or demonstrate how previous benefits are being leveraged

Resource association can only occur before submission. Once a benefit application is submitted (status changes to `IN_REVIEW`), no further associations are allowed. Partners can associate up to 10 resources with each benefit application.

To disassociate a resource, partners use the `DisassociateBenefitApplicationResource` API action. Like association, disassociation can only occur in `PENDING_SUBMISSION` status.

> **Important**
>
> Partners must have appropriate permissions to both access the benefit application and read the specific resource being associated. The API validates these permissions during the association operation.

Submitting Benefit Applications

When a benefit application is complete and ready for AWS review, partners submit it using the `SubmitBenefitApplication` API action. Submission triggers a state transition from `PENDING_SUBMISSION` to `IN_REVIEW` and initiates the benefit-specific approval workflow.

Upon submission, the API performs comprehensive validation including:

- **Required fields validation** - Ensures all mandatory fields in the benefit application details are present
- **Field format validation** - Verifies that field values match expected patterns and data types
- **Business rule validation** - Applies benefit-specific business logic and constraints
- **Resource validation** - Confirms that all associated resources are valid and accessible
- **File validation** - Verifies that all file attachments have completed processing successfully

If validation fails, the API returns a ValidationException with detailed error codes and messages indicating which fields require correction. Partners must address these issues using `UpdateBenefitApplication` before attempting submission again.

File Processing Requirement: Benefit applications cannot be submitted if any attached files are still in `PENDING` status. Partners must wait for file processing to complete (status changes to `SUCCEEDED`) before submitting. If files fail processing (status `FAILED`), partners should upload corrected versions.

After successful submission:

- The application status changes to `IN_REVIEW`
- The benefit owner's team is notified of the new submission
- Partners can no longer modify application details through standard update APIs

- The application enters a defined approval workflow which may include business approval, technical approval, and finance approval stages

Partners can track submission progress by retrieving the application and monitoring the Stage field, which indicates the current approval stage (e.g., "Business Approval", "Tech Approval", "Finance Approval").

Managing Submitted Applications

Once submitted, partners have limited options for managing benefit applications, but the API provides specific actions for common scenarios.

Recalling Benefit Applications

If a partner discovers an error or needs to make changes after submission, they can recall the application using the `RecallBenefitApplication` API action. Recall transitions the application back to `PENDING_SUBMISSION` status, allowing updates and resubmission.

When recalling an application, partners should provide:

- **Identifier** - The ID or ARN of the benefit application to recall
- **Reason** - An optional explanation for the recall (maximum 1000 characters)

The reason field enables better traceability and helps AWS understand common issues that lead to recalls, informing future improvements to the benefit application process.

After recall, partners can use `UpdateBenefitApplication` to make necessary changes, then resubmit using `SubmitBenefitApplication` when ready.

> **Important**
>
> Recall is typically only available during early review stages. Applications that have progressed to later approval stages or have been approved may not be eligible for recall.

Amending Benefit Applications

For minor corrections after submission, partners can use the `AmendBenefitApplication` API action to update specific fields without recalling the entire application. This is particularly useful when AWS reviewers request clarifications or corrections during the review process.

Amendments use a JSON Patch-style approach where partners specify:

- **Path** - A JSONPath expression identifying the field to update (e.g., `$.CreditDisbursementDetails.AwsAccountIdForCredits`)
- **Value** - The new value for the field
- **Operation** - The operation to perform (currently only `REPLACE` is supported)

Partners can submit up to 10 amendments in a single API call. Each amendment must include an updated revision number for optimistic locking.

Amendments should be used for minor corrections. For substantial changes, partners should use `RecallBenefitApplication` to return the application to draft status for comprehensive updates.

Canceling Benefit Applications

If a partner no longer needs a benefit or wishes to withdraw their application, they can cancel it using the `CancelBenefitApplication` API action. Cancellation transitions the application to `CANCELED` status, permanently ending the application process.

When canceling an application, partners should provide:

- **Identifier** - The ID or ARN of the benefit application to cancel
- **Reason** - An optional explanation for the cancellation (maximum 1000 characters)

Once canceled, applications cannot be reactivated. If the partner later decides they need the benefit, they must create a new benefit application.

Common reasons for cancellation include:

- Customer opportunity was lost or delayed
- Partner capacity constraints changed
- Business priorities shifted
- Benefit is no longer needed for the intended purpose

Viewing Benefit Application Details

Partners can retrieve complete information about a benefit application using the `GetBenefitApplication` API action. This provides a comprehensive view of the application including:

Application Metadata:

- Unique identifier (ID) and Amazon Resource Name (ARN)
- Associated benefit identifier
- Application name and description
- Current status (`PENDING_SUBMISSION`, `IN_REVIEW`, `ACTION_REQUIRED`, `APPROVED`, `REJECTED`, or `CANCELED`)
- Current processing stage

Status Information:

- Status Reason - Human-readable explanation of the current status
- Status Reason Codes - Structured codes indicating specific issues or requirements (e.g., "Checklist Incomplete", "Project Plan Missing", "Design Win Missing")

Application Content:

- Complete benefit application details JSON document
- Partner contact information
- File attachments with processing status
- Associated resources (opportunities or allocations)
- Resource tags

Audit Information:

- Creation timestamp
- Last modification timestamp
- Current revision number

The status reason codes are particularly valuable when an application is in `ACTION_REQUIRED` status. These codes provide specific, actionable guidance on what partners need to address for the application to proceed.

Listing Benefit Applications

Partners can view all their benefit applications using the `ListBenefitApplications` API action. This returns a paginated list of application summaries with powerful filtering capabilities.

Partners can filter applications by:

- **Program** - View applications for specific programs (MAP, MDF, Sandbox, POC, etc.)
- **Fulfillment Type** - Filter by delivery method (`CREDITS`, `CASH`, `ACCESS`, etc.)
- **Benefit Identifier** - View applications for a specific benefit
- **Status** - Filter by application status (`PENDING_SUBMISSION`, `IN_REVIEW`, `APPROVED`, `REJECTED`, `CANCELED`)
- **Stage** - Filter by approval stage (Business Approval, Tech Approval, Finance Approval)
- **Associated Resource ARNs** - Find applications linked to specific opportunities or allocations

The list response includes essential summary information such as:

- Application ID, ARN, and name
- Associated benefit ID
- Programs and fulfillment types
- Current status and stage
- Creation and modification timestamps
- Associated resources
- Current revision number
- Selected fields from benefit application details (as defined by the benefit owner)

Partners can configure result sorting using the Sort parameter, with options to sort by creation date, status, program, or benefit identifier in ascending or descending order.

Dashboard Building: The `ListBenefitApplications` API is designed to support partner dashboard creation. By filtering and sorting applications, partners can build views such as:

- Applications requiring action (`ACTION_REQUIRED` status)
- Recently submitted applications (sorted by creation date, `IN_REVIEW` status)
- Approved applications pending allocation (`APPROVED` status)
- Applications by program (filtered by specific programs)

Working with Benefit Allocations

A Benefit Allocation represents a benefit that has been granted to a partner. When a benefit application is approved, AWS creates one or more benefit allocations that provide partners with the actual credits, funding, access, or other benefit fulfillment.

Understanding Benefit Allocations

Benefit allocations can represent various types of fulfillment:

- **Financial Disbursements (CASH)** - Direct financial payments to partners with disbursement tracking
- **AWS Credits (CREDITS)** - Credit codes applicable to AWS accounts with usage tracking
- **Consumable Allocations** - Pre-approved funding amounts or wallets with utilization tracking
- **Access Grants (ACCESS)** - Access to systems, tools, or resources
- **Recognition (RECOGNITION)** - Digital badges, certifications, or status designations
- **Resources (RESOURCE)** - Technical support, advisory services, or ProServe engagements

Each benefit allocation has a lifecycle tracked through its Status field:

- **ACTIVE** - The allocation is available for use
- **INACTIVE** - The allocation is temporarily unavailable
- **FULFILLED** - The allocation has been completely used or delivered

Benefit allocations also include temporal information defining when they become effective (StartsAt) and when they expire (ExpiresAt), enabling partners to understand the timeframes for utilizing benefits.

Listing Benefit Allocations

Partners can view all their benefit allocations using the `ListBenefitAllocations` API action. This returns a paginated list of allocation summaries with comprehensive filtering capabilities.

Partners can filter allocations by:

- **Benefit Identifier** - View allocations for a specific benefit
- **Benefit Application Identifier** - View allocations resulting from a specific application
- **Fulfillment Type** - Filter by allocation type (`CREDITS`, `CASH`, `ACCESS`, etc.)
- **Status** - Filter by allocation status (`ACTIVE`, `INACTIVE`, `FULFILLED`)

The list response includes allocation summaries containing:

- Allocation ID, ARN, and name
- Associated benefit ID and benefit application ID (if applicable)
- Fulfillment type
- Current status
- Creation timestamp
- Start date

This list view enables partners to build allocation tracking dashboards and identify:

- Active allocations available for immediate use
- Allocations approaching expiration
- Fulfilled allocations for historical tracking
- Total allocation value by program or fulfillment type

Viewing Detailed Allocation Information

Partners can retrieve complete allocation details using the `GetBenefitAllocation` API action. This provides comprehensive information about the allocation including fulfillment-specific details that vary based on the allocation type.

Core Allocation Information:

- Unique identifier (ID) and Amazon Resource Name (ARN)
- Allocation name and status
- Status reason (explanation of current status)
- Associated benefit ID and benefit application ID
- Fulfillment type
- Creation, update, start, and expiration timestamps

Fulfillment Details (Type-Specific):

The FulfillmentDetail field contains different information structures based on the fulfillment type:

Cash Disbursement Details (CASH Type)

For direct cash payments, the fulfillment details include:

- **Disbursed Amount** - The total amount of funding disbursed, including amount value and currency code
- **Issuance Details (if applicable)** - Information about purchase orders or issuance events:
  - Issuance ID - Unique identifier for the disbursement
  - Issuance Amount - Amount in local currency
  - Issued At - Timestamp of disbursement

This structure supports both single disbursements and pre-approved funding scenarios where multiple issuances may occur against a single allocation.

Credit Details (CREDITS Type)

For AWS credits, the fulfillment details include:

- **Allocated Amount** - Total credit amount allocated
- **Issued Amount** - Total credit amount issued (may be less than allocated for phased issuance)
- **Credit Codes** - Array of individual credit code details:
  - AWS Account ID - Account where credit is applied
  - Value - Monetary value of the credit code
  - AWS Credit Code - The actual credit code string

- Status - Credit code status (`ACTIVE`, `INACTIVE`, `FULFILLED`)
- Issued At - When the credit code was issued
- Expires At - When the credit code expires

This detailed credit tracking allows partners to monitor credit utilization across multiple accounts and understand which credits are available, partially used, or fully consumed.

Consumable Allocation Details (Pre-Approved Amounts/Wallets)

For pre-approved funding pools, the fulfillment details include:

- **Allocated Amount** - Total amount in the allocation
- **Remaining Amount** - Unused amount still available
- **Utilized Amount** - Amount already consumed
- **Issuance Details (if applicable)** - Purchase order information for the allocation

Consumable allocations enable partners to draw down funds as needed for eligible activities, with real-time tracking of remaining balance.

Access Details (ACCESS Type)

For access grants, the fulfillment details include:

- **Description** - Detailed explanation of the access being granted and how to utilize it

Access allocations might provide access to preview services, specialized tools, or restricted resources. The description field provides instructions on how partners can activate and use the granted access.

Applying Allocations to New Benefit Applications

The ApplicableBenefitIds field in benefit allocations identifies other benefits that can leverage this allocation. This enables benefit chaining where partners can use existing allocations to apply for additional benefits.

For example, a partner might:

1. Receive a pre-approved MAP funding allocation

2. Create benefit applications for individual customer migration projects
3. Associate those applications with the pre-approved allocation
4. Draw down funds from the allocation as projects are approved

This approach streamlines the benefit application process for partners with pre-approved funding, reducing the review cycle for individual project requests.

Using the AWS Partner Central Channel API

AWS Partner Central Channel APIs enable you to programmatically manage your authorized AWS reselling business. These APIs allow AWS Solution Providers, AWS Distributors, and AWS Distribution Sellers to:

- Report AWS management accounts used for resale programs to Partner Central
- Send invitations to AWS accounts to accept partner associations
- Create relationships required to onboard resold end customer AWS accounts and apply partner discounts

With AWS Partner Central Channel APIs, you can build custom automation to:

- Onboard new partner-owned AWS accounts to APN resale programs
- Onboard new end customer accounts to resale programs
- Accelerate customer onboarding and discount qualification

Working with Program Management Accounts (PMAs)

A Program Management Account (PMA) associates your AWS management account with your Channel Program registration. Program management accounts are activated self-service using channel handshakes. When you create and activate a PMA, channel program benefits and discounts are applied to all invoices delivered to the associated AWS management account. Use these APIs to report and manage AWS accounts that handle reselling and Channel Discounts:

- `CreateProgramManagementAccount`: Add new accounts for customer billing management
- `UpdateProgramManagementAccount`: Modify existing accounts
- `DeleteProgramManagementAccount`: Remove accounts from resale programs

- `ListProgramManagementAccounts`: View existing accounts and their status

Working with channel handshakes

Channel handshakes enable secure, mutual consent for critical channel management operations. AWS Partner Central uses channel handshakes to verify consent from AWS management accounts for three distinct purposes: activating Program Management Accounts (PMAs), establishing service periods, and terminating service periods early. There are three types:

- `PROGRAM_MANAGEMENT_ACCOUNT`: Program Management Account handshakes verify consent when activating PMAs to apply channel program benefits
- `START_SERVICE_PERIOD`: Service period creation handshakes establish mutual agreements for billing transfer commitments with minimum notice periods or fixed terms
- `REVOKE_SERVICE_PERIOD`: Service period termination handshakes enable early termination of active service periods when both parties consent. All handshake types require acceptance from the target AWS management account to take effect.

Use these APIs to manage channel handshakes for PMA activation and service period management:

- `CreateChannelHandshake`: Create handshakes for PMA invitations, service period establishment, or service period termination
- `AcceptChannelHandshake`: Accept handshakes from target AWS accounts (can be used by partners or customers depending on handshake type)
- `RejectChannelHandshake`: Reject handshakes from target AWS accounts
- `CancelChannelHandshake`: Cancel pending handshakes before they are accepted, rejected, or expired
- `ListChannelHandshakes`: Track and view handshakes with filtering by type and status

Working with channel relationships

Relationships enable you to manage end customers, internal organizations, or downstream sellers. Each relationship includes:

- The AWS management account of the associated AWS organization
- Required AWS Partner Network metadata for channel discount qualification

When you create a relationship, all usage from the associated AWS organization receives the related Channel Program benefits. Use these APIs to report sellers and end customers:

- `CreateRelationship`: Onboard new end customer, distribution-seller, or internal AWS accounts
- `GetRelationship`: View specific relationship details
- `ListRelationships`: View all relationships
- `UpdateRelationship`: Modify existing relationships
- `DeleteRelationship`: Remove relationships

Using the Revenue Measurement API

Topics

Working with your Revenue Attribution IDs

A Revenue Attribution ID is a deal-level identifier that maps your AWS Marketplace product revenue to specific AWS Marketplace Offers and/or AWS Partner Central Opportunities. It builds on top of product-level Partner Revenue Measurement (PRM) implementation: PRM captures total attributed revenue for a Marketplace product, and a Revenue Attribution ID maps that revenue to specific deals according to the monthly cost allocation percentages you specify.

The Revenue Attribution Service exposes APIs to create, retrieve, list, and update Revenue Attribution IDs and to manage their monthly cost allocation entries asynchronously.

What is a Revenue Attribution ID?

A Revenue Attribution ID has two layers:

- **The attribution** a partner-named, partner-described resource identified by an `ra-` prefix ID (for example, `ra-aabbccddee001`) and an ARN. Each attribution is scoped to a `Catalog` (`AWS` for production, `Sandbox` for testing) and the partner's account.
- **One or more monthly cost allocation entries** each entry maps a single (Offer ID or Opportunity ID, Billing Month) combination to a cost-allocation percentage. Entries are managed in batches asynchronously through an allocation task pattern.

A Revenue Attribution ID can be used in two ways:

- **As a deal-level overlay** on your existing product-level PRM implementation. Create a Revenue Attribution ID, associate your applicable Marketplace Offers and/or ACE Opportunities, and AWS maps your already-measured product revenue to those specific deals — no changes to your existing resource tags or user agent strings are required.
- **As a standalone identifier** used directly as a resource-tag value (`aws-apn-id` = `<RA ID>`) or in a user-agent string (`APN_1.1/pc_<RA ID>$`) to attribute AWS consumption to a specific deal from the start.

Working with your Revenue Attribution IDs

Partners can manage Revenue Attribution IDs and their monthly cost allocation entries through the Revenue Attribution Service. The lifecycle progresses through two independent flows: the attribution-record flow (create, update, retrieve, list) and the allocations flow (start a batch task, poll for results, retrieve individual entries, list entries for an attribution).

Creating a Revenue Attribution ID

The first step is creating a Revenue Attribution ID using the `CreateRevenueAttribution` API action. The returned `Id` and `Arn` can be used immediately as a resource-tag value or in a user-agent string to attribute AWS consumption.

When creating a Revenue Attribution ID, partners must provide:

- **`Catalog`** — `AWS` for production or `Sandbox` for testing.
- **`Name`** — a human-readable name for the attribution. Must be unique within the Catalog and the partner's account. Maximum 128 characters.

Partners can optionally provide:

- **`Description`** — free-text description of the attribution. Maximum 1024 characters.
- **`MarketplaceProduct`** — the AWS Marketplace product to associate with this attribution. Provide `ProductIdentifier` (the 25-character Marketplace product ID) and `TenancyModel` (`MULTI_TENANT` or `SINGLE_TENANT`). If omitted at creation, attribution applies across all consumption measured under the Revenue Attribution ID, regardless of which Marketplace product the customer purchased — useful when a single deployment spans more than one Marketplace listing.

- **Tags** — up to 200 key-value pairs for resource organization. Tag keys must be unique within the request.

Best Practice: Provide `MarketplaceProduct.ProductIdentifier` whenever your deal is anchored to a specific Marketplace listing. This enables AWS to validate that the product is owned by the calling partner account or by a subsidiary account connected via Partner Account Connection (PAC), and surfaces the resolved `ProductCode` and `ProductType` (for example, `SaaS`, `AMI`, `ML`) in the response. If the product is not owned by an authorized account, the API returns `ValidationException` with reason `PRODUCT_NOT_FOUND_OR_NOT_OWNED`.

The response returns the new `Id` (format `ra-[a-z0-9]{13}`), `Arn`, resolved `MarketplaceProduct` attributes, and the initial `Version` number (which starts at 1 and increments on each subsequent update).

Adding monthly cost allocation entries

Once a Revenue Attribution ID is created, partners associate AWS Marketplace Offers and/or ACE Opportunities to it by submitting a batch of monthly cost allocation entries through the `StartRevenueAttributionAllocationsTask` API action. This is an asynchronous operation that accepts up to 250 allocation changes (`CREATE` and/or `UPDATE`) per task.

Each allocation entry specifies:

- **Offer ID or Opportunity ID** — the unique identifier of the deal being associated. If a Marketplace Offer is already linked to an ACE Opportunity, partners only need to provide the Offer ID; AWS automatically attributes revenue to the linked Opportunity.
- **Billing Month** — the calendar month for which the cost allocation applies (for example, `2026-04`).
- **Cost Allocation %** — the portion of the product's total AWS consumption attributable to this deal in this billing month. Required for multi-tenant SaaS products, including partner-hosted components for hybrid deployments, where customers share infrastructure.
- **Customer AWS Account ID** — the AWS account of the customer that consumes the product. Required for multi-tenant SaaS products.

The total Cost Allocation % across all allocation entries for a given Revenue Attribution ID and the same Billing Month must not exceed 100%. If the total exceeds 100%, the offending entry is rejected during async business validation with a per-record error code.

Synchronous validation: `StartRevenueAttributionAllocationsTask` performs basic shape validation synchronously (field types, patterns, batch size). If the basic validation passes, the API returns a `TaskId` with status `IN_PROGRESS`. Business validation (cap checks, immutability rules, dependency lookups against Marketplace and ACE) runs asynchronously and the per-record results are discovered through `GetRevenueAttributionAllocationsTask`.

To monitor a submitted task, partners poll `GetRevenueAttributionAllocationsTask` with the Revenue Attribution Resource Id or ARN. The response progresses from `IN_PROGRESS` to `COMPLETED` (regardless of whether individual records succeeded or failed) and returns:

- **`TaskStatus`** — `IN_PROGRESS`, `COMPLETED`, or `FAILED` (top-level task failure).
- **`RecordResults`** — for each input record: the assigned `AllocationId` (if successful), the per-record status (`SUCCEEDED` or `FAILED`), and a structured error code and message if the record failed.

Partners should retrieve task results promptly. After the task completes, the per-record outcomes can be reconciled and any failed entries can be corrected and resubmitted in a new task.

Editing monthly cost allocation entries

Cost allocation entries are managed per billing month with these rules:

- Partners can **add a new monthly entry** for the current or any future billing month at any time.
- Partners can **update a monthly entry** for the current or any future billing month at any time.
- Partners can **update last month's entry** until the **7th of the current month**. This window allows partners to review actual usage in AWS Cost Explorer before finalizing the prior month's allocation.

After the 7th of the current month, attribution for the prior billing month can no longer be modified. Updates to a current or future month's entry apply from the next monthly billing cycle. Historical attribution for prior months is not retroactively recalculated.

Updates to allocation entries are submitted through the same `StartRevenueAttributionAllocationsTask` API action with `Operation` set to `UPDATE` for each affected entry. The async task pattern handles updates and creations uniformly.

Updating a Revenue Attribution ID

Partners can update the `Description` of an existing Revenue Attribution ID using the `UpdateRevenueAttribution` API action. The attribution record itself is otherwise immutable — `Name`, `MarketplaceProduct`, and tag-on-create values cannot be changed after creation. To re-tag the resource, use the standard AWS tagging operations on the attribution's ARN.

When updating a Revenue Attribution ID, partners must provide:

- **`Catalog`** — `AWS` or `Sandbox`.
- **`Identifier`** — the `ra-` ID of the attribution to update.
- **`Version`** — the current version of the attribution for optimistic locking.

Partners can optionally provide:

- **`Description`** — the updated free-text description. Maximum 1024 characters.
- **`ClientToken`** — an idempotency token for the update.

Optimistic Locking: The `Version` field ensures updates apply only if the attribution hasn't changed since it was last retrieved. If the submitted `Version` doesn't match the current resource version, the update is rejected with `ConflictException` and a message including the submitted and current version numbers. Best practice is to retrieve the latest version with `GetRevenueAttribution` before each update.

Viewing Revenue Attribution ID details

Partners can retrieve complete information for a single Revenue Attribution ID using the `GetRevenueAttribution` API action. This returns:

Resource Metadata:

- Unique identifier (`Id`) and Amazon Resource Name (`Arn`).
- The `Catalog` in which the attribution exists.
- `Name` and `Description`.
- The retrieved `Version` and the `LatestVersion` (use the latest for subsequent updates).

Marketplace Product Attributes (if `MarketplaceProduct` was set at creation):

- ProductId — the Marketplace product ID provided by the partner.
- ProductCode — the AWS Marketplace product code resolved from ProductId.
- ProductType — SaaS, AMI, Container, ML, Data, or Professional Services.
- TenancyModel — MULTI_TENANT or SINGLE_TENANT.

Audit Information:

- CreatedDate and LastModifiedDate.

Partners can optionally provide a specific Version in the request to retrieve a historical version of the attribution. Omit Version to return the latest.

To retrieve a specific monthly cost allocation entry, partners use the GetRevenueAttributionAllocation API action with the entry's AllocationId. This returns the Offer ID or Opportunity ID, Billing Month, Cost Allocation %, Customer AWS Account ID, and the entry's status and audit information.

Listing Revenue Attribution IDs

Partners can view all Revenue Attribution IDs in their account using the ListRevenueAttributions API action. This returns a paginated list of attribution summaries with filtering and sorting capabilities.

Partners can filter results by:

- **Catalog** — AWS or Sandbox (required).
- **Identifiers** — a list of up to 100 specific ra- IDs to retrieve.
- **CreatedAfter / CreatedBefore** — filter by creation timestamp range (inclusive).

Partners can configure result sorting using the Sort parameter:

- SortBy — CreatedDate or LastModifiedDate.
- SortOrder — ASCENDING or DESCENDING.

The response includes a summary for each attribution: Arn, Id, Catalog, Name, resolved MarketplaceProduct attributes, CreatedDate, LastModifiedDate, the latest Version, and

`TotalRevenueAttributionAssociationCount` (the number of active monthly cost allocation entries linked to the attribution).

Use `MaxResults` (default 25, maximum 100) and `NextToken` to paginate through large result sets. The response includes a `NextToken` if additional pages are available.

Listing monthly cost allocation entries

Partners can list the monthly cost allocation entries for a specific Revenue Attribution ID using the `ListRevenueAttributionAllocations` API action. This returns a paginated list of allocation summaries with filtering capabilities.

Partners can filter results by:

- **`MarketplaceOfferId`** — list only entries associated with a specific Marketplace Offer.
- **`AwsPartnerCentralOpportunityId`** — list only entries associated with a specific Partner Central Opportunity.
- **`BillingMonth`** — list only entries for a specific calendar month.

The response includes summary information for each entry: the `AllocationId`, the associated Offer ID or Opportunity ID, the Customer AWS Account ID, the Billing Month, the Cost Allocation %, the entry's name and status, and audit timestamps.

Dashboard Building: The combination of `ListRevenueAttributions` and `ListRevenueAttributionAllocations` is designed to support partner dashboard creation. Partners can build views such as:

- All Revenue Attribution IDs created in the last 30 days, sorted by creation date.
- All monthly cost allocation entries for a specific Marketplace Offer across multiple Revenue Attribution IDs.
- All cost allocation entries for the current billing month, to validate that totals do not exceed 100% per attribution.
- All Revenue Attribution IDs that have at least one active allocation entry (`TotalRevenueAttributionAssociationCount > 0`).

Working with your Marketplace Revenue Shares

A Marketplace Revenue Share is an input for an AWS Marketplace product that declares the portion of the product's total revenue collected through AWS Marketplace. AWS uses this input to correlate a partner's Marketplace-collected revenue against other revenue signals.

The Marketplace Revenue Share Service exposes APIs to create, retrieve, and list Marketplace Revenue Share resources, and to manage their allocations (the partner-declared revenue-share percentages and effective date windows). Standard AWS tagging operations are supported on the Marketplace Revenue Share resource.

What is a Marketplace Revenue Share?

A Marketplace Revenue Share has two layers:

- **The share** — an input for a single AWS Marketplace product, identified by the Marketplace `ProductId`. There is exactly one Marketplace Revenue Share per product.
- **One or more allocations** — each allocation declares a `RevenueSharePercent` and an effective date window (`EffectiveDate`, optionally `ExpirationDate`). A share can have many allocations over time, but at any moment only one `ACTIVE` allocation may apply.

Allocations have these key properties:

- **Status** — `ACTIVE` or `INACTIVE`. On creation, status defaults to `ACTIVE`. Only `ACTIVE` allocations participate in revenue calculations and overlap checks.
- **Overlap invariant** — within a single share, no two `ACTIVE` allocations may have overlapping `[EffectiveDate, ExpirationDate]` ranges. At most one open-ended `ACTIVE` allocation (no `ExpirationDate`) is permitted per share at any time.

Working with your Marketplace Revenue Shares

Partners manage Marketplace Revenue Shares and their allocations through the Marketplace Revenue Share Service. The lifecycle progresses through two flows: the share flow (create, retrieve, list, tag) and the allocations flow (create, update, soft-delete, retrieve, list).

Creating a Marketplace Revenue Share

The first step is creating a Marketplace Revenue Share for an AWS Marketplace product using the `CreateMarketplaceRevenueShare` API action. The share is identified by the Marketplace `ProductId` and serves as the container for all subsequent allocations.

When creating a Marketplace Revenue Share, partners must provide:

- **`Catalog`** — `AWS` for production or `Sandbox` for testing.
- **`ProductId`** — the 25-character Marketplace product ID. Pattern: `[a-z0-9]{25}`.

Partners can optionally provide:

- **`Tags`** — up to 50 key-value pairs for resource organization at creation time. Use `TagResource` to add tags after creation.
- **`ClientToken`** — an idempotency token to prevent duplicate creations on retry. Maximum 64 characters. SDKs auto-generate this token.

Product Validation: On creation, the service verifies with AWS Marketplace that the product is owned by the caller's account or by a subsidiary account connected to the caller via Partner Account Connection (PAC). If the product does not exist or the owning account is not authorized for the caller, the API returns `ValidationException` with reason `PRODUCT_NOT_FOUND_OR_NOT_OWNED`.

One Share Per Product: A second `CreateMarketplaceRevenueShare` call for the same `ProductId` in the same `Catalog` returns `ConflictException`.

The response returns the share's `Arn` (format `arn:{partition}:partnercentral:{region}:{account-id}:catalog/{catalog}/marketplace-revenue-share/{productId}`), the `Catalog`, the `ProductId`, the initial `Version` number (which starts at 1 and increments on each allocation mutation), and any tags applied at creation.

Adding allocations

Once a Marketplace Revenue Share is created, partners declare a revenue-share percentage and an effective date window by creating an allocation through the `CreateMarketplaceRevenueShareAllocation` API action.

When creating an allocation, partners must provide:

- **`Catalog`** — `AWS` or `Sandbox`.
- **`MarketplaceRevenueShareIdentifier`** — the identifier of the parent share.
- **`RevenueSharePercent`** — the percentage of the product's revenue collected through Marketplace provided as a decimal (for example, `0.45` for 45%).
- **`EffectiveDate`** — the calendar date when this allocation becomes effective. Format: `YYYY-MM-DD`.

The response returns the allocation's `Id` (format `mrsa-[A-Za-z0-9]{13}`), the allocation status (`ACTIVE` on creation), and the parent share's bumped `Version` (a single version integer per share is bumped on every allocation mutation, used for optimistic locking on subsequent allocation updates).

Updating allocations

Partners can update an allocation using the `UpdateMarketplaceRevenueShareAllocation` API action. The updatable fields depend on whether the allocation is future-dated, currently-effective, or past-dated.

When updating an allocation, partners must provide:

- **`Catalog`** — `AWS` or `Sandbox`.
- **`MarketplaceRevenueShareIdentifier`** — the identifier of the parent share.
- **`AllocationId`** — the `mrsa-` ID of the allocation to update.
- **`MarketplaceRevenueShareVersion`** — the current version of the parent share for optimistic locking on the allocation. Bump conflicts return `ConflictException`.

Partners can optionally provide:

- **`RevenueSharePercent`** — the updated revenue-share percentage. Editable only on future-dated allocations.
- **`EffectiveDate`** — the updated effective date. Editable only on future-dated allocations.
- **`ExpirationDate`** — the updated expiration date. Editable on future-dated allocations (any value `> EffectiveDate`) and on currently-effective or open-ended allocations (any value `≥ today`).
- **`Status`** — `ACTIVE` or `INACTIVE`. Setting to `INACTIVE` soft-deletes the allocation. Editable only on future-dated allocations.

- **`ClientToken`** — an idempotency token.

Viewing Marketplace Revenue Share details

Partners can retrieve complete information for a single Marketplace Revenue Share using the `GetMarketplaceRevenueShare` API action. This returns the share's `Arn`, `Catalog`, `ProductId`, `Version` (the parent share's current version), `CreatedDate`, `LastModifiedDate`, and any tags applied to the share.

Partners can retrieve a single allocation using the `GetMarketplaceRevenueShareAllocation` API action with the allocation's `Id`. This returns:

- The allocation's `Id` and parent share's `MarketplaceRevenueShareArn`.
- `RevenueSharePercent`, `EffectiveDate`, `ExpirationDate`.
- `Status` (`ACTIVE` or `INACTIVE`).
- The parent share's `MarketplaceRevenueShareVersion` for optimistic locking on subsequent updates.
- `CreatedDate` and `LastModifiedDate`.

Listing Marketplace Revenue Shares

Partners can view all Marketplace Revenue Shares owned by their account using the `ListMarketplaceRevenueShares` API action. This returns a paginated list of share summaries.

Partners can filter results by:

- **`Catalog`** — `AWS` or `Sandbox` (required).
- **`ProductIds`** — a list of specific Marketplace product IDs to retrieve.

The response includes a summary for each share: `Arn`, `Catalog`, `ProductId`, the latest `Version`, `CreatedDate`, and `LastModifiedDate`.

Use `MaxResults` and `NextToken` to paginate through large result sets.

Listing allocations

Partners can list the allocations for a specific Marketplace Revenue Share using the `ListMarketplaceRevenueShareAllocations` API action. This returns a paginated list of allocation summaries with filtering capabilities.

Partners can filter results by:

- **`Status`** — `ACTIVE` or `INACTIVE`.
- **Effective date range** — `EffectiveDateOnOrAfter` / `EffectiveDateOnOrBefore` to retrieve allocations whose effective date falls within a specific window.

The response includes summary information for each allocation: `Id`, `RevenueSharePercent`, `EffectiveDate`, `ExpirationDate`, `Status`, `MarketplaceRevenueShareVersion`, and audit timestamps.

Tagging Marketplace Revenue Shares

Standard AWS tagging operations are supported on the Marketplace Revenue Share resource:

- **`TagResource`** — add or overwrite tags on a Marketplace Revenue Share. Maximum 50 tags per share.
- **`UntagResource`** — remove specific tags by key.
- **`ListTagsForResource`** — retrieve all tags currently applied to a Marketplace Revenue Share.

Allocations do not support tags. Tag the parent share to organize related allocations.

Each tagging operation accepts the share's `Arn` (format `arn:{partition}:partnercentral:{region}:{account-id}:catalog/{catalog}/marketplace-revenue-share/{productId}`) as the resource identifier. Standard AWS tag-on-create is also supported through the `Tags` field on `CreateMarketplaceRevenueShare`.

Supported AWS Regions

The following sections provide information about supported AWS Regions.

Topics

Supported AWS regions for the AWS Partner Central Account API

You can access the AWS Partner Central account service from any AWS US East (N. Virginia) region with the following end point.

```
partnercentral-account.us-east-1.api.aws
```

Supported AWS regions for the AWS Partner Central Selling API

You can access the AWS Partner Central selling service from any AWS US East (N. Virginia) region with the following end point.

```
partnercentral-selling.us-east-1.api.aws
```

Supported AWS regions for the AWS Partner Central Benefits API

You can access the AWS Partner Central benefits service from any AWS US East (N. Virginia) region with the following end point.

```
partnercentral-benefits.us-east-1.api.aws
```

Supported AWS regions for the AWS Partner Central Channel API

You can access the AWS Partner Central channel management service from any AWS commercial region with the following end point.

```
partnercentral-channel.global.api.aws
```

Signing requests with SigV4a

Because the AWS Partner Central Channel API uses a global endpoint, requests are signed with *Signature Version 4a (SigV4a)*, an extension of SigV4 that supports signing with asymmetric cryptography, enabling signatures that are verifiable across multiple AWS Regions.

Using the AWS SDK or CLI

If you use an AWS SDK or the AWS CLI, SigV4a signing is handled automatically when you call the global endpoint. No additional configuration is required.

Signing requests manually

If you sign requests manually without an SDK, be aware of the following differences from standard SigV4:

- SigV4a derives an Elliptic Curve Digital Signature Algorithm (ECDSA) keypair from your existing AWS secret access key, rather than using HMAC-based key derivation.
- The credential scope uses `*` for the region component instead of a specific Region name, because the signature is valid across Regions.

For more information about how SigV4a works, see AWS Signature Version 4 for API requests. For code examples of SigV4a signing, see the sigv4a-signing-examples project on GitHub.

Supported AWS regions for the AWS Partner Central Revenue Measurement API

You can access the AWS Partner Central revenue measurement service from any AWS US East (N. Virginia) region with the following end point.

```
partnercentral-prm.us-east-1.api.aws
```

Permissions

The following sections provide information about permissions.

Topics

- Access for the Account API
- Access for the Selling API

- Access for the Benefits API
- Access for the Channel API
- Access for the Revenue Measurement API

Access for the Account API

Access control and permissions are managed by AWS Identity and Access Management (IAM). This section provides guidance for configuring the necessary permissions to interact with the Account API.

Prerequisites

Before configuring permissions, ensure that your AWS account is linked to and that you created the necessary IAM roles and users. For more information, see Setup and authentication.

Using AWS managed policies

AWS provides managed policies that grant the required permissions to interact with the Account API. To provide the necessary access to manage account resources, attach the `AWSPartnerCentralFullAccess` policy to your IAM identities. For more information, see AWS managed policies for users.

Assigning policies to IAM roles and users

Follow these steps to assign policies to IAM roles and users:

1. Sign in to the AWS Management Console.
2. Navigate to the IAM service.
3. Select roles or users, and choose the IAM role or user to which you want to attach a policy.
4. Attach the `AWSPartnerCentralFullAccess` policy to the selected IAM role or user.

For more information, see Adding and removing IAM identity permissions.

Managing permissions using condition keys

Condition keys in IAM policies provide resource-level permissions for when to enforce statement policies. You can use condition keys to specify conditions that dictate when certain permissions are allowed or denied.

For more information, see IAM JSON policy elements: Condition operators.

Condition keys overview

Condition key	Description	Applicable actions	Valid values
partnercentral:Catalog	filters access by the type of the associated catalog entity	all actions	AWS, sandbox

Summary of required permissions

Summary of required permissions

Action	Description
partnercentral:AcceptConnectionInvitation	allows accepting connection invitations
partnercentral:AssociateAwsTrainingCertificationEmailDomain	allows associating AWS training certification email domains
partnercentral:CancelConnection	allows canceling connections
partnercentral:CancelConnectionInvitation	allows canceling connection invitations
partnercentral:CancelProfileUpdateTask	allows canceling profile update tasks
partnercentral:CreateConnectionInvitation	allows creating connection invitations
partnercentral:CreatePartner	allows creating partners
partnercentral:DisassociateAwsTrainingCertificationEmailDomain	allows disassociating AWS training certification email domains
partnercentral:GetAllianceLeadContact	allows retrieving alliance lead contact details
partnercentral:GetConnection	allows retrieving connection details
partnercentral:GetConnectionInvitation	allows retrieving connection invitation details
partnercentral:GetConnectionPreferences	allows retrieving connection preferences

Action	Description
partnercentral:GetPartner	allows retrieving partner details
partnercentral:GetProfileUpdateTask	allows retrieving profile update task details
partnercentral:GetProfileVisibility	allows retrieving profile visibility settings
partnercentral:GetVerification	allows retrieving verification details
partnercentral:ListConnectionInvitations	allows listing connection invitations
partnercentral:ListConnections	allows listing connections
partnercentral:ListPartners	allows listing partners
partnercentral:PutAllianceLeadContact	allows updating alliance lead contact details
partnercentral:PutProfileVisibility	allows updating profile visibility settings
partnercentral:RejectConnectionInvitation	allows rejecting connection invitations
partnercentral:SendEmailVerificationCode	allows sending email verification codes
partnercentral:StartProfileUpdateTask	allows starting profile update tasks
partnercentral:StartVerification	allows starting verification processes
partnercentral:UpdateConnectionPreferences	allows updating connection preferences

Access for the Selling API

Access control and permissions are managed by AWS Identity and Access Management (IAM). This section provides guidance for configuring the necessary permissions to interact with the API, including the permissions required to list AWS Marketplace entities.

Prerequisites

Before configuring permissions, ensure that your AWS account is linked to Partner Central and that you created the necessary IAM roles and users. For more information, see Setup and Authentication.

Using AWS managed policies

AWS provides managed policies that grant the required permissions to interact with the API. To provide the necessary access to manage opportunities, attach the `AWSPartnerCentralOpportunityManagement` policy to your IAM identities. For more information, see AWS managed policies for AWS Partner Central users.

AWSPartnerCentralOpportunityManagement policy

This policy grants full access to Partner Central opportunity management actions.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "OpportunityManagement",
            "Effect": "Allow",
            "Action": [
                "partnercentral:AcceptEngagementInvitation",
                "partnercentral:AssignOpportunity",
                "partnercentral:AssociateOpportunity",
                "partnercentral:CreateEngagement",
                "partnercentral:CreateEngagementInvitation",
                "partnercentral:CreateOpportunity",
                "partnercentral:CreateResourceSnapshot",
                "partnercentral:CreateResourceSnapshotJob",
                "partnercentral:DeleteResourceSnapshotJob",
                "partnercentral:DisassociateOpportunity",
                "partnercentral:GetAwsOpportunitySummary",
                "partnercentral:GetEngagement",
                "partnercentral:GetEngagementInvitation",
                "partnercentral:GetOpportunity",
                "partnercentral:GetResourceSnapshot",
                "partnercentral:GetResourceSnapshotJob",
                "partnercentral:ListEngagementByAcceptingInvitationTasks",
                "partnercentral:ListEngagementFromOpportunityTasks",
                "partnercentral:ListEngagementInvitations",
                "partnercentral:ListEngagementMembers",
                "partnercentral:ListEngagementResourceAssociations",
                "partnercentral:ListEngagements",
```

```
                "partnercentral:ListOpportunities",
                "partnercentral:ListResourceSnapshotJobs",
                "partnercentral:ListResourceSnapshots",
                "partnercentral:ListSolutions",
                "partnercentral:RejectEngagementInvitation",
                "partnercentral:StartEngagementByAcceptingInvitationTask",
                "partnercentral:StartEngagementFromOpportunityTask",
                "partnercentral:StartResourceSnapshotJob",
                "partnercentral:StopResourceSnapshotJob",
                "partnercentral:SubmitOpportunity",
                "partnercentral:UpdateOpportunity"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ListingAWSMarketplaceEntities",
            "Effect": "Allow",
            "Action": ["aws-marketplace:ListEntities"],
            "Resource": "*"
        },
        {
            "Sid": "AWSMarketplaceOffersAccess",
            "Effect": "Allow",
            "Action": ["aws-marketplace:DescribeEntity"],
            "Resource": [
              "arn:aws:aws-marketplace:*:*:AWSMarketplace/Offer/*",
              "arn:aws:aws-marketplace:*:*:AWSMarketplace/OfferSet/*"
            ]
        }
    ]
}
```

Custom policies

If the managed policies don't meet your needs, create custom IAM policies that grant the permissions required for your use case. The following example is a custom policy that grants permissions to list AWS Marketplace entities:

Example of custom policy

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "partnercentral:ListOpportunities",
                "aws-marketplace:ListEntities"
            ],
            "Resource": "*"
        }
    ]
}
```

Custom permissive policy

This policy provides broad access to Partner Central selling actions, including features that may be added in the future without requiring policy updates. By using the wild card action `partnercentral:*`, this policy automatically grants access to new Partner Central selling features as they become available, reducing the need for manual updates. This policy also includes permissions for interacting with AWS Marketplace entities, which helps to ensure access is maintained for both selling and Marketplace actions.

JSON

```
{
    "Version":"2012-10-17",
    "Statement":
    [
        {
            "Effect": "Allow",
            "Action":
            [
                "partnercentral:*"
            ],
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action":
            [
                "aws-marketplace:ListEntities",
                "aws-marketplace:DescribeEntity"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action":
            [
                "aws-marketplace:SearchAgreements",
                "aws-marketplace:DescribeAgreement"
            ],
            "Resource": "*",
            "Condition":
            {
                "StringEquals":
                {
                    "aws-marketplace:PartyType": "Proposer"
                }
            }
        }
    ]
}
```

Assigning policies to IAM roles and users

Follow these steps to assign policies to IAM roles and users:

1. Sign in to the AWS Management Console.
2. Navigate to the IAM service.
3. Select roles or users, and choose the IAM role or user to which you want to attach a policy.
4. Attach the `AWSPartnerCentralOpportunityManagement` policy or your custom policy to the selected IAM role or user.

For more information, see Adding and removing IAM identity permissions.

Managing permissions using condition keys

Condition keys in IAM policies provide resource-level permissions for when to enforce statement policies. You can use condition keys to specify conditions that dictate when certain permissions are allowed or denied.

For more information, see IAM JSON policy elements: Condition operators.

Condition keys overview

Condition key	Description	Applicable actions	Valid values
partnercentral:Catalog	filters access by the type of the associated catalog entity	all actions	AWS, sandbox
aws-marketplace:PartyType	filters access based on the type of party (e.g., proposer)	SearchAgreements, DescribeAgreement	proposer

Summary of required permissions

Summary of required permissions

Action	Description
partnercentral:CreateOpportunity	allows creating opportunities
partnercentral:UpdateOpportunity	allows updating opportunities
partnercentral:ListOpportunities	allows listing opportunities
partnercentral:GetOpportunity	allows retrieving opportunity details
partnercentral:ListSolutions	allows listing solutions
partnercentral:AssociateOpportunity	allows associating opportunities with other entities

Action	Description
partnercentral:DisassociateOpportunity	allows disassociating opportunities from other entities
partnercentral:AcceptEngagementInvitation	allows accepting engagement invitations
partnercentral:RejectEngagementInvitation	allows rejecting engagement invitations
partnercentral:GetEngagementInvitation	allows retrieving engagement invitation details
partnercentral:ListEngagementInvitations	allows listing engagement invitations
partnercentral:SubmitOpportunity	allows submitting opportunities
partnercentral:GetAwsOpportunitySummary	allows retrieving AWS opportunity summary
partnercentral:StartProspectingFromEngagementTask	Creates a prospecting task from engagements
partnercentral:GetProspectingFromEngagementTask	Returns prospecting task details
partnercentral:ListProspectingFromEngagementTasks	Returns a list of prospecting tasks
aws-marketplace:ListEntities	allows listing AWS Marketplace entities
aws-marketplace:DescribeEntity	allows describing AWS Marketplace entities
aws-marketplace:SearchAgreements	allows searching agreements in AWS Marketplace
aws-marketplace:DescribeAgreement	allows describing agreements in AWS Marketplace

Access for the Benefits API

Access control and permissions are managed by AWS Identity and Access Management (IAM). This section provides guidance for configuring the necessary permissions to interact with the Benefits API.

Prerequisites

Before configuring permissions, ensure that your AWS account is linked to and that you created the necessary IAM roles and users. For more information, see Setup and authentication.

Using AWS managed policies

AWS provides managed policies that grant the required permissions to interact with the Benefits API. To provide the necessary access to manage benefits resources, attach the `AWSPartnerCentralFullAccess` policy to your IAM identities. For more information, see AWS managed policies for users.

Assigning policies to IAM roles and users

Follow these steps to assign policies to IAM roles and users:

1. Sign in to the AWS Management Console.
2. Navigate to the IAM service.
3. Select roles or users, and choose the IAM role or user to which you want to attach a policy.
4. Attach the `AWSPartnerCentralFullAccess` policy to the selected IAM role or user.

For more information, see Adding and removing IAM identity permissions.

Managing permissions using condition keys

Condition keys in IAM policies provide resource-level permissions for when to enforce statement policies. You can use condition keys to specify conditions that dictate when certain permissions are allowed or denied.

For more information, see IAM JSON policy elements: Condition operators.

Condition keys overview

Condition key	Description	Applicable actions	Valid values
partnercentral:Catalog	filters access by the type of the associated catalog entity	all actions	AWS, sandbox

Summary of required permissions

Summary of required permissions

Action	Description
partnercentral:AmendBenefitApplication	allows amending benefit applications
partnercentral:AssociateBenefitApplicationResource	allows associating resources with benefit applications
partnercentral:CancelBenefitApplication	allows canceling benefit applications
partnercentral:CreateBenefitApplication	allows creating benefit applications
partnercentral:DisassociateBenefitApplicationResource	allows disassociating resources from benefit applications
partnercentral:GetBenefit	allows retrieving benefit details
partnercentral:GetBenefitAllocation	allows retrieving benefit allocation details
partnercentral:GetBenefitApplication	allows retrieving benefit application details
partnercentral:ListBenefitAllocations	allows listing benefit allocations
partnercentral:ListBenefitApplications	allows listing benefit applications
partnercentral:ListBenefits	allows listing benefits
partnercentral:RecallBenefitApplication	allows recalling benefit applications
partnercentral:SubmitBenefitApplication	allows submitting benefit applications

Action	Description
partnercentral:UpdateBenefitApplication	allows updating benefit applications

Access for the Channel API

Access control and permissions are managed by AWS Identity and Access Management (IAM). This section provides guidance for configuring the necessary permissions to interact with the AWS Partner Central Channel API.

Channel management account setup

Channel management features on AWS Partner Central require IAM roles to be deployed in two types of AWS accounts:

1. AWS Partner Central linked AWS account:

   This is the AWS account associated with your AWS Partner Network and Partner Central account. There is only one AWS Partner Central linked AWS account per partner. As a one time set up activity, you will create an IAM role in this AWS account using a Partner Central channel managed policy.

2. Program management account AWS account:

   This is the AWS account used as a Bill-Transfer account to manage billing and payment for end customer consumption. This AWS account is reported as a program management account in AWS Partner Central channel management. You may have multiple program management accounts, and you will need to create an IAM role in each AWS account you report as a program management account.

For each AWS account type, you will need to associate different IAM roles with specific permission and trust policies.

Access for the AWS Partner Central linked AWS account

Within the AWS Partner Central linked AWS account, create an IAM role to manage channel resources, which will be granted to Partner Central users. This IAM role allows users to view and manage channel management resources on AWS Partner Central. The role name must start with

PartnerCentralRoleFor, and the recommended name is PartnerCentralRoleForChannel. To configure the role, complete the following:

- Add the following custom trust policy to allow AWS Partner Central cloud admins and alliance leads to map IAM roles to Partner Central users:

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

- Associate the AWSPartnerCentralChannelManagement managed policy to the PartnerCentralRoleForChannel role.

Access for the program management account AWS account(s)

Within each AWS account reported as a program management account in AWS Partner Central, create the following IAM roles, with the required names below:

- PartnerCentralChannelHandshakeApprovalManagement: Utilize this role to manage handshakes between Partner Central and your program management AWS account. This IAM role can be utilized to respond to activate your program management accounts in AWS Partner Central. When creating this role, associate the AWSPartnerCentralChannelHandshakeApprovalManagement managed policy.
- PartnerCentralChannelBillingTransferReadOnly: Utilize this cross-account role to enable visibility to billing transfer status from AWS Partner Central. This role will share billing transfer status from your AWS account to AWS Partner Central to centrally view billing transfer status. If this role is created, users in AWS Partner Central can view billing transfer status in AWS Partner Central. However, this role does not grant AWS Partner Central users permission to access billing transfers in AWS Billing and Cost Management console.
  - Trust policy:

- Within the JSON below, replace the {PartnerCentralLinkedAccountID} with your actual AWS Partner Central linked AWS account 12-digit ID when creating the role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::{PartnerCentralLinkedAccountId}:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- Permission policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "BillingTransferReadOnly",
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeResponsibilityTransfer",
        "organizations:ListHandshakesForOrganization",
        "organizations:ListInboundResponsibilityTransfers",
        "organizations:ListOutboundResponsibilityTransfers"
    ],
      "Resource": "*"
    }
  ]
}
```

- PartnerCentralChannelBillingTransferManagement (optional): Utilize this cross-account role to enable creation and management of billing transfers from AWS Partner Central. This role will enable AWS Partner Central users accessing the IAM role containing AWSPartnerCentralChannelManagement managed policy to manage billing transfers in program management account AWS account. Using this cross-account role, AWS Partner Central users will be able to access and manage billing transfers in AWS Billing and Cost Management console using the "Manage in AWS Billing" button in AWS Partner Central channel management.

- Trust policy:
  - Within the JSON below, replace the {PartnerCentralLinkedAccountID} with your actual AWS Partner Central linked AWS account 12-digit ID when creating the role.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::{PartnerCentralLinkedAccountId}:root"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- Permission policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "BillingTransferManagement",
      "Effect": "Allow",
      "Action": [
        "billingconductor:CreateBillingGroup",
        "billingconductor:ListPricingPlans",
        "organizations:AcceptHandshake",
        "organizations:CancelHandshake",
        "organizations:DeclineHandshake",
        "organizations:DescribeResponsibilityTransfer",
        "organizations:InviteOrganizationToTransferResponsibility",
        "organizations:ListHandshakesForAccount",
        "organizations:ListHandshakesForOrganization",
        "organizations:ListInboundResponsibilityTransfers",
        "organizations:ListOutboundResponsibilityTransfers",
        "organizations:TerminateResponsibilityTransfer",
        "organizations:UpdateResponsibilityTransfer"
    ],
      "Resource": "*"
    }
```

```
    ]
}
```

Using AWS managed policies

AWS provides managed policies that grant the required permissions to interact with the Channel API. To provide the necessary access to manage all channel resources, attach the `AWSPartnerCentralChannelManagement` policy to your IAM identities. To provide necessary access to view and respond to channel handshake requests, attach the `AWSPartnerCentralChannelHandshakeApprovalManagement` policy to your IAM identities. For more information, see AWS managed policies for AWS Partner Central users.

AWSPartnerCentralChannelManagement policy

This policy grants full access to Partner Central channel management actions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ChannelManagement",
      "Effect": "Allow",
      "Action": [
        "partnercentral:CreateProgramManagementAccount",
        "partnercentral:UpdateProgramManagementAccount",
        "partnercentral:DeleteProgramManagementAccount",
        "partnercentral:ListProgramManagementAccounts",
        "partnercentral:GetProgramManagementAccount",
        "partnercentral:CreateRelationship",
        "partnercentral:UpdateRelationship",
        "partnercentral:DeleteRelationship",
        "partnercentral:GetRelationship",
        "partnercentral:ListRelationships",
        "partnercentral:CreateChannelHandshake",
        "partnercentral:AcceptChannelHandshake",
        "partnercentral:RejectChannelHandshake",
        "partnercentral:CancelChannelHandshake",
        "partnercentral:ListChannelHandshakes"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
```

```
          "partnercentral:Catalog": [
            "AWS",
            "Sandbox"
          ]
        }
      }
    },
    {
      "Sid": "ChannelBillingTransferRoleAccess",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/PartnerCentralChannelBillingTransferManagement",
        "arn:aws:iam::*:role/PartnerCentralChannelBillingTransferReadOnly"
      ]
    },
    {
      "Sid": "TaggingAccess",
      "Effect": "Allow",
      "Action": [
        "partnercentral:TagResource",
        "partnercentral:UntagResource",
        "partnercentral:ListTagsForResource"
      ],
      "Resource": [
        "arn:aws:partnercentral:*:*:catalog/*/program-management-account/*",
        "arn:aws:partnercentral:*:*:catalog/*/channel-handshake/*"
      ],
      "Condition": {
        "StringEquals": {
          "partnercentral:Catalog": [
            "AWS",
            "Sandbox"
          ]
        }
      }
    }
  ]
}
```

AWSPartnerCentralChannelHandshakeApprovalManagement policy

This policy grants access to view and respond to channel handshake requests. This policy should be applied to roles in the AWS accounts receiving channel handshake requests.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ChannelHandshakeManagement",
      "Effect": "Allow",
      "Action": [
        "partnercentral:ListChannelHandshakes",
        "partnercentral:AcceptChannelHandshake",
        "partnercentral:RejectChannelHandshake"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "partnercentral:Catalog": [
            "AWS",
            "Sandbox"
          ]
        }
      }
    }
  ]
}
```

Assigning policies to IAM roles and users

Follow these steps to assign policies to IAM roles and users:

1. Sign in to the AWS Management Console.
2. Navigate to the IAM service.
3. Select roles or users, and choose the IAM role or user to which you want to attach a policy.
4. Attach the `AWSPartnerCentralChannelManagement` policy or your custom policy to the selected IAM role or user.

For more information, see Adding and removing IAM identity permissions.

Managing permissions using condition keys

Condition keys in IAM policies provide resource-level permissions for when to enforce statement policies. You can use condition keys to specify conditions that dictate when certain permissions are allowed or denied.

For more information, see IAM JSON policy elements: Condition operators.

Condition keys overview

Condition key	Description	Applicable actions	Valid values
partnercentral:Catalog	Filters access by the type of the associated catalog entity	All Channel Management actions	AWS, Sandbox
partnercentral:ChannelHandshakeType	Filters access based on the type of channel handshake	CreateChannelHandshake, AcceptChannelHandshake, RejectChannelHandshake, CancelChannelHandshake, ListChannelHandshakes	PROGRAM_MANAGEMENT_ACCOUNT, START_SERVICE_PERIOD, REVOKE_SERVICE_PERIOD

Summary of required permissions

Summary of required permissions

Action	Description
partnercentral:CreateProgramManagementAccount	allows creating program management accounts
partnercentral:UpdateProgramManagementAccount	allows updating program management accounts
partnercentral:DeleteProgramManagementAccount	allows deleting program management accounts

Action	Description
partnercentral:ListProgramManagementAccounts	allows listing program management accounts
partnercentral:GetProgramManagementAccount	allows retrieving program management account details
partnercentral:CreateRelationship	allows creating relationships
partnercentral:UpdateRelationship	allows updating relationships
partnercentral:DeleteRelationship	allows deleting relationships
partnercentral:GetRelationship	allows retrieving relationship details
partnercentral:ListRelationships	allows listing relationships
partnercentral:CreateChannelHandshake	allows creating channel handshakes
partnercentral:AcceptChannelHandshake	allows accepting channel handshakes
partnercentral:RejectChannelHandshake	allows rejecting channel handshakes
partnercentral:CancelChannelHandshake	allows canceling channel handshakes
partnercentral:ListChannelHandshakes	allows listing channel handshakes

Access for the Revenue Measurement API

Access control and permissions are managed by AWS Identity and Access Management (IAM). This section provides guidance for configuring the necessary permissions to interact with the Revenue Measurement API, including the permissions required to list AWS Marketplace entities.

Prerequisites

Before configuring permissions, ensure that your AWS account is linked to Partner Central and that you created the necessary IAM roles and users. For more information, see Setup and Authentication.

Using AWS managed policies

AWS provides managed policies that grant the required permissions to interact with the Revenue Measurement API. Two managed policies are available depending on your persona:

- **Partners** — attach the `AWSPartnerCentralRevenueAttributionManagement` policy to your IAM identities.
- **Customers** — attach the `AWSRevenueAttributionManagement` policy to your IAM identities.

For more information, see AWS managed policies for AWS Partner Central users.

AWSPartnerCentralRevenueAttributionManagement policy

This policy provides necessary access for revenue attribution management activities for AWS accounts registered with AWS Partner Central.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RevenueMeasurementCreation",
            "Effect": "Allow",
            "Action": [
                "partnercentral:CreateRevenueAttribution",
                "partnercentral:CreateMarketplaceRevenueShare",
                "partnercentral:CreateMarketplaceRevenueShareAllocation"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "AWS",
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "RevenueMeasurementListing",
            "Effect": "Allow",
            "Action": [
                "partnercentral:ListRevenueAttributions",
```

```
                "partnercentral:ListRevenueAttributionAllocations",
                "partnercentral:ListMarketplaceRevenueShares",
                "partnercentral:ListMarketplaceRevenueShareAllocations"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "AWS",
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "RevenueAttributionManagement",
            "Effect": "Allow",
            "Action": [
                "partnercentral:GetRevenueAttribution",
                "partnercentral:UpdateRevenueAttribution",
                "partnercentral:GetRevenueAttributionAllocation",
                "partnercentral:StartRevenueAttributionAllocationsTask",
                "partnercentral:GetRevenueAttributionAllocationsTask"
            ],
            "Resource": "arn:aws:partnercentral:*:*:catalog/*/revenue-attribution/*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "AWS",
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "MarketplaceRevenueShareManagement",
            "Effect": "Allow",
            "Action": [
                "partnercentral:GetMarketplaceRevenueShare",
                "partnercentral:GetMarketplaceRevenueShareAllocation",
                "partnercentral:UpdateMarketplaceRevenueShareAllocation"
            ],
            "Resource": "arn:aws:partnercentral:*:*:catalog/*/marketplace-revenue-
share/*",
```

```
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "AWS",
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "TaggingAccess",
            "Effect": "Allow",
            "Action": [
                "partnercentral:TagResource",
                "partnercentral:UntagResource",
                "partnercentral:ListTagsForResource"
            ],
            "Resource": [
                "arn:aws:partnercentral:*:*:catalog/*/revenue-attribution/*",
                "arn:aws:partnercentral:*:*:catalog/*/marketplace-revenue-share/*"
            ],
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "AWS",
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "OpportunityAccess",
            "Effect": "Allow",
            "Action": [
                "partnercentral:ListOpportunities",
                "partnercentral:GetOpportunity"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "AWS",
                        "Sandbox"
                    ]
```

```
                }
            }
        },
        {
            "Sid": "PartnerResourceAccess",
            "Effect": "Allow",
            "Action": [
                "partnercentral:ListPartners",
                "partnercentral:GetPartner"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "AWS",
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "ListingAWSMarketplaceEntities",
            "Effect": "Allow",
            "Action": [
                "aws-marketplace:ListEntities"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AWSMarketplaceEntityAccess",
            "Effect": "Allow",
            "Action": [
                "aws-marketplace:DescribeEntity"
            ],
            "Resource": [
                "arn:aws:aws-marketplace:*:*:AWSMarketplace*/Offer/*"
            ]
        },
        {
            "Sid": "AmazonQPartnerAssistantAccess",
            "Effect": "Allow",
            "Action": [
                "q:StartConversation",
                "q:SendMessage",
```

```
                "q:GetConversation",
                "q:ListConversations",
                "q:PassRequest"
            ],
            "Resource": "*"
        }
    ]
}
```

AWSRevenueAttributionManagement policy

This policy provides necessary access for revenue attribution management activities for AWS accounts who are not registered with AWS Partner Central.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RevenueAttributionCreation",
            "Effect": "Allow",
            "Action": [
                "partnercentral:CreateRevenueAttribution"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "AWS",
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "RevenueAttributionListing",
            "Effect": "Allow",
            "Action": [
                "partnercentral:ListRevenueAttributions"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
```

```
                        "AWS",
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "RevenueAttributionManagement",
            "Effect": "Allow",
            "Action": [
                "partnercentral:GetRevenueAttribution",
                "partnercentral:UpdateRevenueAttribution"
            ],
            "Resource": "arn:aws:partnercentral:*:*:catalog/*/revenue-attribution/*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "AWS",
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "TaggingAccess",
            "Effect": "Allow",
            "Action": [
                "partnercentral:TagResource",
                "partnercentral:UntagResource",
                "partnercentral:ListTagsForResource"
            ],
            "Resource": [
                "arn:aws:partnercentral:*:*:catalog/*/revenue-attribution/*"
            ],
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "AWS",
                        "Sandbox"
                    ]
                }
            }
        }
    ]
```

```
}
```

Assigning policies to IAM roles and users

Follow these steps to assign policies to IAM roles and users:

1. Sign in to the AWS Management Console.
2. Navigate to the IAM service.
3. Select roles or users, and choose the IAM role or user to which you want to attach a policy.
4. Attach the `AWSPartnerCentralRevenueAttributionManagement` policy (for partners) or the `AWSRevenueAttributionManagement` policy (for customers) to the selected IAM role or user.

For more information, see Adding and removing IAM identity permissions.

Managing permissions using condition keys

Condition keys in IAM policies provide resource-level permissions for when to enforce statement policies. You can use condition keys to specify conditions that dictate when certain permissions are allowed or denied.

For more information, see IAM JSON policy elements: Condition operators.

Condition keys overview

Condition key	Description	Applicable actions	Valid values
partnercentral:Catalog	Filters access by the type of the associated catalog entity	All actions	AWS, Sandbox

Summary of required permissions

Summary of required permissions

Action	Description
partnercentral:CreateRevenueAttribution	Allows creating a new revenue attribution record

Action	Description
partnercentral:UpdateRevenueAttribution	Allows updating an existing revenue attribution record
partnercentral:GetRevenueAttribution	Allows retrieving the details of a specific revenue attribution
partnercentral:ListRevenueAttributions	Allows listing revenue attributions with optional filters
partnercentral:GetRevenueAttributionAllocation	Allows retrieving a single allocation by its ID
partnercentral:ListRevenueAttributionAllocations	Allows listing committed allocations with filtering support
partnercentral:StartRevenueAttributionAllocationsTask	Allows submitting a batch of up to 250 allocation changes for async processing
partnercentral:GetRevenueAttributionAllocationsTask	Allows retrieving the status of a previously submitted allocations task
partnercentral:CreateMarketplaceRevenueShare	Allows creating a new marketplace revenue share resource
partnercentral:GetMarketplaceRevenueShare	Allows retrieving details of a specific marketplace revenue share
partnercentral:ListMarketplaceRevenueShares	Allows listing marketplace revenue shares with optional filters
partnercentral:CreateMarketplaceRevenueShareAllocation	Allows creating a new marketplace revenue share allocation
partnercentral:GetMarketplaceRevenueShareAllocation	Allows retrieving details of a specific marketplace revenue share allocation
partnercentral:UpdateMarketplaceRevenueShareAllocation	Allows updating an existing marketplace revenue share allocation

Action	Description
partnercentral:ListMarketplaceRevenueShareAllocations	Allows listing allocations under a marketplace revenue share
partnercentral:TagResource	Allows adding or overwriting tags for a resource
partnercentral:UntagResource	Allows removing tags from a resource
partnercentral:ListTagsForResource	Allows listing tags associated with a resource

Quotas for the AWS Partner Central API

The following sections provide information about service quotas.

Topics

- Quotas for the AWS Partner Central Account API
- Quotas for the AWS Partner Central Selling API
- Quotas for the AWS Partner Central Benefits API
- Quotas for the AWS Partner Central Channel API
- Quotas for the AWS Partner Central Revenue Measurement API

Quotas for the AWS Partner Central Account API

The AWS Partner Central Account API has the following quotas.

Additional quotas

Additional quotas

Display name	Catalog	Description	Default value
Open connection invitations per account	AWS	The maximum number of open connection invitations you can maintain	1,000

Display name	Catalog	Description	Default value
		with partner accounts in the AWS catalog	
Active connections per account	AWS	The maximum number of active connections you can maintain with partner accounts in the AWS catalog	1,000
Email domains per partner	AWS	The maximum number of email domains that can be associated with a partner account for AWS training certification in the AWS catalog	50
Rate of connectio n invitations per account	AWS	The maximum number of connectio n invitations per day that you can send in the AWS catalog	50
Rate of profile update tasks per account	AWS	The maximum number of profile update tasks per day that you can initiate in the AWS catalog	10

Display name	Catalog	Description	Default value
Rate of profile visibility updates per account	AWS	The maximum number of profile visibility updates per day that you can initiate in the AWS catalog	5
Rate of email verification codes per email address	AWS	The maximum number of email verification codes per day that you can request in the AWS catalog	5
Open connection invitations per account	Sandbox	The maximum number of open connection invitations you can maintain with partner accounts in the Sandbox catalog	1,000
Active connections per account	Sandbox	The maximum number of active connections you can maintain with partner accounts in the Sandbox catalog	1,000

Display name	Catalog	Description	Default value
Email domains per partner	Sandbox	The maximum number of email domains that can be associated with a partner account for AWS training certification in the Sandbox catalog	50
Rate of connection invitations per account	Sandbox	The maximum number of connection invitations per day that you can send in the Sandbox catalog	50
Rate of profile update tasks per account	Sandbox	The maximum number of profile update tasks per day that you can initiate in the Sandbox catalog	10
Rate of profile visibility updates per account	Sandbox	The maximum number of profile visibility updates per day that you can initiate in the Sandbox catalog	5

Display name	Catalog	Description	Default value
Rate of email verification codes per email address	Sandbox	The maximum number of email verification codes per day that you can request in the Sandbox catalog. No emails are sent from Sandbox catalog.	N/A

Understanding and managing quotas

Rate limiting

When an API rate limit is reached, the service will respond with a ThrottlingException. To better handle rate limiting, AWS recommends implementing exponential backoff and retry strategies in your application.

Requesting a quota increase

If the default quotas do not meet your requirements, you can request a quota increase through the Service Quotas page. The Service Quotas console is a browser-based interface that you can use to view and manage your service quotas. You can access Service Quotas from any AWS Management Console page by choosing it on the top navigation bar, or by searching for Service Quotas in the AWS Management Console.

Quotas for the AWS Partner Central Selling API

AWS Partner Central selling API enforces quotas to ensure fair usage and to protect the service from misuse. Below are the detailed quotas for various API operations and associations per opportunity.

API operation quotas

Type	API operation	Quota (per partner account)
Read actions	GetOpportunity	10 per second; 100,000 per 24 hours

Type	API operation	Quota (per partner account)
	GetAwsOpportunitySummary	
	ListOpportunities	
	ListSolutions	
	GetEngagementInvitation	
	ListEngagementInvitations	
Prospecting read actions	GetProspectingFrom EngagementTask	100 per second
Prospecting read actions	ListProspectingFro mEngagementTasks	50 per second
Write actions	CreateOpportunity	1 per second; 10,000 per 24 hours
	UpdateOpportunity	
	AssociateOpportunity	
	DisassociateOpportunity	
	RejectEngagementInvitation	
	AssignOpportunity	
	StartEngagementFro mOpportunityTask	
	StartEngagementByA cceptingInvitationTask	
Prospecting write actions	StartProspectingFr omEngagementTask	2 per second
Engagement write actions	CreateEngagement	15 per second

Association quotas per opportunity

Related entity	Quota
AWS products	20 per opportunity
Partner Solutions	10 per opportunity
AWS Marketplace solutions	10 per opportunity
AWS Marketplace products	10 per opportunity
AWS Marketplace private offers	1 per opportunity

Understanding and managing quotas

Rate limiting

When an API rate limit is reached, the service will respond with a ThrottlingException. To better handle rate limiting, AWS recommends implementing exponential backoff and retry strategies in your application.

Time window for quotas

The daily quotas reset on a rolling 24 hour period. Your requests would be throttled e.g. if you have performed 10,000 write actions in the last 24 hours and are trying to perform the 10,001st request. Ensure that your application's usage patterns take this into account to prevent unintentional throttling.

Requesting a quota increase

If the default quotas do not meet your requirements, you can request a quota increase through the Service Quotas page. The Service Quotas console is a browser-based interface that you can use to view and manage your service quotas. You can access Service Quotas from any AWS Management Console page by choosing it on the top navigation bar, or by searching for Service Quotas in the AWS Management Console.

Quotas for the AWS Partner Central Benefits API

The AWS Partner Central Benefits API has the following quotas.

Request quotas

Request quotas

API operations	Quota (per AWS account)
ListBenefits	20 per second
GetBenefit	20 per second
AmendBenefitApplication	10 per second
CancelBenefitApplication	10 per second
CreateBenefitApplication	10 per second
GetBenefitApplication	20 per second
ListBenefitApplications	30 per second
RecallBenefitApplication	10 per second
SubmitBenefitApplication	10 per second
UpdateBenefitApplication	10 per second
GetBenefitAllocation	20 per second
ListBenefitAllocations	20 per second
AssociateBenefitApplicationResource	10 per second
DisassociateBenefitApplicationResource	10 per second

Additional quotas

Additional quotas

Display name	Catalog	Description	Default value
Benefit applications	AWS	The maximum number of benefit	10,000

Display name	Catalog	Description	Default value
		applications you can create in the AWS catalog	
Benefit applications	Sandbox	The maximum number of benefit applications you can create in the Sandbox catalog	10,000

Understanding and managing quotas

Rate limiting

When an API rate limit is reached, the service will respond with a ThrottlingException. To better handle rate limiting, AWS recommends implementing exponential backoff and retry strategies in your application.

Requesting a quota increase

If the default quotas do not meet your requirements, you can request a quota increase through the Service Quotas page. The Service Quotas console is a browser-based interface that you can use to view and manage your service quotas. You can access Service Quotas from any AWS Management Console page by choosing it on the top navigation bar, or by searching for Service Quotas in the AWS Management Console.

Quotas for the AWS Partner Central Channel API

The AWS Partner Central Channel API has the following quotas.

Request quotas

Request quotas

API operations	Quota (per AWS account)
CreateProgramManagementAccount	3 per second

API operations	Quota (per AWS account)
UpdateProgramManagementAccount	3 per second
ListProgramManagementAccounts	5 per second
DeleteProgramManagementAccount	3 per second
CreateChannelHandshake	3 per second
ListChannelHandshakes	5 per second
AcceptChannelHandshake	3 per second
RejectChannelHandshake	3 per second
CancelChannelHandshake	3 per second
CreateRelationship	3 per second
GetRelationship	5 per second
ListRelationships	5 per second
UpdateRelationship	3 per second
DeleteRelationship	3 per second

Additional quotas

Additional quotas

Display name	Catalog	Description	Default value
Program management accounts	AWS	The maximum number of program management accounts you can create in the AWS catalog	50

Display name	Catalog	Description	Default value
Relationships per program managemen t account	AWS	The maximum number of relations hips you can establish per program management account in the AWS catalog	1,000
Program managemen t account handshakes per account	AWS	The maximum number of active program managemen t account handshakes in the AWS catalog	100
Start service period handshakes per account	AWS	The maximum number of active handshakes for establishing service periods in the AWS catalog	200
Revoke service period handshakes per account	AWS	The maximum number of active handshakes for revoking service periods in the AWS catalog	200
Rate of program management account handshakes per day	AWS	The maximum number of program management account handshakes you can send to the same AWS account per day in the AWS catalog	5

Display name	Catalog	Description	Default value
Rate of start service period handshakes per day	AWS	The maximum number of start service period handshakes you can send to the same AWS account per day in the AWS catalog	5
Rate of revoke service period handshakes per day	AWS	The maximum number of revoke service period handshakes you can send to the same AWS account per day in the AWS catalog	5
Program management accounts	Sandbox	The maximum number of program management accounts you can create in the Sandbox catalog	50
Relationships per program management account	Sandbox	The maximum number of relationships you can establish per program management account in the Sandbox catalog	1,000

Display name	Catalog	Description	Default value
Program management account handshakes per account	Sandbox	The maximum number of active program management account handshakes in the Sandbox catalog	100
Start service period handshakes per account	Sandbox	The maximum number of active handshakes for establishing service periods in the Sandbox catalog	200
Revoke service period handshakes per account	Sandbox	The maximum number of active handshakes for revoking service periods in the Sandbox catalog	200
Rate of program management account handshakes per day	Sandbox	The maximum number of program management account handshakes you can send to the same AWS account per day in the Sandbox catalog	5

Display name	Catalog	Description	Default value
Rate of start service period handshakes per day	Sandbox	The maximum number of start service period handshakes you can send to the same AWS account per day in the Sandbox catalog	5
Rate of revoke service period handshakes per day	Sandbox	The maximum number of revoke service period handshakes you can send to the same AWS account per day in the Sandbox catalog	5

Understanding and managing quotas

Rate limiting

When an API rate limit is reached, the service will respond with a ThrottlingException. To better handle rate limiting, AWS recommends implementing exponential backoff and retry strategies in your application.

Requesting a quota increase

If the default quotas do not meet your requirements, you can request a quota increase through the Service Quotas page. The Service Quotas console is a browser-based interface that you can use to view and manage your service quotas. You can access Service Quotas from any AWS Management Console page by choosing it on the top navigation bar, or by searching for Service Quotas in the AWS Management Console.

Quotas for the AWS Partner Central Revenue Measurement API

AWS Partner Central Revenue Measurement API enforces quotas to ensure fair usage and to protect the service from misuse. Below are the detailed quotas for various API operations.

API operation quotas

Type	API operation	Quota (per partner account)
Read actions	GetRevenueAttribution	50 per second
	GetRevenueAttributionAllocation	
	GetRevenueAttributionAllocationsTask	
	GetMarketplaceRevenueShare	
	GetMarketplaceRevenueShareAllocation	
List actions	ListRevenueAttributions	10 per second
	ListRevenueAttributionAllocations	
	ListMarketplaceRevenueShares	
	ListMarketplaceRevenueShareAllocations	
	ListTagsForResource	
Write actions	CreateRevenueAttribution	2 per second
	UpdateRevenueAttribution	

Type	API operation	Quota (per partner account)
	CreateMarketplaceRevenueShare	
	CreateMarketplaceRevenueShareAllocation	
	UpdateMarketplaceRevenueShareAllocation	
	TagResource	
	UntagResource	
Async write actions	StartRevenueAttributionAllocationsTask	1 per second

Understanding and managing quotas

Rate limiting

When an API rate limit is reached, the service will respond with a ThrottlingException. To better handle rate limiting, AWS recommends implementing exponential backoff and retry strategies in your application.

Requesting a quota increase

If the default quotas do not meet your requirements, you can request a quota increase through the Service Quotas page. The Service Quotas console is a browser-based interface that you can use to view and manage your service quotas. You can access Service Quotas from any AWS Management Console page by choosing it on the top navigation bar, or by searching for Service Quotas in the AWS Management Console.

Logging for the AWS Partner Central API

The following sections provide information about logging.

Topics

- Logging the AWS Partner Central Account API
- Logging the AWS Partner Central Selling API
- Logging the AWS Partner Central Benefits API
- Logging the AWS Partner Central Channel API
- Logging the AWS Partner Central Revenue Attribution API

Logging the AWS Partner Central Account API

AWS CloudTrail is a service that enables governance, compliance, operational auditing, and risk auditing of your AWS account. With AWS CloudTrail, you can log, continuously monitor, and retain account activity related to actions across your AWS infrastructure. AWS Partner Central Account API activity is recorded as events in CloudTrail. You can create a trail, a configuration that enables delivery of events as log files to an Amazon S3 bucket.

Overview

The AWS Partner Central Account API is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Partner Central. CloudTrail captures all API calls for AWS Partner Central Account API as events. The calls captured include calls from the AWS Partner Central and from code calls to the AWS Partner Central Account API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Partner Central Account API. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in Event history.

Using the information collected by CloudTrail, you can determine the request that was made to AWS Partner Central Account API, the IP address from which the request was made, who made the request, when it was made, and additional details.

Understanding AWS Partner Central Account API log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket. When your trail tracks AWS Partner Central Account API events, CloudTrail processes the events as log files across all the regions. Each log file can contain one or more events.

The following example shows a CloudTrail log entry that demonstrates the `CreatePartner` action on AWS Partner Central Account API:

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/CloudTrailTestUser",
        "accountId": "123456789012",
        "accessKeyId": "ABCDEFGHIJKLMNOP1234",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAEXAMPLE52AGFT725JGDZ",
                "arn": "arn:aws:iam::123456789012:role/ExampleRole",
                "accountId": "123456789012",
                "userName": "ExampleRole"
            },
            "attributes": {
                "creationDate": "2025-11-07T19:45:28Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2025-11-07T19:45:58Z",
    "eventSource": "partnercentral-account.amazonaws.com",
    "eventName": "CreatePartner",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "PostmanRuntime/7.18.0",
    "requestParameters": {
        "catalog": "AWS",
        "clientToken": "abcdef12-3456-7890-bcde-f123456789ab",
        "legalName": "ExampleLegalName",
        "primarySolutionType": "VALUE_ADDED_RESALE_AWS_SERVICES",
        "allianceLeadContact": {
            "firstName": "ExampleFirstName",
            "lastName": "ExampleLastName",
            "email": "example@domain.com",
            "businessTitle": "ExampleBusinessTitle"
        },
        "emailVerificationCode": "ExampleVerificationCode",
        "tags": [
            {
                "key": "office",
```

```
                "value": "1"
            }
        ]
    },
    "responseElements": {
        "catalog": "AWS",
        "arn": "arn:aws:partnercentral:us-east-1:123456789012:catalog/AWS/partner/
EXAMPLE_PARTNER_ID",
        "id": "EXAMPLE_PARTNER_ID",
        "legalName": "ExampleLegalName",
        "createdAt": "2025-11-07T19:45:57.867007329Z",
        "profile": {
            "primarySolutionType": "VALUE_ADDED_RESALE_AWS_SERVICES"
        },
        "allianceLeadContact": {
            "firstName": "ExampleFirstName",
            "lastName": "ExampleLastName",
            "email": "example@domain.com",
            "businessTitle": "ExampleBusinessTitle"
        }
    },
    "requestID": "12345678-1234-5678-9abc-def012345678",
    "eventID": "87654321-4321-8765-cba9-fed098765432",
    "readOnly": false,
    "resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::PartnerCentral::Partner",
            "ARN": "arn:aws:partnercentral:us-east-1:123456789012:catalog/AWS/partner/
EXAMPLE_PARTNER_ID"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "partnercentral-account.partner.us-east-1.api.aws"
    }
}
```

In this example, the `CreatePartner` action was called by the IAM user named CloudTrailTestUser. The request was made on November 7, 2025 at 19:45:58 UTC. The request created a new partner with ID `EXAMPLE_PARTNER_ID` for the AWS catalog with the legal name "ExampleLegalName".

Fields in AWS Partner Central Account API log file entries

Each entry in a CloudTrail log file contains information about who made a request, the resources acted upon in the request, and the response elements returned by AWS Partner Central Account API. The list of fields in a log entry, such as `eventVersion`, `userIdentity`, and `eventTime`, provide detailed information about the action. For example, the `sourceIPAddress` field shows the IP address that the request was made from.

Logging the AWS Partner Central Selling API

AWS CloudTrail is a service that enables governance, compliance, operational auditing, and risk auditing of your AWS account. With AWS CloudTrail, you can log, continuously monitor, and retain account activity related to actions across your AWS infrastructure. AWS Partner Central API activity is recorded as events in CloudTrail. You can create a trail, a configuration that enables delivery of events as log files to an Amazon S3 bucket.

Overview

The AWS Partner Central API is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Partner Central. CloudTrail captures all API calls for AWS Partner Central as events. The calls captured include calls from the AWS Partner Central and from code calls to the AWS Partner Central API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Partner Central. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in Event history.

Using the information collected by CloudTrail, you can determine the request that was made to AWS Partner Central, the IP address from which the request was made, who made the request, when it was made, and additional details.

Understanding AWS Partner Central Selling API log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket. When your trail tracks AWS Partner Central events, CloudTrail processes the events as log files across all the regions. Each log file can contain one or more events.

The following example shows a CloudTrail log entry that demonstrates the `ListOpportunities` action on AWS Partner Central:


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "ABCDEFGHIJKLMNOP12345",
        "arn": "arn:aws:iam::123456789010:user/CloudTrailTestUser",
        "accountId": "123456789010",
        "accessKeyId": "ABCDEFGHIJKLMNOP1234",
        "userName": "CloudTrailTestUser"
    },
    "eventTime": "2023-10-17T21:49:23Z",
    "eventSource": "partnercentral-selling.amazonaws.com",
    "eventName": "ListOpportunities",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "PostmanRuntime/7.18.0",
    "requestParameters": {
        "MaxResults": 20
    },
    "responseElements": null,
    "requestID": "fEXAMPLE-cb3e-4e21-86fd-6b3EXAMPLEd1",
    "eventID": "7EXAMPLE-97d6-4139-91e3-01aEXAMPLE48",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789010"
}
```


In this example, the `ListOpportunities` action was called by the IAM user named CloudTrailTestUser. The action was called in the *us-east-1* AWS Region, and the request was made on October 17, 2023 at 21:49:23 UTC.

Fields in AWS Partner Central Selling API log file entries

Each entry in a CloudTrail log file contains information about who made a request, the resources acted upon in the request, and the response elements returned by AWS Partner Central. The list of fields in a log entry, such as `eventVersion`, `userIdentity`, and `eventTime`, provide detailed information about the action. For example, the `sourceIPAddress` field shows the IP address that the request was made from.

Logging the AWS Partner Central Benefits API

AWS CloudTrail is a service that enables governance, compliance, operational auditing, and risk auditing of your AWS account. With AWS CloudTrail, you can log, continuously monitor, and retain account activity related to actions across your AWS infrastructure. AWS Partner Central Benefits API activity is recorded as events in CloudTrail. You can create a trail, a configuration that enables delivery of events as log files to an Amazon S3 bucket.

Overview

The AWS Partner Central Benefits API is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Partner Central. CloudTrail captures all API calls for AWS Partner Central Benefits API as events. The calls captured include calls from the AWS Partner Central and from code calls to the AWS Partner Central Benefits API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Partner Central Benefits API. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in Event history.

Using the information collected by CloudTrail, you can determine the request that was made to AWS Partner Central Benefits API, the IP address from which the request was made, who made the request, when it was made, and additional details.

Understanding AWS Partner Central Benefits API log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket. When your trail tracks AWS Partner Central Benefits API events, CloudTrail processes the events as log files across all the regions. Each log file can contain one or more events.

The following example shows a CloudTrail log entry that demonstrates the `ListBenefitApplications` action on AWS Partner Central Benefits API:

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/CloudTrailTestUser",
        "accountId": "123456789012",
```

```
        "accessKeyId": "EXAMPLE_KEY_ID",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::123456789012:role/TestRole",
                "accountId": "123456789012",
                "userName": "TestRole"
            },
            "attributes": {
                "creationDate": "2025-10-21T03:08:23Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2025-10-21T03:10:59Z",
    "eventSource": "partnercentral-benefits.amazonaws.com",
    "eventName": "ListBenefitApplications",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "python-requests/2.32.4",
    "requestParameters": {
        "catalog": "AWS"
    },
    "responseElements": null,
    "requestID": "12345678-1234-5678-9abc-def012345678",
    "eventID": "87654321-4321-8765-cba9-fed098765432",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```


In this example, the `ListBenefitApplications` action was called by the IAM user named Alice. The request was made on October 21, 2025 at 03:10:59 UTC. The request listed benefit applications for the AWS catalog.

Fields in AWS Partner Central Benefits API log file entries

Each entry in a CloudTrail log file contains information about who made a request, the resources acted upon in the request, and the response elements returned by AWS Partner Central Benefits API. The list of fields in a log entry, such as `eventVersion`, `userIdentity`, and `eventTime`,

provide detailed information about the action. For example, the `sourceIPAddress` field shows the IP address that the request was made from.

Logging the AWS Partner Central Channel API

AWS CloudTrail is a service that enables governance, compliance, operational auditing, and risk auditing of your AWS account. With AWS CloudTrail, you can log, continuously monitor, and retain account activity related to actions across your AWS infrastructure. AWS Partner Central Channel API activity is recorded as events in CloudTrail. You can create a trail, a configuration that enables delivery of events as log files to an Amazon S3 bucket.

Overview

The AWS Partner Central Channel API is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Partner Central. CloudTrail captures all API calls for AWS Partner Central Channel API as events. The calls captured include calls from the AWS Partner Central and from code calls to the AWS Partner Central Channel API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Partner Central Channel API. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in Event history.

Using the information collected by CloudTrail, you can determine the request that was made to AWS Partner Central Channel API, the IP address from which the request was made, who made the request, when it was made, and additional details.

Understanding AWS Partner Central Channel API log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket. When your trail tracks AWS Partner Central Channel API events, CloudTrail processes the events as log files across all the regions. Each log file can contain one or more events.

The following example shows a CloudTrail log entry that demonstrates the `CreateProgramManagementAccount` action on AWS Partner Central Channel API:

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAEXAMPLE52AGFT725JGDZ:example-user-Isengard",
```

```
        "arn": "arn:aws:sts::123456789012:assumed-role/Admin/example-user-Isengard",
        "accountId": "123456789012",
        "accessKeyId": "ASIAEXAMPLE52AGL6IXQLZ5",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAEXAMPLE52AGFT725JGDZ",
                "arn": "arn:aws:iam::123456789012:role/ExampleRole",
                "accountId": "123456789012",
                "userName": "ExampleRole"
            },
            "attributes": {
                "creationDate": "2025-10-21T17:06:47Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2025-10-21T17:07:26Z",
    "eventSource": "partnercentral-channel.amazonaws.com",
    "eventName": "CreateProgramManagementAccount",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "PostmanRuntime/7.18.0",
    "requestParameters": {
        "catalog": "AWS",
        "program": "SOLUTION_PROVIDER",
        "displayName": "ExampleDisplayName",
        "accountId": "987654321098",
        "clientToken": "abcdef12-3456-7890-bcde-f123456789ab"
    },
    "responseElements": {
        "programManagementAccountDetail": {
            "id": "pma-example123456789",
            "arn": "arn:aws:partnercentral:us-east-1:123456789012:catalog/AWS/program-
management-account/pma-example123456789"
        }
    },
    "requestID": "12345678-1234-5678-9abc-def012345678",
    "eventID": "87654321-4321-8765-cba9-fed098765432",
    "readOnly": false,
    "resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::PartnerCentralChannel::ProgramManagementAccount",
```

```
            "ARN": "arn:aws:partnercentral:us-east-1:123456789012:catalog/AWS/program-
management-account/pma-example123456789"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management",
    "tlsDetails": {
        "clientProvidedHostHeader": "partnercentral-channel.global.api.aws"
    }
}
```


In this example, the `CreateProgramManagementAccount` action was called by the IAM role named ExampleRole through an assumed role session. The request was made on October 21, 2025 at 17:07:26 UTC. The request created a new Program Management Account with ID `pma-example123456789` for the Solution Provider program.

Fields in AWS Partner Central Channel API log file entries

Each entry in a CloudTrail log file contains information about who made a request, the resources acted upon in the request, and the response elements returned by AWS Partner Central Channel API. The list of fields in a log entry, such as `eventVersion`, `userIdentity`, and `eventTime`, provide detailed information about the action. For example, the `sourceIPAddress` field shows the IP address that the request was made from.

Logging the AWS Partner Central Revenue Attribution API

AWS CloudTrail is a service that enables governance, compliance, operational auditing, and risk auditing of your AWS account. With AWS CloudTrail, you can log, continuously monitor, and retain account activity related to actions across your AWS infrastructure. AWS Partner Central Revenue Measurement API activity is recorded as events in CloudTrail. You can create a trail, a configuration that enables delivery of events as log files to an Amazon S3 bucket.

Overview

The AWS Partner Central Revenue Measurement API is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Partner Central. CloudTrail captures all API calls for AWS Partner Central Revenue Attribution as events. The calls captured include calls from the AWS Partner Central console and from code calls to the AWS Partner Central Revenue Measurement API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Partner Central Revenue Measurement. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**.

Using the information collected by CloudTrail, you can determine the request that was made to AWS Partner Central Revenue Measurement, the IP address from which the request was made, who made the request, when it was made, and additional details.

The following AWS Partner Central Revenue Measurement API actions are logged in CloudTrail:

- `CreateRevenueAttribution`
- `UpdateRevenueAttribution`
- `GetRevenueAttribution`
- `ListRevenueAttributions`
- `TagResource`
- `UntagResource`
- `ListTagsForResource`
- `CreateMarketplaceRevenueShare`
- `GetMarketplaceRevenueShare`
- `ListMarketplaceRevenueShares`
- `CreateMarketplaceRevenueShareAllocation`
- `GetMarketplaceRevenueShareAllocation`
- `UpdateMarketplaceRevenueShareAllocation`
- `ListMarketplaceRevenueShareAllocations`
- `StartRevenueAttributionAllocationsTask`
- `GetRevenueAttributionAllocationsTask`
- `ListRevenueAttributionAllocations`
- `GetRevenueAttributionAllocation`

Understanding AWS Partner Central Revenue Measurement API log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket. When your trail tracks AWS Partner Central Revenue Measurement events, CloudTrail processes the events as log files across all the regions. Each log file can contain one or more events.

The following example shows a CloudTrail log entry that demonstrates the `CreateRevenueAttribution` action on AWS Partner Central Revenue Attribution:


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAEXAMPLEID",
        "arn": "arn:aws:iam::123456789012:user/CloudTrailTestUser",
        "accountId": "123456789012",
        "accessKeyId": "AKIAEXAMPLEKEY",
        "userName": "CloudTrailTestUser"
    },
    "eventTime": "2026-06-02T11:53:54Z",
    "eventSource": "partnercentral-prm.amazonaws.com",
    "eventName": "CreateRevenueAttribution",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "aws-sdk-java/2.20.0",
    "requestParameters": {
        "catalog": "AWS",
        "name": "my-revenue-attribution",
        "marketplaceProduct": {
            "tenancyModel": "MULTI_TENANT"
        }
    },
    "responseElements": {
        "id": "ra-0123456789abcdef0",
        "arn": "arn:aws:partnercentral:us-east-1:123456789012:catalog/AWS/revenue-
attribution/ra-0123456789abcdef0",
        "name": "my-revenue-attribution",
        "revision": "1"
    },
    "requestID": "5ce2f907-3850-412a-b1a4-r012r1234567",
    "eventID": "80b39b72-eefe-4578-8db0-01ee2e34ee56",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```


In this example, the `CreateRevenueAttribution` action was called by the IAM user named CloudTrailTestUser. The action was called in the *us-east-1* AWS Region, and the request was made

on June 2, 2026 at 11:53:54 UTC. A new revenue attribution resource was created with the ID `ra-0123456789abcdef0`.

Fields in AWS Partner Central Revenue Measurement API log file entries

Each entry in a CloudTrail log file contains information about who made a request, the resources acted upon in the request, and the response elements returned by AWS Partner Central Revenue Measurement. The list of fields in a log entry, such as `eventVersion`, `userIdentity`, and `eventTime`, provide detailed information about the action. For example, the `sourceIPAddress` field shows the IP address that the request was made from.

Notifications for the AWS Partner Central API

The following sections provide information about notifications.

Topics

Notifications for the AWS Partner Central Account API

Partner Account Connection notifications enable partners to stay informed about connection lifecycle events that directly impact their business relationships and operational workflows. These events are critical for partners to:

- React promptly to new collaboration
- Maintain awareness of their connection portfolio status
- Take appropriate action when connections are established or terminated
- Ensure business continuity by knowing when relationships change

Topics

- Learn more about account connections API events

Complete the prerequisite to monitor events

Users require the appropriate IAM permissions to access and manage events published by the account connections API. For more information about the available actions, resources, and condition keys for EventBridge, see Using IAM policy conditions in Amazon EventBridge in the *Amazon EventBridge User Guide*. One of the condition keys is `events:detail-type`, which can be used to scope permissions to specific event types.

The following example policy demonstrates how to customize and scope the permissions for the proposed events. The `AllowPutRuleForPartnercentralAccountEvents` statement allows the creation of rules, but only for events from the `aws.partnercentral-account` source.

For detailed IAM policy examples, refer to the AWS documentation on EventBridge permissions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPutRuleForPartnerCentralAccountEvents",
      "Effect": "Allow",
      "Action": "events:PutRule",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "events:source": "aws.partnercentral-account.connection"
        }
      }
    }
  ]
}
```

Configure Amazon EventBridge to monitor events

To monitor account connections API events, you create an EventBridge rule that matches the events that you want to capture. You can use the AWS Management Console or the AWS SDKs to create and manage rules. The following sections explain how to create rules using both methods. Regardless of the method you use, you must create the rule in the US East (N. Virginia) `us-east-1` Region.

AWS Management Console setup

To set up an EventBridge rule using the AWS Management Console, follow the steps in Creating rules that react to events in Amazon EventBridge in the *Amazon EventBridge User Guide*. When creating the rule, you must set the event bus to **default**, and create the rule in the US East (N. Virginia) `us-east-1` Region.

Following is an example of an event rule:

```
{
    "source": ["aws.partnercentral-account"],
    "detail": {
        "catalog": ["AWS"]
    }
}
```

AWS SDK setup

You can use the AWS SDKs to create and manage EventBridge rules programmatically. For more information, see PutRule in the *Amazon EventBridge API Reference*.

The following example uses the AWS SDK for Python (Boto3):

```
import boto3

client = boto3.client('events', region_name='us-east-1')

response = client.put_rule(
    Name='MyConnectionInvitationReceivedRule',
    EventPattern=
    '{
        "source": ["aws.partnercentral-account"],
        "detail-type": ["Partner Connection Invitation Received"],
        "detail": {"catalog": ["AWS"]}
    }',
    State='ENABLED'
)
print('Rule ARN:', response['RuleArn'])
```

Learn more about account connections API events

The following sections describe the account connections API event types, scenarios that trigger them, and event examples.

Event types

Following are the event types and their triggers.

- Partner Connection Invitation Received: Notifies the receiver that another partner wants to establish a connection
- Partner Connection Invitation Accepted: Notifies the sender that their invitation was accepted and a connection is now active
- Partner Connection Invitation Rejected: Notifies the sender that their invitation was declined
- Partner Connection Cancelled: Notifies the other connected participant when an active connection is terminated
- Partner Connection Invitation Cancelled: Notifies the receiver that the sender has withdrawn their invitation before any action was taken
- Partner Connection Invitation Expired: Notifies both sender and receiver that the invitation has expired

Example events

The following sections provide examples of the events listed earlier in the previous section.

Topics

Partner Connection Invitation Received

Triggering Context

This event is generated when a connection invitation is successfully created and persisted in PAC's data store via the CreateConnectionInvitation API. The event is sent immediately after the invitation creation completes successfully and is persisted in PAC's data store, not just when the API call is made.

The event will be automatically sent when a new connection invitation is created. One event per invitation created with no duplicate events for the same invitation.

Recipients

The AWS account of the receiver in the connection invitation.

Expected handling

Typical Customer Actions:

- Immediate Notification: Trigger alerts to business teams about new partnership opportunities
- Workflow Automation: Automatically update partner management systems with pending invitations
- Decision Support: Route invitations to appropriate decision makers based on connection type
- Audit Logging: Record invitation receipt for compliance and partnership tracking
- Response Automation: For trusted partners, potentially auto-accept certain invitation types

Common Integration Patterns:

- Lambda function → Update partner portal dashboard
- SQS queue → Batch process invitations for review
- SNS topic → Email/Slack notifications to business teams
- API destination → Webhook to external CRM/partner management systems

Example

```
{
  "version": "1",
```

```
  "id": "<event id>",
  "detail-type": "Partner Connection Invitation Received",
  "source": "aws.partnercentral-account",
  "time": "<ISO 8601 date time>",
  "region": "us-east-1",
  "account": "<corresponding partner AWS account>",
  "resources": [ "<<Connection Invitation ARN>>" ],
  "detail": {
    "catalog": "AWS",
    "connectionInvitation" :{
        "arn": "<<Connection Invitation ARN>>",
        "id": "pacinv-****",
        "connectionType": "OpportunityCollaboration",
        "inviterEmail": "abc@def.com",
        "invitationMessage": "We'd like to collaborate on a joint solution for cloud
 security. Please accept this connection invitation to proceed.",
        "senderCompanyName": "<<Sender Partner Account Name>>",
        "senderProfileId": "pprofile-****",
        "expiresAt": "<ISO 8601 date time>"
    }
  }
}
```

Partner Connection Invitation Accepted

Triggering Context

This event is generated when a connection invitation is successfully accepted and a connection is created and persisted in PAC's data store via the AcceptConnectionInvitation API. The event is sent immediately after the invitation acceptance completes successfully and the connection is established.

The event will be automatically sent when a connection invitation is accepted. One event per invitation accepted with no duplicate events for the same acceptance.

Recipients

Both AWS accounts involved in the connection invitation: the sender account that originally sent the invitation and the receiver account that accepted the invitation.

Expected handling

Typical Customer Actions:

- Partnership Activation: Trigger workflows to enable Layer 2 business activities
- Status Updates: Update partner management systems with active connection status
- Notification: Alert business teams about successful partnership establishment
- Access Provisioning: Automatically grant appropriate permissions for collaboration
- Analytics: Track partnership conversion rates and success metrics

Common Integration Patterns:

- Lambda function → Enable data sharing permissions and update partner portal
- SQS queue → Batch process connection activations for business setup
- SNS topic → Email/Slack notifications to business and technical teams
- API destination → Webhook to CRM systems to update partnership status

Example

```
{
  "version": "1",
  "id": "<event id>",
  "detail-type":  "Partner Connection Invitation Accepted",
  "source": "aws.partnercentral-account",
  "time": "<ISO 8601 date time>",
  "region": "us-east-1",
  "resources": [ "<<Connection Invitation ARN>>" , "<<Connection ARN>>" ],
  "account": "<corresponding partner AWS account>",
  "detail": {
    "catalog": "AWS",
    "connectionInvitation" :{
        "arn": "<<Connection Invitation ARN>>",
        "id": "pacinv-****",
        "connectionType": "OpportunityCollaboration"
    },
    "connection": {
        "arn": "<<Connection ARN>>",
         "id": "pac-*****",
         "Participant1AccountId": "<<Sender AWS AccountId>>",
         "Participant2AccountId": "<<Receiver AWS AccountId>>",
         "status": "ACTIVE"
     }
  }
```

```
}
```

Partner Connection Invitation Rejected

Triggering Context

This event is generated when a connection invitation is successfully rejected via the RejectConnectionInvitation API. The event is sent immediately after the invitation rejection is processed and persisted in PAC's data store.

The event will be automatically sent when a connection invitation is rejected. One event per invitation rejected with no duplicate events for the same rejection.

Recipients

The AWS account that originally sent the connection invitation (the sender account).

Expected handling

Typical Customer Actions:

- Status Updates: Update partner management systems with rejection status
- Follow-up Actions: Trigger workflows for alternative partnership approaches
- Notification: Alert business teams about partnership decision
- Cleanup: Remove pending invitation references from internal systems

Common Integration Patterns:

- Lambda function → Update partner portal and trigger follow-up workflows
- SQS queue → Batch process rejections for business analysis
- SNS topic → Email/Slack notifications to business development teams
- API destination → Webhook to CRM systems to update opportunity status

Example

```
{
  "version": "1",
  "id": "<event id>",
```

```
  "detail-type": "Partner Connection Invitation Rejected",
  "source": "aws.partnercentral-account",
  "time": "<ISO 8601 date time>",
  "region": "us-east-1",
  "account": "<corresponding partner AWS account>",
  "resources": [ "<<Connection Invitation ARN>>" ],
  "detail": {
    "catalog": "AWS",
    "connectionInvitation" :{
        "arn": "<<Connection Invitation ARN>>",
        "id": "pacinv-****",
        "connectionType": "OpportunityCollaboration",
        "invitationMessage": "We'd like to collaborate on a joint solution for cloud
 security. Please accept this connection invitation to proceed.",
        "receiverProfileId": "pprofile-****"
    }
  }
}
```

Partner Connection Cancelled

Triggering Context

This event is generated when an active connection is successfully cancelled and the active status is updated to terminated in PAC's data store via the CancelConnection API. The event is sent immediately after the connection cancellation is processed and the connection status is updated.

The event will be automatically sent when a connection is terminated. One event per connection cancelled with no duplicate events for the same cancellation.

Recipients

The AWS account of the other participant in the connection (not the one who initiated cancellation).

Expected handling

Typical Customer Actions:

- Access Revocation: Immediately revoke permissions and disable data sharing
- Status Updates: Update partner management systems with terminated connection status
- Notification: Alert business and technical teams about partnership termination

- Cleanup: Remove connection references and clean up shared resources
- Analytics: Track connection duration and termination patterns

Common Integration Patterns:

- Lambda function → Revoke permissions and update partner portal status
- SQS queue → Batch process connection terminations for cleanup workflows
- SNS topic → Email/Slack notifications to business and technical teams
- API destination → Webhook to CRM systems to update partnership status

Example

```
{
  "version": "1",
  "id": "<event id>",
  "detail-type": "Partner Connection Cancelled",
  "source": "aws.partnercentral-account",
  "time": "<ISO 8601 date time>",
  "region": "us-east-1",
  "resources": [ "<<Connection ARN>>" ],
  "account": "<corresponding partner AWS account>",
  "detail": {
    "catalog": "AWS",
    "connection" :{
        "arn": "<<Connection Invitation ARN>>",
        "id": "pac-****",
        "connectionType": "OpportunityCollaboration",
        "canceledBy": "<<AWS account ID>>"
    }
  }
}
```

Partner Connection Invitation Cancelled

Triggering Context

This event is generated when a pending connection invitation is successfully cancelled via the CancelConnectionInvitation API. The event is sent immediately after the invitation cancellation is processed and the invitation status is updated to CANCELED.

The event will be automatically sent when a connection invitation is canceled. One event per connection invitation cancelled with no duplicate events for the same cancellation.

Recipients

The AWS account that was supposed to receive the connection invitation (the receiver account).

Expected handling

Typical Customer Actions:

- Status Updates: Update partner management systems to remove pending invitation references
- Notification: Alert business teams that a potential partnership opportunity was withdrawn
- Cleanup: Remove invitation references from internal tracking systems
- Analytics: Track invitation cancellation patterns for partnership insights

Common Integration Patterns:

- Lambda function → Update partner portal and remove pending invitation notifications
- SQS queue → Batch process cancellations for business analysis
- SNS topic → Email/Slack notifications to business development teams
- API destination → Webhook to CRM systems to update opportunity status

Example

```
{
  "version": "1",
  "id": "<event id>",
  "detail-type": "Partner Connection Invitation Cancelled",
  "source": "aws.partnercentral-account",
  "time": "<ISO 8601 date time>",
  "region": "us-east-1",
  "account": "<corresponding partner AWS account>",
  "resources": [ "<<Connection Invitation ARN>>" ],
  "detail": {
    "catalog": "AWS",
    "connectionInvitation" :{
        "arn": "<<Connection Invitation ARN>>",
        "id": "pacinv-****",
```

```
        "connectionType": "OpportunityCollaboration",
        "invitationMessage": "We'd like to collaborate on a joint solution for cloud
 security. Please accept this connection invitation to proceed.",
        "senderProfileId": "pprofile-****"
    }
  }
}
```

Partner Connection Invitation Expired

Triggering Context

This event is generated when a pending connection invitation automatically expires after reaching its ExpiresAt timestamp without being accepted or rejected by the receiver. The event is automatically triggered by the system when the invitation expiration time is reached.

The event will be automatically sent when a connection invitation expires. One event per invitation expired with no duplicate events for the same expiration.

Recipients

Both AWS accounts involved in the connection invitation: the sender account that originally sent the invitation and the receiver account that received the invitation.

Expected handling

Typical Customer Actions:

- Status Updates: Update partner management systems to mark invitation as expired
- Follow-up Actions: Trigger workflows for re-engagement or alternative partnership approaches
- Notification: Alert business teams about missed partnership opportunities
- Cleanup: Remove expired invitation references from internal tracking systems

Common Integration Patterns:

- Lambda function → Update partner portal and trigger follow-up workflows
- SQS queue → Batch process expirations for business analysis
- SNS topic → Email/Slack notifications to business development teams
- API destination → Webhook to CRM systems to update opportunity status

Example

```
{
  "version": "1",
  "id": "<event id>",
  "detail-type": "Partner Connection Invitation Expired",
  "source": "aws.partnercentral-account",
  "time": "<ISO 8601 date time>",
  "region": "us-east-1",
  "resources": [ "<<Connection Invitation ARN>>" ],
  "detail": {
    "catalog": "AWS",
    "connectionInvitation": {
        "arn": "<<Connection Invitation ARN>>",
        "id": "pacinv-****",
        "connectionType": "OpportunityCollaboration",
        "inviterEmail": "abc@def.com",
        "invitationMessage": "We'd like to collaborate on a joint solution for cloud
 security. Please accept this connection invitation to proceed.",
        "senderProfileId": "pprofile-****",
        "receiverProfileId": "pprofile-****"
    }
  }
}
```

Notifications for the AWS Partner Central Selling API

Events for the selling API provide real-time notifications about changes in the status or the details of opportunities. These events help keep your systems in sync with AWS Partner Central, and help ensure timely responses and updates.

Topics

- Complete the prerequisite to monitor events
- Configure Amazon EventBridge to monitor events
- Learn more about selling API events

Complete the prerequisite to monitor events

Users require the appropriate IAM permissions to access and manage events published by the Partner Central selling API. For more information about the available actions, resources, and

condition keys for EventBridge, see Using IAM policy conditions in Amazon EventBridge in the *Amazon EventBridge User Guide*. One of the condition keys is `events:detail-type`, which can be used to scope permissions to specific event types.

The following example policy demonstrates how to customize and scope the permissions for the proposed events. The `AllowPutRuleForPartnercentralSellingEvents` statement allows the creation of rules, but only for events from the `aws.partnercentral-selling` source.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "AllowPutRuleForPartnercentralSellingEvents",
    "Effect": "Allow",
    "Action": "events:PutRule",
    "Resource": "*",
    "Condition": {
      "ForAllValues:StringEquals": {
        "events:source": "aws.partnercentral-selling"
      }
    }
  }
}
```

Configure Amazon EventBridge to monitor events

To monitor selling API events, you create an EventBridge rule that matches the events that you want to capture. You can use the AWS Management Console or the AWS SDKs to create and manage rules. The following sections explain how to create rules using both methods. Regardless of the method you use, you must create the rule in the US East (N. Virginia) `us-east-1` Region.

AWS Management Console setup

To set up an EventBridge rule using the AWS Management Console, follow the steps in Creating rules that react to events in Amazon EventBridge in the *Amazon EventBridge User Guide*. When creating the rule, you must set the event bus to **default**, and create the rule in the US East (N. Virginia) `us-east-1` Region.

Following is an example of an event rule:

```
{
    "source": ["aws.partnercentral-selling"],
    "detail": {
        "catalog": ["AWS"]
    }
}
```

AWS SDK setup

You can use the AWS SDKs to create and manage EventBridge rules programmatically. For more information, see PutRule in the *Amazon EventBridge API Reference*.

The following example uses the AWS SDK for Python (Boto3):

```
import boto3

client = boto3.client('events', region_name='us-east-1')

response = client.put_rule(
    Name='MyOpportunityCreatedRule',
    EventPattern=
    '{
        "source": ["aws.partnercentral-selling"],
        "detail-type": ["Opportunity Created"],
        "detail": {"catalog": ["AWS"]}
    }',
    State='ENABLED'
)
print('Rule ARN:', response['RuleArn'])
```

Learn more about selling API events

The following sections describe the selling API event types, scenarios that trigger them, and event examples.

Event types

Following are the event types and their triggers.

- Opportunity Created: Triggered when a new opportunity is created.
- Opportunity Updated: Triggered when an opportunity (Opportunity or its corresponding AWS Opportunity Summary) is updated.

- Engagement Invitation Created: Triggered when an AWS Referral invitation is created.
- Engagement Invitation Accepted: Triggered when a partner accepts an AWS Engagement Invitation, confirming their interest in collaborating with AWS on the opportunity.
- Engagement Invitation Rejected: Triggered when a partner rejects an AWS Engagement Invitation.
- Engagement Invitation Expired: This event is triggered when a Partner rejects an EngagementInvitation. It notifies the sending and receiving partner that the invitation has expired.
- Engagement Member Added: This event is triggered when a new member joins the Engagement after accepting an invitation. It notifies all current members of the Engagement about the new member being added to the Engagement.
- Engagement Resource Snapshot Created: This event is triggered when the new revision of the Snapshot of the resources (such as opportunities) associated with an Engagement is created. It notifies all the current members of the Engagement about the changes to the associated resource Snapshot.
- Engagement Created: Triggered when a new engagement is created.
- Engagement Updated: Triggered when an engagement is updated.

Event scenarios

The following table describes how events operate under different scenarios. The following assumptions are made for these scenarios:

- AWS is also a partner in these scenarios.
- `ResourceSnapshotJob` is configured correctly by the partner.
- The fields updated on the opportunity are also on the opportunity template for Resource Snapshot.

Action by you as a partner	Events received	Participant type
You create an opportunity	Opportunity Created	
You use `StartEnga gementFromOpportun`	Opportunity Updated, Engagement Created, Engagement Resource	

Action by you as a partner	Events received	Participant type
`ityTask` to submit an opportunity	Snapshot Created, Engagement Member Added, Engagement Invitation Created	
AWS approves your submitted opportunity	Engagement Invitation Accepted, Engagement Member Added, Opportuni ty Updated, Engagement Resource Snapshot Created	
AWS rejects your submitted opportunity	Engagement Invitation Rejected, Opportunity Updated, Engagement Resource Snapshot Created	
You associate an AWS Marketplace offer with an opportunity	Opportunity Updated	
You associate a solution with an opportunity	Opportunity Updated, Engagement Resource Snapshot Created	
You update an opportunity	Opportunity Updated, Engagement Resource Snapshot Created (Optional if `ResourceSnapshotJob` is set up and the field updates are on the template)	
AWS updates your opportuni ty	Opportunity Updated, Engagement Resource Snapshot Created	
You invite another partner to an engagement	Engagement Invitation Created	Sender

Action by you as a partner	Events received	Participant type
Your invitation to join an engagement is accepted by a partner	Engagement Invitation Accepted, Engagement Member Added	Sender
Your invitation to join an engagement is rejected by a partner	Engagement Invitation Rejected	Sender
Your invitation to join an engagement is not acted on by a partner for 15 days	Engagement Invitation Expired	Sender
You receive an invitation from another partner to join an engagement	Engagement Invitation Created	Receiver
You accept the invitation from another partner to join an engagement via `StartEngagementByAcceptingInvitationTask`	Engagement Invitation Accepted, Engagement Member Added, Opportunity Created, Opportunity Updated, Engagement Resource Snapshot Created	Receiver
You reject the invitation to join an engagement from another partner	Engagement Invitation Rejected	Receiver
You do not act on invitation to join an engagement from another partner for 15 days	Engagement Invitation Expired	Receiver
You create an engagement	Engagement Created	Sender
You update an engagement	Engagement Updated	Sender, Receiver

Example events

The following sections provide examples of the events listed earlier in the previous section.

Topics

Opportunity Created

Partners use this event to:

- Notify their users that a new Opportunity has been created.
- Trigger automated workflows such as updating CRM systems or notifying sales teams about new opportunity creations.

The following example shows a typical Opportunity Created event.

```
{
    "version": "1",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "source": "aws.partnercentral-selling",
    "detail-type": "Opportunity Created",
    "time": "2023-04-16T15:23:45Z",
    "region": "us-east-1",
    "account": "123456789012",
    "detail": {
        "schemaVersion": "<version number>",
        "catalog": "<Sandbox | AWS>",
```

```
        "opportunity": {
            "identifier": "String"
        }
    }
}
```

Opportunity Updated

Partners use this event to:

- Notify their users that an existing Opportunity has been updated.
- Trigger automated workflows such as updating CRM systems or notifying sales teams.

The following example shows a typical Opportunity Updated event.

```
{
    "version": "1",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "source": "aws.partnercentral-selling",
    "detail-type": "Opportunity Updated",
    "time": "2023-04-16T15:23:45Z",
    "region": "us-east-1",
    "account": "123456789012",
    "detail": {
        "schemaVersion": "<version number>",
        "catalog": "<Sandbox | AWS>",
        "opportunity": {
             "identifier": "String"
        }
    }
}
```

Engagement Invitation Created

Partners use this event to:

- Notify their users about the new EngagementInvitation they have received.
- Trigger automated workflows to accept or reject the invitation, such as updating CRM systems or notifying sales teams.

The following example shows a typical Engagement Invitation Created event.

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Engagement Invitation Created",
    "source": "aws.partnercentral-selling",
    "account": "123456789012",
    "time": "2023-04-15T12:34:56Z",
    "region": "us-east-1",
    "resources": [
      "arn:aws:partnercentral:us-east-1::catalog/AWS/engagement-invitation/engi-
v7p8z56whnauo"
    ],
    "detail": {
      "catalog": "AWS",
      "engagementInvitation": {
        "arn": "arn:aws:partnercentral:us-east-1::catalog/AWS/engagement-invitation/
engi-v7p8z56whnauo",
        "id": "engi-v7p8z56whnauo",
        "engagementId": "eng-12345678901234",
        "senderAccountId": "string",
        "receiverAccountId": "string",
        "senderCompanyName": "string",
        "expirationDate": "string",
        "participantType": "Enum", //Sender/Receiver
  "payloadType": "LeadInvitation" //OpportunityInvitation|LeadInvitation
      }
    }
}
```

Engagement Invitation Accepted

Partners use this event to:

- Notify their users that the EngagementInvitation has been accepted.
- Update their internal records to reflect the new partner in the Engagement.
- Trigger automated workflows to synchronize data or notify other team members about the new Engagement member.

The following example shows a typical Engagement Invitation Accepted event.

```
{
```

```
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Engagement Invitation Accepted",
    "source": "aws.partnercentral-selling",
    "account": "123456789012",
    "time": "2023-04-16T15:23:45Z",
    "region": "us-east-1",
    "resources": ["arn:aws:partnercentral:us-east-1::catalog/AWS/engagement-invitation/
engi-v7p8z56whnauo"],
    "detail": {
        "catalog": "AWS",
        "engagementInvitation": {
            "arn": "arn:aws:partnercentral:us-east-1::catalog/AWS/engagement-
invitation/engi-v7p8z56whnauo",
            "id": "engi-v7p8z56whnauo",
            "engagementId": "eng-12356whnauo",
            "participantType": "Enum", //Sender/Receiver
            "senderAccountId": "string",
            "receiverAccountId": "string",
            "payloadType": "LeadInvitation" //OpportunityInvitation|LeadInvitation
        }
    }
}
```

Engagement Invitation Rejected

Partners use this event to:

- Notify their users that the EngagementInvitation has been rejected.
- Update their internal records to reflect the rejected invitation.
- Trigger any necessary workflows, such as notifying the sales team about the rejection.

The following example shows a typical Engagement Invitation Rejected event.

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Engagement Invitation Rejected",
    "source": "aws.partnercentral-selling",
    "account": "123456789012",
    "time": "2023-04-17T09:48:27Z",
    "region": "us-east-1",
```

```
    "resources": ["arn:aws:partnercentral:us-east-1::catalog/AWS/engagement-invitation/
engi-v7p8z56whnauo"],
    "detail": {
        "catalog": "AWS",
        "engagementInvitation": {
            "arn": "arn:aws:partnercentral:us-east-1::catalog/AWS/engagement-
invitation/engi-v7p8z56whnauo",
            "id": "engi-v7p8z56whnauo",
            "engagementId": "eng-12356whnauo",
            "participantType": "Enum", //Sender/Receiver
            "senderAccountId": "string",
            "receiverAccountId": "string",
            "payloadType": "LeadInvitation" //OpportunityInvitation|LeadInvitation
        }
    }
}
```

Engagement Invitation Expired

Partners use this event to:

- Notify their users that the EngagementInvitation has expired and can no longer be acted upon.
- Update their internal records to reflect the expired invitation.
- Trigger any necessary workflows, such as notifying the sales team about the expired invitation.

The following example shows a typical Engagement Invitation Expired event.

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Engagement Invitation Expired",
    "source": "aws.partnercentral-selling",
    "account": "123456789012",
    "time": "2023-04-18T18:20:15Z",
    "region": "us-east-1",
    "resources": [
      "arn:aws:partnercentral:us-east-1::catalog/AWS/engagement-invitation/engi-
v7p8z56whnauo"
    ],
    "detail": {
      "catalog" : "AWS",
      "engagementInvitation": {
```

```
        "arn": "arn:aws:partnercentral:us-east-1::catalog/AWS/engagement-invitation/
engi-v7p8z56whnauo",
        "id": "engi-v7p8z56whnauo",
        "engagementId": "eng-12356whnauo",
        "participantType": "Enum", //Sender/Receiver
        "senderAccountId": "string",
        "receiverAccountId": "string",
        "payloadType": "LeadInvitation" //OpportunityInvitation|LeadInvitation
      }
    }
}
```

Engagement Member Added

Partners use this event to:

- Notify their users about the changes to the Engagement membership.
- Update their internal records to reflect the new Engagement member.
- Trigger any necessary workflows, such as notifying team members about the changes.

The following example shows a typical Engagement Member Added event.

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Engagement Member Added",
    "source": "aws.partnercentral-selling",
    "account": "123456789012",
    "time": "2023-04-16T15:23:45Z",
    "region": "us-east-1",
    "resources": [
      "arn:aws:partnercentral:us-east-1::catalog/AWS/engagement/eng-v7p8z56whnauo"
    ],
    "detail": {
      "catalog" : "AWS",
      "engagement": {
        "id": "eng-v7p8z56whnauo"
      },
      "engagementMember": {
        "accountId": "string",
        "companyName": "string"
      }
```

```
    }
}
```

Engagement Resource Snapshot Created

Partners use this event to track changes to resources associated with an engagement and trigger automated workflows, such as updating CRM systems or notifying sales teams. The snapshot template determines the type of update:

`OpportunitySummaryView`

- Notify all members of an engagement that the engagement has been updated.

`AwsOpportunitySummaryFullView`

- Track updates made specifically by AWS to opportunities within an engagement.
- This notification and snapshot are only visible to the opportunity owner.
- Includes deal sizing updates, which are not available through the `OpportunitySummaryView` template.

The following example shows a typical Engagement Resource Snapshot Created event.

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Engagement Resource Snapshot Created",
    "source": "aws.partnercentral-selling",
    "account": "123456789012",
    "time": "2023-04-18T18:20:15Z",
    "region": "us-east-1",
    "resources": [
      "arn:aws:partnercentral:us-east-1::catalog/AWS/engagement/eng-v7p8z56whnauo"
    ],
    "detail": {
      "catalog" : "AWS",
      "resourceSnapshot": {
        "arn": "arn:aws:partnercentral-selling:us-east-1::catalog/AWS/engagement/eng-
v7p8z56whnauo/resource/Opportunity/o-12312/template/temp-name/snapshot/snapshot-19232",
        "engagementId": "eng-v7p8z56whnauo",
        "resourceType": "Opportunity",
        "resourceId" : "0123211231",
        "createdBy": "123123123123",
```

```
        "targetMemberAccounts": ["123123123123", "aws"],
        "resourceSnapshotTemplateName": "temp-name"
      }
    }
}
```

Engagement Created

Partners use this event to:

- Notify their users that a new Engagement has been created.
- Trigger automated workflows such as updating CRM systems or notifying sales teams about new engagement creations.

The following example shows a typical Engagement Created event.

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Engagement Created",
    "source": "aws.partnercentral-selling",
    "account": "123456789012",
    "time": "2023-04-18T18:20:15Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:partnercentral:us-east-1::catalog/AWS/engagement/eng-567a1j5hxp35lo"
    ],
    "detail": {
        "catalog": "AWS",
        "engagement": {
            "engagementArn": "arn:aws:partnercentral:us-east-1::catalog/AWS/engagement/
eng-567a1j5hxp35lo",
            "engagementId": "eng-567a1j5hxp35lo",
            "CreatedAt": "2023-04-18T18:20:15Z",
            "CreatedBy": "aws",
            "ContextTypes": ["Lead"]

        }
    }
}
```

Engagement Updated

Partners use this event to:

- Notify their users that an existing Engagement has been updated.
- Trigger automated workflows such as updating CRM systems or notifying sales teams.

The following example shows a typical Engagement Updated event.

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Engagement Updated",
    "source": "aws.partnercentral-selling",
    "account": "123456789012",
    "time": "2023-04-18T18:20:15Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:partnercentral:us-east-1::catalog/AWS/engagement/eng-567a1j5hxp35lo"
    ],
    "detail": {
        "catalog": "AWS",
        "engagement": {
            "engagementArn": "arn:aws:partnercentral:us-east-1::catalog/AWS/engagement/
eng-567a1j5hxp35lo",
            "engagementId": "eng-567a1j5hxp35lo",
            "LastModifiedAt": "2023-04-18T18:20:15Z",
            "LastModifiedBy": "aws",
            "ContextTypes": ["Lead", "CustomerProject"]
        }
    }
}
```

Notifications for the AWS Partner Central Benefits API

The Benefits Service provides EventBridge events for both BenefitApplications and BenefitAllocations. These events enable customers to build comprehensive event-driven architectures that respond to changes throughout the entire resources lifecycles - from initial Application through Allocation and final Fulfillment.

Topics

- Complete the prerequisite to monitor events
- Configure Amazon EventBridge to monitor events
- Learn more about benefits API events

Complete the prerequisite to monitor events

Users require the appropriate IAM permissions to access and manage events published by the benefits API. For more information about the available actions, resources, and condition keys for EventBridge, see Using IAM policy conditions in Amazon EventBridge in the *Amazon EventBridge User Guide*. One of the condition keys is `events:detail-type`, which can be used to scope permissions to specific event types.

The following example policy demonstrates how to customize and scope the permissions for the proposed events. The `AllowPutRuleForPartnercentralBenefitsEvents` statement allows the creation of rules, but only for events from the `aws.partnercentral-benefits` source.

For detailed IAM policy examples, refer to the AWS documentation on EventBridge permissions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPutRuleForPartnerCentralBenefitsEvents",
      "Effect": "Allow",
      "Action": "events:PutRule",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "events:source": "aws.partnercentral-benefits"
        }
      }
    }
  ]
}
```

Configure Amazon EventBridge to monitor events

To monitor benefits API events, you create an EventBridge rule that matches the events that you want to capture. You can use the AWS Management Console or the AWS SDKs to create and

manage rules. The following sections explain how to create rules using both methods. Regardless of the method you use, you must create the rule in the US East (N. Virginia) `us-east-1` Region.

AWS Management Console setup

To set up an EventBridge rule using the AWS Management Console, follow the steps in Creating rules that react to events in Amazon EventBridge in the *Amazon EventBridge User Guide*. When creating the rule, you must set the event bus to **default**, and create the rule in the US East (N. Virginia) `us-east-1` Region.

Following is an example of an event rule:

```
{
    "source": ["aws.partnercentral-benefits"],
    "detail": {
        "catalog": ["AWS"]
    }
}
```

AWS SDK setup

You can use the AWS SDKs to create and manage EventBridge rules programmatically. For more information, see PutRule in the *Amazon EventBridge API Reference*.

The following example uses the AWS SDK for Python (Boto3):

```
import boto3

client = boto3.client('events', region_name='us-east-1')

response = client.put_rule(
    Name='MyBenefitApplicationCreatedRule',
    EventPattern=
    '{
        "source": ["aws.partnercentral-benefits"],
        "detail-type": ["Benefit Application Created"],
        "detail": {"catalog": ["AWS"]}
    }',
    State='ENABLED'
)
print('Rule ARN:', response['RuleArn'])
```

Learn more about benefits API events

The following sections describe the benefits API event types, scenarios that trigger them, and event examples.

Event types

Following are the event types and their triggers.

Benefit Applications

- Benefit Application Created: When a Benefit Application is created
- Benefit Application In Review: When a Benefit Application is submitted or when the Application transitions from Action Required back to In Review, this event is triggered.
- Benefit Application Action Required: This event occurs when an internal reviewer updates the application status to indicate that the partner needs to provide additional information, clarification, or modifications before the review can continue.
- Benefit Application Rejected: This event occurs when an internal reviewer updates the application status to indicate that the application does not meet the benefit criteria or requirements and has been declined
- Benefit Application Approved: This event occurs when an internal reviewer updates the application status to indicate that the application has been approved for benefit allocation.
- Benefit Application Stage Changed: This event occurs when an internal reviewer updates the stage of the application during review processes such as Business Review, AWS Review, Financial Review, etc.

Benefit Allocations

- Benefit Allocation Activated: When a Benefit Allocation is created or updated back to Activated by an internal service.
- Benefit Allocation Inactivated: This event occurs when an internal reviewer updates the allocation status or when a Benefit Allocation expires automatically.
- Benefit Allocation Fulfilled: This event occurs when an internal reviewer updates the allocation status to indicate that the allocation has been completely utilized.

Example events

The following sections provide examples of the events listed earlier in the previous section.

Topics

Benefit Application Created

Triggering Context

The Benefit Application Created event is published to Amazon EventBridge when a new Benefit Application is successfully created through the Benefits Service. This event occurs when the CreateBenefitApplication API call completes successfully, indicating that a partner has initiated a request for a specific benefit.

This event represents the beginning of the benefit application lifecycle and provides customers with immediate notification when new applications are submitted to their account.

Recipients

The owning account will receive the event for the Benefit Allocation.

Example


```
{
  "id": "7gh88765-k0jh-d08g-i292-2ff1796m030n",
  "detail-type": "Benefit Application Created",
  "source": "aws.partnercentral-benefits",
  "account": "123456789012",
```

```
  "time": "2025-02-10T15:45:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:partnercentral:us-east-1:123456789012:benefit-application/
benappl-12345abcdef123"
  ],
  "detail": {
    "catalog": "AWS",
    "benefitApplication": {
      "id": "benappl-12345abcdef123",
      "fulfillmentTypes": ["CREDITS"],
      "status": "PENDING_SUBMISSION",
      "revision": "hh7e8400-e29b-41d4-a716-446655440012"
    },
    "benefit": {
      "id": "ben-xyz789uvw012",
      "name": "AWS Partner Credits",
      "program": ["AWS Partner Network"]
    }
  }
}
```


Benefit Application In Review

Triggering Context

The Benefit Application In Review event is published to Amazon EventBridge when a Benefit Application transitions to the IN_REVIEW status through the Benefits Service. This event occurs when:

- A partner submits their application via the SubmitBenefitApplication API call
- An application transitions from ACTION_REQUIRED back to IN_REVIEW after the partner addresses feedback
- An application is recalled and then resubmitted

This event indicates that the application has entered the formal review process and is being evaluated by AWS internal teams.

Recipients

The owning account will receive the event for the Benefit Allocation.

Example


```
{
  "id": "8hi99876-l1ki-e19h-j303-3gg2807n141o",
  "detail-type": "Benefit Application In Review",
  "source": "aws.partnercentral-benefits",
  "account": "123456789012",
  "time": "2025-02-10T16:00:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:partnercentral:us-east-1:123456789012:benefit-application/
benappl-12345abcdef123"
  ],
  "detail": {
    "catalog": "AWS",
    "benefitApplication": {
      "id": "benappl-12345abcdef123",
      "fulfillmentTypes": ["CREDITS"],
      "status": "IN_REVIEW",
      "stage": "BUSINESS_REVIEW",
      "revision": "ii8f9511-f30c-52e5-b827-557766551123"
    },
    "benefit": {
      "id": "ben-xyz789uvw012",
      "name": "AWS Partner Credits",
      "program": ["AWS Partner Network"]
    }
  }
}
```


Benefit Application Action Required

Triggering Context

The Benefit Application Action Required event is published to Amazon EventBridge when a Benefit Application transitions to the ACTION_REQUIRED status through the Benefits Service. This event occurs when an internal reviewer updates the application status via the UpdateBenefitApplicationInternal API to indicate that the partner needs to provide additional information, clarification, or modifications before the review can continue.

This event represents a critical point in the application lifecycle where partner intervention is needed to move the application forward in the review process.

Recipients

The owning account will receive the event for the Benefit Allocation

Example


```
{
  "id": "9ij00987-m2lj-f20i-k414-4hh3918o252p",
  "detail-type": "Benefit Application Action Required",
  "source": "aws.partnercentral-benefits",
  "account": "123456789012",
  "time": "2025-02-12T10:30:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:partnercentral:us-east-1:123456789012:benefit-application/
benappl-12345abcdef123"
  ],
  "detail": {
    "catalog": "AWS",
    "benefitApplication": {
      "id": "benappl-12345abcdef123",
      "fulfillmentTypes": ["CREDITS"],
      "status": "ACTION_REQUIRED",
      "stage": "BUSINESS_REVIEW",
      "revision": "jj9g0622-g41d-63f6-c938-668877662234"
    },
    "benefit": {
      "id": "ben-xyz789uvw012",
      "name": "AWS Partner Credits",
      "program": ["AWS Partner Network"]
    }
  }
}
```


Benefit Application Rejected

Triggering Context

The Benefit Application Rejected event is published to Amazon EventBridge when a Benefit Application transitions to the REJECTED status through the Benefits Service. This event occurs when an internal reviewer updates the application status via the UpdateBenefitApplicationInternal API to indicate that the application does not meet the benefit criteria or requirements and has been declined.

This event represents a terminal state in the application lifecycle where the application has been formally declined and will not proceed to benefit allocation.

Recipients

The owning account will receive the event for the Benefit Allocation.

Example


```
{
  "id": "0kl11098-n3mk-g31j-l525-5ii4029p363q",
  "detail-type": "Benefit Application Rejected",
  "source": "aws.partnercentral-benefits",
  "account": "123456789012",
  "time": "2025-02-15T14:20:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:partnercentral:us-east-1:123456789012:benefit-application/
benappl-12345abcdef123"
  ],
  "detail": {
    "catalog": "AWS",
    "benefitApplication": {
      "id": "benappl-12345abcdef123",
      "fulfillmentTypes": ["CREDITS"],
      "status": "REJECTED",
      "revision": "kk0h1733-h52e-74g7-d049-779988773345"
    },
    "benefit": {
      "id": "ben-xyz789uvw012",
      "name": "AWS Partner Credits",
      "program": ["AWS Partner Network"]
    }
  }
}
```


Benefit Application Approved

Triggering Context

The Benefit Application Approved event is published to Amazon EventBridge when a Benefit Application transitions to the APPROVED status through the Benefits Service. This event occurs

when an internal reviewer updates the application status via the UpdateBenefitApplicationInternal API to indicate that the application meets all benefit criteria and requirements and has been approved for benefit allocation.

This event represents a successful completion of the application review process and typically triggers the creation of a corresponding Benefit Allocation.

Recipients

The owning account will receive the event for the Benefit Allocation.

Example

```
{
  "id": "1lm22109-o4nl-h42k-m636-6jj5130q474r",
  "detail-type": "Benefit Application Approved",
  "source": "aws.partnercentral-benefits",
  "account": "123456789012",
  "time": "2025-02-14T16:45:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:partnercentral:us-east-1:123456789012:benefit-application/
benappl-12345abcdef123"
  ],
  "detail": {
    "catalog": "AWS",
    "benefitApplication": {
      "id": "benappl-12345abcdef123",
      "fulfillmentTypes": ["CREDITS"],
      "status": "APPROVED",
      "revision": "ll1i2844-i63f-85h8-e150-880099884456"
    },
    "benefit": {
      "id": "ben-xyz789uvw012",
      "name": "AWS Partner Credits",
      "program": ["AWS Partner Network"]
    }
  }
}
```

Benefit Application Stage Changed

Triggering Context

The Benefit Application Stage Changed event is published to Amazon EventBridge when a Benefit Application's review stage is updated through the Benefits Service. This event occurs when an internal reviewer updates the Stage of the application to one of the possible Enum values (FINANCE_REVIEW, BUSINESS_REVIEW, TECH_REVIEW and AWS_REVIEW).

Unlike status changes, stage changes are read-only informational updates that provide visibility into the internal review workflow progression without requiring partner action.

Recipients

The owning account will receive the event for the Benefit Allocation.

Example


```
{
  "id": "2mn33210-p5om-i53l-n747-7kk6241r585s",
  "detail-type": "Benefit Application Stage Changed",
  "source": "aws.partnercentral-benefits",
  "account": "123456789012",
  "time": "2025-02-13T11:15:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:partnercentral:us-east-1:123456789012:benefit-application/
benappl-12345abcdef123"
  ],
  "detail": {
    "catalog": "AWS",
    "benefitApplication": {
      "id": "benappl-12345abcdef123",
      "fulfillmentTypes": ["CREDITS"],
      "status": "IN_REVIEW",
      "stage": "FINANCIAL_REVIEW",
      "revision": "mm2j3955-j74g-96i9-f261-991100995567"
    },
    "benefit": {
      "id": "ben-xyz789uvw012",
      "name": "AWS Partner Credits",
      "program": ["AWS Partner Network"]
    }
```

```
  }
}
```

Benefit Allocation Activated

Triggering Context

The Benefit Allocation Activated event is published to Amazon EventBridge when a Benefit Allocation transitions to the ACTIVE status through the Benefits Service. This event occurs when an internal reviewer updates the allocation status via the UpdateBenefitAllocationInternal API to reactivate a previously inactive allocation.

This event typically occurs when an allocation that was temporarily inactivated (due to expiration, policy changes, or administrative reasons) is restored to active status, making the benefits available for partner utilization again.

Recipients

The AWS Account belonging to the Partner will receive the message. In other words, the AWS Account which owns the resource.

Sample Event

```
{
  "id": "3no44321-q6pn-j64m-o858-8ll7352s696t",
  "detail-type": "Benefit Allocation Activated",
  "source": "aws.partnercentral-benefits",
  "account": "123456789012",
  "time": "2025-03-01T08:15:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:partnercentral:us-east-1:123456789012:benefit-allocation/benalloc-
abc123def456"
  ],
  "detail": {
    "catalog": "AWS",
    "benefitAllocation": {
      "id": "benalloc-abc123def456",
      "fulfillmentType": "CREDITS",
      "status": "ACTIVE"
    },
    "benefitApplication": {
```

```
      "id": "benappl-12345abcdef123"
    },
    "benefit": {
      "id": "ben-xyz789uvw012",
      "name": "AWS Partner Credits",
      "program": ["AWS Partner Network"]
    }
  }
}
```

Benefit Allocation Inactivated

Triggering Context

The Benefit Allocation Inactivated event is published to Amazon EventBridge when a Benefit Allocation transitions to the INACTIVE status through the Benefits Service. This event occurs when an internal reviewer updates the allocation status via the UpdateBenefitAllocationInternal API or when a Benefit Allocation expires automatically.

This event typically occurs when an allocation is temporarily suspended (due to expiration, policy changes, compliance issues, or administrative reasons), making the benefits unavailable for partner utilization until reactivated.

Recipients

The AWS Account belonging to the Partner will receive the message. In other words, the AWS Account which owns the resource.

Sample Event

```
{
  "id": "4op55432-r7qo-k75n-p969-9mm8463t707u",
  "detail-type": "Benefit Allocation Inactivated",
  "source": "aws.partnercentral-benefits",
  "account": "123456789012",
  "time": "2025-06-30T23:59:59Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:partnercentral:us-east-1:123456789012:benefit-allocation/benalloc-
abc123def456"
  ],
  "detail": {
```

```
    "catalog": "AWS",
    "benefitAllocation": {
      "id": "benalloc-abc123def456",
      "fulfillmentType": "CREDITS",
      "status": "INACTIVE"
    },
    "benefitApplication": {
      "id": "benappl-12345abcdef123"
    },
    "benefit": {
      "id": "ben-xyz789uvw012",
      "name": "AWS Partner Credits",
      "program": ["AWS Partner Network"]
    }
  }
}
```

Benefit Allocation Fulfilled

Triggering Context

The Benefit Allocation Fulfilled event is published to Amazon EventBridge when a Benefit Allocation transitions to the FULFILLED status through the Benefits Service. This event occurs when an internal reviewer updates the allocation status via the UpdateBenefitAllocationInternal API to indicate that the allocation has been completely utilized or that the benefit terms have been satisfied.

This event represents the completion of the benefit lifecycle, indicating that the partner has successfully utilized the allocated benefit or that the allocation has reached its natural conclusion.

Recipients

The AWS Account belonging to the Partner will receive the message. In other words, the AWS Account which owns the resource.

Sample Event

```
{
  "id": "5pq66543-s8rp-l86o-q070-0nn9574u818v",
  "detail-type": "Benefit Allocation Fulfilled",
  "source": "aws.partnercentral-benefits",
  "account": "123456789012",
```

```
  "time": "2025-12-15T17:30:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:partnercentral:us-east-1:123456789012:benefit-allocation/benalloc-
abc123def456"
  ],
  "detail": {
    "catalog": "AWS",
    "benefitAllocation": {
      "id": "benalloc-abc123def456",
      "fulfillmentType": "CREDITS",
      "status": "FULFILLED"
    },
    "benefitApplication": {
      "id": "benappl-12345abcdef123"
    },
    "benefit": {
      "id": "ben-xyz789uvw012",
      "name": "AWS Partner Credits",
      "program": ["AWS Partner Network"]
    }
  }
}
```

Notifications for the AWS Partner Central Channel API

Events for the channel API provide real-time notifications about changes in the status or the details of channel-related activities. These events help keep your systems in sync with AWS Partner Central, and help ensure timely responses and updates.

Prerequisites

You need the appropriate IAM permissions to access and manage events from the AWS Partner Central Channel API. For information about the available actions, resources, and condition keys for EventBridge, see Using IAM policy conditions in Amazon EventBridge in the *Amazon EventBridge User Guide*. One of the condition keys is `events:detail-type`, which you can use to scope permissions to specific event types.

The following example policy demonstrates how to customize and scope permissions for the events. The `AllowPutRuleForPartnercentralChannelEvents` statement allows the creation of rules, but only for events from the `aws.partnercentral-channel` source.

For detailed IAM policy examples, refer to the AWS documentation on EventBridge permissions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPutRuleForPartnerCentralChannelEvents",
      "Effect": "Allow",
      "Action": "events:PutRule",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "events:source": "aws.partnercentral-channel"
        }
      }
    }
  ]
}
```

Configure Amazon EventBridge to monitor events

To monitor channel API events, you create an EventBridge rule that matches the events that you want to capture. You can use the AWS Management Console or the AWS SDKs to create and manage rules. The following sections explain how to create rules using both methods. Regardless of the method you use, you must create the rule in the US East (N. Virginia) `us-east-1` Region.

AWS Management Console setup

To set up an EventBridge rule using the AWS Management Console, follow the steps in Creating rules that react to events in Amazon EventBridge in the *Amazon EventBridge User Guide*. When creating the rule, you must set the event bus to **default**, and create the rule in the US East (N. Virginia) `us-east-1` Region.

Following is an example of an event rule:

```
{
    "source": ["aws.partnercentral-channel"],
    "detail": {
        "catalog": ["AWS"]
    }
}
```

AWS SDK setup

You can use the AWS SDKs to create and manage EventBridge rules programmatically. For more information, see PutRule in the *Amazon EventBridge API Reference*.

The following example uses the AWS SDK for Python (Boto3):

```
import boto3

client = boto3.client('events', region_name='us-east-1')

response = client.put_rule(
    Name='ChannelHandshakeCreatedRule',
    EventPattern=
    '{
        "source": ["aws.partnercentral-channel"],
        "detail-type": ["Channel Handshake Created"],
        "detail": {"catalog": ["AWS"]}
    }',
    State='ENABLED'
)
print('Rule ARN:', response['RuleArn'])
```

Learn more about channel API events

The following sections describe the channel API event types, scenarios that trigger them, and event examples.

Event types

Following are the event types and their triggers.

- Channel Handshake Created: Triggered when a new channel handshake is created.
- Channel Handshake Accepted: Triggered when a new channel handshake is accepted.
- Channel Handshake Rejected: Triggered when a new channel handshake is rejected.

Channel Handshake Created

The following examples show typical Channel Handshake Created events for different handshake types.

Start Service Period Handshake:

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Channel Handshake Created",
    "source": "aws.partnercentral-channel",
    "account": "789789789789",
    "time": "2025-02-01T00:00:00Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:partnercentral:us-east-1:123123123123:catalog/AWS/channel-handshake/
ch-abc123def456g"
    ],
    "detail": {
        "requestId": "12345678-1234-1234-1234-123456789abc",
        "catalog": "AWS",
        "associatedResourceIdentifier": "rs-jkl012mno345p",
        "handshakeType": "START_SERVICE_PERIOD",
        "id": "ch-abc123def456g",
        "receiverAccountId": "789789789789",
        "ownerAccountId": "123123123123",
        "senderAccountId": "456456456456",
        "senderDisplayName": "TestDisplayName",
        "handshakeDetail": {
            "startServicePeriodHandshakeDetail": {
                "servicePeriodType": "MINIMUM_NOTICE_PERIOD",
                "minimumNoticeDays": "14",
                "endDate": null,
                "startDate": "2025-02-02T00:00:00Z",
                "note": "Test note example"
            }
        }
    }
}
```

Revoke Service Period Handshake:

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Channel Handshake Created",
    "source": "aws.partnercentral-channel",
    "account": "789789789789",
    "time": "2025-02-01T00:00:00Z",
```

```
    "region": "us-east-1",
    "resources": [
        "arn:aws:partnercentral:us-east-1:123123123123:catalog/AWS/channel-handshake/
ch-def456ghi789j"
    ],
    "detail": {
        "requestId": "12345678-1234-1234-1234-123456789abc",
        "catalog": "AWS",
        "associatedResourceIdentifier": "rs-jkl012mno345p",
        "handshakeType": "REVOKE_SERVICE_PERIOD",
        "id": "ch-def456ghi789j",
        "ownerAccountId": "123123123123",
        "receiverAccountId": "789789789789",
        "senderAccountId": "456456456456",
        "senderDisplayName": "TestDisplayName",
        "handshakeDetail": {
            "revokeServicePeriodHandshakeDetail": {
                "servicePeriodType": "MINIMUM_NOTICE_PERIOD",
                "minimumNoticeDays": "14",
                "endDate": null,
                "startDate": "2025-02-02T00:00:00Z",
                "note": "Test note example"
            }
        }
    }
}
```

Program Management Account Handshake:

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Channel Handshake Created",
    "source": "aws.partnercentral-channel",
    "account": "456456456456",
    "time": "2025-02-01T00:00:00Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:partnercentral:us-east-1:123123123123:catalog/AWS/channel-handshake/
ch-ghi789jkl012m"
    ],
    "detail": {
        "requestId": "12345678-1234-1234-1234-123456789abc",
```

```
        "catalog": "AWS",
        "associatedResourceIdentifier": "pma-abc123def456g",
        "handshakeType": "PROGRAM_MANAGEMENT_ACCOUNT",
        "id": "ch-ghi789jkl012m",
        "ownerAccountId": "123123123123",
        "receiverAccountId": "456456456456",
        "senderAccountId": "123123123123",
        "senderDisplayName": "TestDisplayName",
        "handshakeDetail": {
            "programManagementAccountHandshakeDetail": {
                "program": "SOLUTION_PROVIDER"
            }
        }
    }
}
```

Channel Handshake Accepted

The following examples show typical Channel Handshake Accepted events for different handshake types.

Start Service Period Handshake:

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Channel Handshake Accepted",
    "source": "aws.partnercentral-channel",
    "account": "123123123123",
    "time": "2025-02-01T00:00:00Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:partnercentral:us-east-1:123123123123:catalog/AWS/channel-handshake/
ch-abc123def456g"
    ],
    "detail": {
        "requestId": "12345678-1234-1234-1234-123456789abc",
        "catalog": "AWS",
        "associatedResourceIdentifier": "rs-jkl012mno345p",
        "handshakeType": "START_SERVICE_PERIOD",
        "id": "ch-abc123def456g",
        "ownerAccountId": "123123123123",
        "receiverAccountId": "789789789789",
```

```
        "senderAccountId": "456456456456",
        "senderDisplayName": "TestDisplayName",
        "handshakeDetail": {
            "startServicePeriodHandshakeDetail": {
                "servicePeriodType": "MINIMUM_NOTICE_PERIOD",
                "minimumNoticeDays": "14",
                "endDate": null,
                "startDate": "2025-02-02T00:00:00Z",
                "note": "Test note example"
            }
        }
    }
}
```

Revoke Service Period Handshake:

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Channel Handshake Accepted",
    "source": "aws.partnercentral-channel",
    "account": "123123123123",
    "time": "2025-02-01T00:00:00Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:partnercentral:us-east-1:123123123123:catalog/AWS/channel-handshake/
ch-def456ghi789j"
    ],
    "detail": {
        "requestId": "12345678-1234-1234-1234-123456789abc",
        "catalog": "AWS",
        "associatedResourceIdentifier": "rs-jkl012mno345p",
        "handshakeType": "REVOKE_SERVICE_PERIOD",
        "id": "ch-def456ghi789j",
        "ownerAccountId": "123123123123",
        "receiverAccountId": "789789789789",
        "senderAccountId": "456456456456",
        "senderDisplayName": "TestDisplayName",
        "handshakeDetail": {
            "revokeServicePeriodHandshakeDetail": {
                "servicePeriodType": "MINIMUM_NOTICE_PERIOD",
                "minimumNoticeDays": "14",
                "endDate": null,
```

```
                "startDate": "2025-02-02T00:00:00Z",
                "note": "Test note example"
            }
        }
    }
}
```

Program Management Account Handshake:

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Channel Handshake Accepted",
    "source": "aws.partnercentral-channel",
    "account": "123123123123",
    "time": "2025-02-01T00:00:00Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:partnercentral:us-east-1:123123123123:catalog/AWS/channel-handshake/
ch-ghi789jkl012m"
    ],
    "detail": {
        "requestId": "12345678-1234-1234-1234-123456789abc",
        "catalog": "AWS",
        "associatedResourceIdentifier": "pma-abc123def456g",
        "handshakeType": "PROGRAM_MANAGEMENT_ACCOUNT",
        "id": "ch-ghi789jkl012m",
        "ownerAccountId": "123123123123",
        "receiverAccountId": "456456456456",
        "senderAccountId": "123123123123",
        "senderDisplayName": "TestDisplayName",
        "handshakeDetail": {
            "programManagementAccountHandshakeDetail": {
                "program": "SOLUTION_PROVIDER"
            }
        }
    }
}
```

Channel Handshake Rejected

The following examples show typical Channel Handshake Rejected events for different handshake types.

Start Service Period Handshake:


```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Channel Handshake Rejected",
    "source": "aws.partnercentral-channel",
    "account": "123123123123",
    "time": "2025-02-01T00:00:00Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:partnercentral:us-east-1:123123123123:catalog/AWS/channel-handshake/
ch-abc123def456g"
    ],
    "detail": {
        "requestId": "12345678-1234-1234-1234-123456789abc",
        "catalog": "AWS",
        "associatedResourceIdentifier": "rs-jkl012mno345p",
        "handshakeType": "START_SERVICE_PERIOD",
        "id": "ch-abc123def456g",
        "ownerAccountId": "123123123123",
        "receiverAccountId": "789789789789",
        "senderAccountId": "456456456456",
        "senderDisplayName": "TestDisplayName",
        "handshakeDetail": {
            "startServicePeriodHandshakeDetail": {
                "servicePeriodType": "MINIMUM_NOTICE_PERIOD",
                "minimumNoticeDays": "14",
                "endDate": null,
                "startDate": "2025-02-02T00:00:00Z",
                "note": "Test note example"
            }
        }
    }
}
```


Revoke Service Period Handshake:


```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Channel Handshake Rejected",
    "source": "aws.partnercentral-channel",
```

```
    "account": "123123123123",
    "time": "2025-02-01T00:00:00Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:partnercentral:us-east-1:123123123123:catalog/AWS/channel-handshake/
ch-def456ghi789j"
    ],
    "detail": {
        "requestId": "12345678-1234-1234-1234-123456789abc",
        "catalog": "AWS",
        "associatedResourceIdentifier": "rs-jkl012mno345p",
        "handshakeType": "REVOKE_SERVICE_PERIOD",
        "id": "ch-def456ghi789j",
        "ownerAccountId": "123123123123",
        "receiverAccountId": "789789789789",
        "senderAccountId": "456456456456",
        "senderDisplayName": "TestDisplayName",
        "handshakeDetail": {
            "revokeServicePeriodHandshakeDetail": {
                "servicePeriodType": "MINIMUM_NOTICE_PERIOD",
                "minimumNoticeDays": "14",
                "endDate": null,
                "startDate": "2025-02-02T00:00:00Z",
                "note": "Test note example"
            }
        }
    }
}
```

Program Management Account Handshake:

```
{
    "version": "0",
    "id": "01234567-0123-0123-0123-0123456789ab",
    "detail-type": "Channel Handshake Rejected",
    "source": "aws.partnercentral-channel",
    "account": "123123123123",
    "time": "2025-02-01T00:00:00Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:partnercentral:us-east-1:123123123123:catalog/AWS/channel-handshake/
ch-ghi789jkl012m"
    ],
```

```
    "detail": {
        "requestId": "12345678-1234-1234-1234-123456789abc",
        "catalog": "AWS",
        "associatedResourceIdentifier": "pma-abc123def456g",
        "handshakeType": "PROGRAM_MANAGEMENT_ACCOUNT",
        "id": "ch-ghi789jkl012m",
        "ownerAccountId": "123123123123",
        "receiverAccountId": "456456456456",
        "senderAccountId": "123123123123",
        "senderDisplayName": "TestDisplayName",
        "handshakeDetail": {
            "programManagementAccountHandshakeDetail": {
                "program": "SOLUTION_PROVIDER"
            }
        }
    }
}
```

Testing in a sandbox

Partners can use a sandbox to test and validate their AWS Partner Central API interactions in a secure and isolated environment, ensuring smooth operation before promoting their solution to the production environment. AWS offers a dynamic sandbox to AWS Partner Central API users that returns responses similar to the production environment. AWS does not provide a user interface to the sandbox environment. Therefore, partners need to rely on the programmatic responses to test their solutions.

Access to the sandbox environment

Partners gain access to the testing environment as soon as they link their AWS account to the Partner Central account. For more information, see Linking your AWS account to AWS Partner Central account. Each request includes a `catalog` parameter, which determines the data environment. When `catalog` is set to `AWS`, it references production data, and when it's set to `Sandbox`, it references sandbox data.

Important details about the sandbox environment

1. Data refresh: Once per year, AWS refreshes the data in the sandbox environment (typically at the beginning of the year). After this refresh, you may lose some of the data in your testing environment.

2. Testing scope: The sandbox environment is typically used for functional testing and not for testing scalability or performance.

Topics

Testing in a sandbox for the AWS Partner Central Account API

How to use the sandbox

To use the sandbox, complete the following steps:

1. Create an IAM role:

   Create an IAM role in the AWS account linked with your AWS Partner Central account.
2. Assign Policy:

   Assign the following policy to the IAM role. For more information, see Adding and removing IAM identity permissions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AccountManagement",
      "Effect": "Allow",
      "Action": [
        "partnercentral:AcceptConnectionInvitation",
        "partnercentral:AssociateAwsTrainingCertificationEmailDomain",
        "partnercentral:CancelConnection",
        "partnercentral:CancelConnectionInvitation",
        "partnercentral:CancelProfileUpdateTask",
        "partnercentral:CreateConnectionInvitation",
        "partnercentral:CreatePartner",
        "partnercentral:DisassociateAwsTrainingCertificationEmailDomain",
```

```
        "partnercentral:GetAllianceLeadContact",
        "partnercentral:GetConnection",
        "partnercentral:GetConnectionInvitation",
        "partnercentral:GetConnectionPreferences",
        "partnercentral:GetPartner",
        "partnercentral:GetProfileUpdateTask",
        "partnercentral:GetProfileVisibility",
  "partnercentral:GetVerification",
        "partnercentral:ListConnectionInvitations",
        "partnercentral:ListConnections",
        "partnercentral:ListPartners",
        "partnercentral:PutAllianceLeadContact",
        "partnercentral:PutProfileVisibility",
        "partnercentral:RejectConnectionInvitation",
        "partnercentral:SendEmailVerificationCode",
        "partnercentral:StartProfileUpdateTask",
  "partnercentral:StartVerification",
        "partnercentral:UpdateConnectionPreferences"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "partnercentral:Catalog": [
            "Sandbox"
          ]
        }
      }
    },
    {
      "Sid": "TaggingAccess",
      "Effect": "Allow",
      "Action": [
        "partnercentral:TagResource",
        "partnercentral:UntagResource",
        "partnercentral:ListTagsForResource"
      ],
      "Resource": [
        "arn:aws:partnercentral:*:*:catalog/Sandbox/partner/*"
      ],
      "Condition": {
        "StringEquals": {
          "partnercentral:Catalog": [
            "Sandbox"
          ]
```

```
        }
      }
    }
  ]
}
```

3. Use IAM role credentials:

   Use the credentials (secret key and access key) of this IAM role in your solution to perform the API actions.

Testing AWS actions

During the testing phase, it is often necessary to simulate AWS actions. This simulation enables partners to thoroughly test the complete end-to-end flow of their integration with AWS services. Each request includes a catalog parameter, which determines the data environment.

Simulating the creation of a Partner

To simulate the creation of a Partner, in the payload of the `CreatePartner` action, include `"catalog": "Sandbox"` in the payload.

For example:

```
{
  "ClientToken": "client-generated-uuid",
  "Catalog": "Sandbox",
  "LegalName": "Your Name",
  "PrimarySolutionType": "SOLUTION_TYPE",
  "AllianceLeadContact": {
    "FirstName": "Example First Name",
    "LastName": "Example Last Name",
    "BusinessTitle": "Example Title",
    "Email": "example@domain.com"
  },
  "EmailVerificationCode": "123456"
}
```

Note: In Sandbox email verification is disabled. You can use any six-digit EmailVerificationCode for testing. Also, don't use "Tags" in the request.

Additional testing notes

1. To test other actions, set `"catalog": "Sandbox"` in the payload.
2. To test partner account connections in sandbox, you must execute `StartProfileUpdateTask` action after creating partner in sandbox catalog.
3. To move to production, change the catalog value from `Sandbox` to `AWS`.

Testing events in the sandbox environment

Partners can consume events from the sandbox environment to help test the event-based implementations. Set up EventBridge in the same AWS account with rules to listen for sandbox events by specifying `catalog: Sandbox` in the event details. For more information, see Notifications for the AWS Partner Central Account API.

Example event rule:

```
{
  "source": ["aws.partnercentral-account"],
  "detail": {
    "catalog": ["Sandbox"]
  }
}
```

Event rules that specify `catalog` as `Sandbox` will only match events coming from the sandbox, generated by the actions you perform in the sandbox environment.

Testing in a sandbox for the AWS Partner Central Selling API

How to use the sandbox

1. Create an IAM role:

   Create an IAM role in the AWS account linked with your AWS Partner Central account.

2. Assign Policy:

   Assign the following policy to the IAM role. For more information, see Adding and removing IAM identity permissions.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "partnercentral:CreateOpportunity",
                "partnercentral:UpdateOpportunity",
                "partnercentral:ListOpportunities",
                "partnercentral:GetOpportunity",
                "partnercentral:GetAwsOpportunitySummary",
                "partnercentral:ListSolutions",
                "partnercentral:AssociateOpportunity",
                "partnercentral:DisassociateOpportunity",
                "partnercentral:AssignOpportunity",
                "partnercentral:SubmitOpportunity",
                "partnercentral:AcceptEngagementInvitation",
                "partnercentral:CreateEngagementInvitation",
                "partnercentral:RejectEngagementInvitation",
                "partnercentral:GetEngagementInvitation",
                "partnercentral:ListEngagementInvitations",
                "partnercentral:StartEngagementFromOpportunityTask",
                "partnercentral:StartEngagementByAcceptingInvitationTask",
                "partnercentral:CreateResourceSnapshotJob",
                "partnercentral:StartResourceSnapshotJob",
                "partnercentral:CreateEngagement"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": "Sandbox"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "aws-marketplace:ListEntities",
                "aws-marketplace:DescribeEntity"
            ],
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "aws-marketplace:SearchAgreements",
                "aws-marketplace:DescribeAgreement"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws-marketplace:PartyType": "Proposer"
                }
            }
        }
    ]
}
```

3. Use IAM role credentials:

   Use the credentials (secret key and access key) of this IAM role in your solution to perform the API actions.

Testing AWS actions

During the testing phase, it is often necessary to simulate AWS actions. This simulation enables partners to thoroughly test the complete end-to-end flow of their integration with AWS services.

Simulating the creation of an AWS originated opportunity

To simulate the creation of an AWS originated opportunity, in the payload of the `CreateOpportunity` action, include `"Catalog": "Sandbox"` and `"Origin": "AWS Referral"`.

For example:

```
{
    "Catalog": "Sandbox",
    "Origin": "AWS Referral",
    "OpportunityIdentifier": "O123456",
    "Title": "Test Opportunity",
    ...
}
```

Simulating updates to a partner-originated opportunity

To simulate updates or other actions on a partner-originated opportunity, use the `UpdateOpportunity` action with `"Catalog": "Sandbox"` and `Lifecycle.ReviewStatus: “Approved”` or `“Action Required”` in the payload.

If you are taking the payload from the `GetOpportunity` action to do the update, ensure that you change the ID to `Identifier`, and remove the following fields:

- `CreatedDate`
- `OpportunityTeam`
- `RelatedEntityIdentifiers`

For example:

```
{
    "Catalog": "Sandbox",
    "Identifier": "O123456",
    "Title": "Updated Test Opportunity",
    "Lifecycle": {
        "ReviewStatus": "Approved"
    }
    ...
}
```

Testing events in the sandbox environment

Partners can consume opportunity events from the sandbox environment to help test the event-based implementations. Set up EventBridge in the same AWS account with rules to listen for sandbox events by specifying `catalog: sandbox` in the event details. For more information, see Notifications for the AWS Partner Central Selling API.

Example event rule:

```
{
    "source": ["aws.partnercentral-selling"],
    "detail": {
        "catalog": ["Sandbox"]
    }
}
```

Event rules that specify `catalog` as `Sandbox` will only match events coming from the sandbox, generated by the actions you perform in the sandbox environment.

Additional testing notes

1. Testing `AssociateOpportunity` action:
   a. Use the default solution `"S-1234567"` for testing the `AssociateOpportunity` action.
   b. For testing Marketplace offers, use a real offer ID from your account.
2. To move to production, change the `catalog` value from `Sandbox` to `AWS`.

Getting help

If you encounter challenges integrating your CRM with AWS, or if you need to test a specific scenario not covered here, please reach out to support by raising a case through the following steps:

1. Sign in to the AWS Partner Central with your AWS Partner Network credentials.
2. On the Support Center for Partner Central, choose `Open New Case` to log a new case. Complete the fields as follows:
   a. Type of support case: `AWS Partner Central`.
   b. Question regarding: `Partner Central Tools or ACE leads and opportunities`.
   c. Get specific: Select the most appropriate CRM Integration case type.
   d. Subject: Include a brief description of the request.
   e. Description: Provide a detailed description of issues, questions, errors, and troubleshooting steps.
   f. Attachments: Include logs and screenshots, where applicable.

Testing in a sandbox for the AWS Partner Central Benefits API

How to use the sandbox

To use the sandbox, complete the following steps:

1. Create an IAM role:

   Create an IAM role in the AWS account linked with your AWS Partner Central account.

2. Assign Policy:

Assign the following policy to the IAM role. For more information, see Adding and removing IAM identity permissions.

```
{
  "Version": "2012-10-17",
    "Statement": [{
            "Sid": "BenefitsManagement",
            "Effect": "Allow",
            "Action": [
                "partnercentral:ListBenefits",
                "partnercentral:GetBenefit",
                "partnercentral:CreateBenefitApplication",
                "partnercentral:AmendBenefitApplication",
                "partnercentral:UpdateBenefitApplication",
                "partnercentral:SubmitBenefitApplication",
                "partnercentral:GetBenefitApplication",
                "partnercentral:CancelBenefitApplication",
                "partnercentral:RecallBenefitApplication",
                "partnercentral:ListBenefitApplications",
                "partnercentral:AssociateBenefitApplicationResource",
                "partnercentral:DisasssociateBenefitApplicationResource",
                "partnercentral:ListBenefitAllocations",
                "partnercentral:GetBenefitAllocation",
                "partnercentral:TagResource",
                "partnercentral:UntagResource",
                "partnercentral:ListTagsForResource"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "AWSPartnerOpportunityAccess",
            "Effect": "Allow",
            "Action": [
                "partnercentral:GetAwsOpportunitySummary",
```

```
                "partnercentral:GetOpportunity",
                "partnercentral:ListOpportunities"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "ListingAWSMarketplaceEntities",
            "Effect": "Allow",
            "Action": [
                "aws-marketplace:ListEntities"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AWSMarketplaceOffersAccess",
            "Effect": "Allow",
            "Action": [
                "aws-marketplace:DescribeEntity"
            ],

            "Resource": [
                "arn:aws:aws-marketplace:::AWSMarketplace/Offer/*"
            ]
        },
        {
            "Sid": "S3PutObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": ["arn:aws:s3:::aws-partner-central-marketplace-ephemeral-
writeonly-files/*"]
        },
        {
            "Sid": "TaggingAccess",
            "Effect": "Allow",
            "Action": [
```

```
                "partnercentral:TagResource",
                "partnercentral:UntagResource",
                "partnercentral:ListTagsForResource"
            ],
            "Resource": [
                "arn:aws:partnercentral:us-east-1:*:catalog/Sandbox/benefit-
application/*",
                "arn:aws:partnercentral:us-east-1:*:catalog/Sandbox/benefit-
allocation/*"
            ],
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "Sandbox"
                    ]
                }
            }
        }
    ]
}
```

3. Use IAM role credentials:

   Use the credentials (secret key and access key) of this IAM role in your solution to perform the API actions.

Testing AWS actions

During the testing phase, it is often necessary to simulate AWS actions. This simulation enables partners to thoroughly test the complete end-to-end flow of their integration with AWS services. Each request includes a catalog parameter, which determines the data environment.

Simulating the creation of a Benefit Application

To simulate the creation of a benefit application, in the payload of the `CreateBenefitApplication` action, include `"catalog": "Sandbox"` in the payload.

For example:

```
{
  "Catalog": "Sandbox",
  "ClientToken": "String",
```

```
  "BenefitIdentifier": "String",
  "FulfillmentTypes": ["String"],
  "BenefitApplicationDetails": JsonDocument,
  "Name": "String",
  "Description": "String",
  "Tags": [
    {
      "Key": "string",
      "Value": "string"
    }
  ],
  "PartnerContacts": [
    {
      "BusinessTitle": "string",
      "Email": "string",
      "FirstName": "string",
      "LastName": "string",
      "Phone": "string"
    }
  ],
  "FileDetails": [
    {
      "FileURI": "String",
      "BusinessUseCase": "String"
    }
  ],
  "AssociatedResources": [
    {
      "ResourceType": "BENEFIT_ALLOCATION | OPPORTUNITY",
      "ResourceArn": "ARN"
    }
  ]
}
```

Additional testing notes

1. To test other actions, set `"catalog": "Sandbox"` in the payload.
2. The Benefit IDs in Sandbox catalog and AWS catalog are different. Make `GetBenefit` API call with `"catalog": "Sandbox"` to get Benefit Identifier in the Sandbox.
3. The Benefit allocation in Sandbox catalog and AWS catalog are different.
4. To move to production, change the catalog value from `Sandbox` to `AWS`.

Testing events in the sandbox environment

Partners can consume events from the sandbox environment to help test the event-based implementations. Set up EventBridge in the same AWS account with rules to listen for sandbox events by specifying `catalog: Sandbox` in the event details. For more information, see Notifications for the AWS Partner Central Benefits API.

Example event rule:

```
{
  "source": ["aws.partnercentral-benefits"],
  "detail": {
    "catalog": ["Sandbox"]
  }
}
```

Event rules that specify `catalog` as `Sandbox` will only match events coming from the sandbox, generated by the actions you perform in the sandbox environment.

Testing in a sandbox for the AWS Partner Central Channel API

Accounts in sandbox are not eligible for channel benefits, and are not validated against all program requirements.

How to use the sandbox

To use the sandbox, complete the following steps:

1. Create an IAM role:

   Create an IAM role in the AWS account linked with your AWS Partner Central account.
2. Assign Policy:

   Assign the following policy to the IAM role. For more information, see Adding and removing IAM identity permissions.

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

```
      "Effect": "Allow",
      "Action": [
        "partnercentral:CreateProgramManagementAccount",
        "partnercentral:UpdateProgramManagementAccount",
        "partnercentral:DeleteProgramManagementAccount",
        "partnercentral:ListProgramManagementAccounts",
        "partnercentral:GetProgramManagementAccount",
        "partnercentral:CreateRelationship",
        "partnercentral:UpdateRelationship",
        "partnercentral:DeleteRelationship",
        "partnercentral:GetRelationship",
        "partnercentral:ListRelationships",
        "partnercentral:CreateChannelHandshake",
        "partnercentral:AcceptChannelHandshake",
        "partnercentral:RejectChannelHandshake",
        "partnercentral:CancelChannelHandshake",
        "partnercentral:ListChannelHandshakes"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "partnercentral:Catalog": [
            "Sandbox"
          ]
        }
      }
    },
    {
      "Sid": "TaggingAccess",
      "Effect": "Allow",
      "Action": [
        "partnercentral:TagResource",
        "partnercentral:UntagResource",
        "partnercentral:ListTagsForResource"
      ],
      "Resource": [
        "arn:aws:partnercentral:*:*:catalog/Sandbox/program-management-account/*",
        "arn:aws:partnercentral:*:*:catalog/Sandbox/channel-handshake/*"
      ],
      "Condition": {
        "StringEquals": {
          "partnercentral:Catalog": [
            "Sandbox"
          ]
```

```
        }
      }
    }
  ]
}
```

3. Use IAM role credentials:

   Use the credentials (secret key and access key) of this IAM role in your solution to perform the API actions.

Testing AWS actions

During the testing phase, it is often necessary to simulate AWS actions. This simulation enables partners to thoroughly test the complete end-to-end flow of their integration with AWS services. Each request includes a catalog parameter, which determines the data environment.

Simulating the creation of a Program Management Account

To simulate the creation of a Program Management Account, in the payload of the `CreateProgramManagementAccount` action, include `"catalog": "Sandbox"` in the payload.

For example:

```
{
  "catalog": "Sandbox",
  "accountId": "123456789012",
  "displayName": "Test PMA Name",
  "clientToken": "123abc456def789ghi",
  "program": "SOLUTION_PROVIDER",
  "tags": [
    {
      "key": "testkey",
      "value": "testvalue"
    }
  ]
}
```

Simulating updates to a Relationship

To simulate updates on a relationship, use the `UpdateRelationship` action with `"catalog": "Sandbox"` in the payload.

For example:

```
{
  "catalog": "Sandbox",
  "displayName": "Updated Test PMA",
  "identifier": "rs-abc123def456g",
  "programManagementAccountIdentifier": "pma-123abc456def7"
}
```

Additional testing notes

1. To test other actions, set `"catalog": "Sandbox"` in the payload.
2. To move to production, change the catalog value from `Sandbox` to `AWS`.

Testing events in the sandbox environment

Partners can consume events from the sandbox environment to help test the event-based implementations. Set up EventBridge in the same AWS account with rules to listen for sandbox events by specifying `catalog: Sandbox` in the event details. For more information, see Notifications for the AWS Partner Central Channel API.

Example event rule:

```
{
  "source": ["aws.partnercentral-channel"],
  "detail": {
    "catalog": ["Sandbox"]
  }
}
```

Event rules that specify `catalog` as `Sandbox` will only match events coming from the sandbox, generated by the actions you perform in the sandbox environment.

Getting help

If you encounter challenges in testing, or if you need to test a specific scenario not covered here, please reach out to support by raising a case through the following steps:

1. Sign in to the AWS Partner Central with your AWS Partner Network credentials.

2. On the Support Center for Partner Central, choose `Open New Case` to log a new case. Complete the fields as follows:
   a. Type of support case: `Channel Program`
   b. Question regarding: `General Program Inquiry`
   c. Get specific: Select the most appropriate Channel case type.
   d. Subject: Include a brief description of the request.
   e. Description: Provide a detailed description of issues, questions, errors, and troubleshooting steps.
   f. Attachments: Include logs and screenshots, where applicable.

Testing in a sandbox for the AWS Partner Central Revenue Measurement API

How to use the sandbox

To use the sandbox, complete the following steps:

1. Create an IAM role:

   Create an IAM role in the AWS account linked with your AWS Partner Central account.
2. Assign Policy:

   Assign the following policy to the IAM role. For more information, see Adding and removing IAM identity permissions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RevenueMeasurementCreation",
            "Effect": "Allow",
            "Action": [
                "partnercentral:CreateRevenueAttribution",
                "partnercentral:CreateMarketplaceRevenueShare",
                "partnercentral:CreateMarketplaceRevenueShareAllocation"
            ],
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "RevenueMeasurementListing",
            "Effect": "Allow",
            "Action": [
                "partnercentral:ListRevenueAttributions",
                "partnercentral:ListRevenueAttributionAllocations",
                "partnercentral:ListMarketplaceRevenueShares",
                "partnercentral:ListMarketplaceRevenueShareAllocations"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "RevenueAttributionManagement",
            "Effect": "Allow",
            "Action": [
                "partnercentral:GetRevenueAttribution",
                "partnercentral:UpdateRevenueAttribution",
                "partnercentral:GetRevenueAttributionAllocation",
                "partnercentral:StartRevenueAttributionAllocationsTask",
                "partnercentral:GetRevenueAttributionAllocationsTask"
            ],
            "Resource": "arn:aws:partnercentral:*:*:catalog/Sandbox/revenue-
attribution/*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "Sandbox"
                    ]
                }
            }
```

```
        },
        {
            "Sid": "MarketplaceRevenueShareManagement",
            "Effect": "Allow",
            "Action": [
                "partnercentral:GetMarketplaceRevenueShare",
                "partnercentral:GetMarketplaceRevenueShareAllocation",
                "partnercentral:UpdateMarketplaceRevenueShareAllocation"
            ],
            "Resource": "arn:aws:partnercentral:*:*:catalog/Sandbox/marketplace-
revenue-share/*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "TaggingAccess",
            "Effect": "Allow",
            "Action": [
                "partnercentral:TagResource",
                "partnercentral:UntagResource",
                "partnercentral:ListTagsForResource"
            ],
            "Resource": [
                "arn:aws:partnercentral:*:*:catalog/Sandbox/revenue-attribution/*",
                "arn:aws:partnercentral:*:*:catalog/Sandbox/marketplace-revenue-
share/*"
            ],
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "OpportunityAccess",
            "Effect": "Allow",
            "Action": [
```

```
                "partnercentral:ListOpportunities",
                "partnercentral:GetOpportunity"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "partnercentral:Catalog": [
                        "Sandbox"
                    ]
                }
            }
        },
        {
            "Sid": "ListingAWSMarketplaceEntities",
            "Effect": "Allow",
            "Action": [
                "aws-marketplace:ListEntities"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AWSMarketplaceEntityAccess",
            "Effect": "Allow",
            "Action": [
                "aws-marketplace:DescribeEntity"
            ],
            "Resource": [
                "arn:aws:aws-marketplace:*:*:AWSMarketplace*/Offer/*"
            ]
        }
    ]
}
```

3. Use IAM role credentials:

   Use the credentials (secret key and access key) of this IAM role in your solution to perform the API actions.

Testing AWS actions

During the testing phase, it is often necessary to simulate AWS actions. This simulation enables partners to thoroughly test the complete end-to-end flow of their integration with AWS services. Each request includes a catalog parameter, which determines the data environment.

Simulating the creation of a Revenue Attribution ID

To simulate the creation of a Revenue Attribution ID, in the payload of the `CreateRevenueAttribution` action, include `"Catalog": "Sandbox"` in the payload.

For example:

```
{
  "Catalog": "Sandbox",
  "ClientToken": "String",
  "Name": "String",
  "Description": "String",
  "TenancyModel": "MULTI_TENANT | SINGLE_TENANT",
  "ProductIdentifier": "String",
  "Tags": [
    {
      "Key": "String",
      "Value": "String"
    }
  ]
}
```

Additional testing notes

1. To test other actions, set `"Catalog": "Sandbox"` in the payload.
2. To move to production, change the catalog value from `Sandbox` to `AWS`.

Code examples for Partner Central using AWS SDKs

The following code examples show how to use Partner Central with an AWS software development kit (SDK).

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios.

Scenarios are code examples that show you how to accomplish specific tasks by calling multiple functions within a service or combined with other AWS services.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Code examples

Basic examples for Partner Central using AWS SDKs

The following code examples show how to use the basics of AWS Partner Central with AWS SDKs.

Examples

- Actions for Partner Central using AWS SDKs
  - Use AssignOpportunity with an AWS SDK
  - Use AssociateOpportunity with an AWS SDK
  - Use CreateOpportunity with an AWS SDK
  - Use DisassociateOpportunity with an AWS SDK
  - Use GetAwsOpportunitySummary with an AWS SDK
  - Use GetEngagementInvitation with an AWS SDK
  - Use GetOpportunity with an AWS SDK
  - Use ListEngagementInvitations with an AWS SDK
  - Use ListOpportunities with an AWS SDK
  - Use ListSolutions with an AWS SDK
  - Use RejectEngagementInvitation with an AWS SDK
  - Use StartEngagementByAcceptingInvitationTask with an AWS SDK
  - Use StartEngagementFromOpportunityTask with an AWS SDK
  - Use UpdateOpportunity with an AWS SDK

Actions for Partner Central using AWS SDKs

The following code examples demonstrate how to perform individual Partner Central actions with AWS SDKs. Each example includes a link to GitHub, where you can find instructions for setting up and running the code.

These excerpts call the Partner Central API and are code excerpts from larger programs that must be run in context. You can see actions in context in Scenarios for Partner Central using AWS SDKs.

The following examples include only the most commonly used actions. For a complete list, see the AWS Partner Central API Reference.

Examples

- Use AssignOpportunity with an AWS SDK

- Use AssociateOpportunity with an AWS SDK
- Use CreateOpportunity with an AWS SDK
- Use DisassociateOpportunity with an AWS SDK
- Use GetAwsOpportunitySummary with an AWS SDK
- Use GetEngagementInvitation with an AWS SDK
- Use GetOpportunity with an AWS SDK
- Use ListEngagementInvitations with an AWS SDK
- Use ListOpportunities with an AWS SDK
- Use ListSolutions with an AWS SDK
- Use RejectEngagementInvitation with an AWS SDK
- Use StartEngagementByAcceptingInvitationTask with an AWS SDK
- Use StartEngagementFromOpportunityTask with an AWS SDK
- Use UpdateOpportunity with an AWS SDK

Use `AssignOpportunity` with an AWS SDK

The following code examples show how to use `AssignOpportunity`.

Java

SDK for Java 2.x

Reassign an existing Opportunity to another user.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityRequest;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssigneeContact;

/*
Purpose
PC-API-07 Assigning a new owner
*/

public class AssignOpportunity {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String assigneeFirstName = "John";

     String assigneeLastName = "Doe";

     String assigneeEmail = "test@test.com";

     String businessTitle = "PartnerAccountManager";

     AssignOpportunityResponse response = getResponse(opportunityId,
 assigneeFirstName, assigneeLastName, assigneeEmail, businessTitle);

     ReferenceCodesUtils.formatOutput(response);
    }

 static AssignOpportunityResponse getResponse(String opportunityId, String
 assigneeFirstName, String assigneeLastName, String assigneeEmail, String
 businessTitle) {

  AssignOpportunityRequest assignOpportunityRequest =
 AssignOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
```

```
            .identifier(opportunityId)
            .assignee(AssigneeContact.builder()
              .firstName(assigneeFirstName)
              .lastName(assigneeLastName)
              .email(assigneeEmail)
              .businessTitle(businessTitle)
              .build())
            .build();

        AssignOpportunityResponse response =
 client.assignOpportunity(assignOpportunityRequest);

        return response;
 }
}
```

- For API details, see AssignOpportunity in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Reassign an existing Opportunity to another user.

```
#!/usr/bin/env python

"""
Purpose
PC-API-07 Assigning a new owner
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
```

```
        region_name='us-east-1'
)

def assign_opportunity(identifier):
    assign_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier,
        "Assignee": {
            "BusinessTitle": "OpportunityOwner",
            "Email": "test@test.com",
            "FirstName": "John",
            "LastName": "Doe"
        }
    }
    try:
        # Perform an API call
        response =
 partner_central_client.assign_opportunity(**assign_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    identifier = "O4236468"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Assigning a new owner to an opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(assign_opportunity(identifier))

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

- For API details, see AssignOpportunity in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `AssociateOpportunity` with an AWS SDK

The following code examples show how to use `AssociateOpportunity`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Update associated entity of an opportunity

Java

SDK for Java 2.x

Create a formal association between an Opportunity and various related entities.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse

/*
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/
```

```
public class AssociateOpportunity {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";

     String entityIdentifier = "S-0000000";

     AssociateOpportunityResponse response = getResponse(opportunityId,
 entityType, entityIdentifier );

     ReferenceCodesUtils.formatOutput(response);
    }

static AssociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

        AssociateOpportunityRequest associateOpportunityRequest =
 AssociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        AssociateOpportunityResponse response =
 client.associateOpportunity(associateOpportunityRequest);

        return response;
 }
}
```

- For API details, see AssociateOpportunity in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Create a formal association between an Opportunity and various related entities.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def associate_opportunity(entity_type, entity_identifier, opportunityIdentifier):
    associate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : entity_type,
        "RelatedEntityIdentifier" : entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.associate_opportunity(**associate_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
```

```
        print(err.response)

def usage_demo():
    #entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
    entity_type = "Solutions"
    entity_identifier = "S-0059717"
    opportunityIdentifier = "05465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Associate Opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(associate_opportunity(entity_type,
 entity_identifier, opportunityIdentifier))

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

- For API details, see AssociateOpportunity in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `CreateOpportunity` with an AWS SDK

The following code examples show how to use `CreateOpportunity`.

.NET

SDK for .NET

Create an opportunity.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0

using System;
using Newtonsoft.Json;
using Amazon;
```

```
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;

namespace AWSExample
{
    class Program
    {
        static readonly string catalogToUse = "AWS";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                var client = new
 AmazonPartnerCentralSellingClient(awsCredentials);

                var request = new CreateOpportunityRequest
                {
                    Catalog = catalogToUse,
                    Origin = "Partner Referral",
                    Customer = new Customer
                    {
                        Account = new Account
                        {
                            Address = new Address
                            {
                                CountryCode = "US",
                                PostalCode = "99502",
                                StateOrRegion = "Alaska"
                            },
                            CompanyName = "TestCompanyName",
                            Duns = "123456789",
                            WebsiteUrl = "www.test.io",
                            Industry = "Automotive"
                        },
                        Contacts = new List<Contact>
                        {
                            new Contact
```

```
                            {
                                Email = "test@test.io",
                                FirstName = "John  ",
                                LastName = "Doe",
                                Phone = "+14444444444",
                                BusinessTitle = "test title"
                            }
                        }
                    },
                    LifeCycle = new LifeCycle
                    {
                        ReviewStatus = "Submitted",
                        TargetCloseDate = "2024-12-30"
                    },
                    Marketing = new Marketing
                    {
                        Source = "None"
                    },
                    OpportunityType = "Net New Business",
                    PrimaryNeedsFromAws = new List<string> { "Co-Sell -
Architectural Validation" },
                    Project = new Project
                    {
                        Title = "Moin Test UUID",
                        CustomerBusinessProblem = "Sandbox is not working as
expected",
                        CustomerUseCase = "AI Machine Learning and Analytics",
                        DeliveryModels = new List<string> { "SaaS or PaaS" },
                        ExpectedCustomerSpend = new List<ExpectedCustomerSpend>
                        {
                            new ExpectedCustomerSpend
                            {
                                Amount = "2000.0",
                                CurrencyCode = "USD",
                                Frequency = "Monthly",
                                TargetCompany = "Ibexlabs"
                            }
                        },
                        SalesActivities = new List<string> { "Initialized
discussions with customer" }
                    }
                };

                try
```

```
                {
                    var response = await client.CreateOpportunityAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    string formattedJson = JsonConvert.SerializeObject(response,
 Formatting.Indented);
                    Console.WriteLine(formattedJson);
                }
                catch (ValidationException ex)
                {
                    Console.WriteLine("Validation error: " + ex.Message);
                }
                catch (AmazonPartnerCentralSellingException e)
                {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
        }
    }
}
```

- For API details, see CreateOpportunity in *AWS SDK for .NET API Reference*.

Java

SDK for Java 2.x

Create an opportunity.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;
```

```
import org.example.entity.Root;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import software.amazon.awssdk.services.partnercentralselling.model.MonetaryValue;
import
 software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import
 software.amazon.awssdk.services.partnercentralselling.model.SoftwareRevenue;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

public class CreateOpportunity {

 static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
```

```
            .build();

public static void main(String[] args) {

 String inputFile = "CreateOpportunity2.json";

 if (args.length > 0)
  inputFile = args[0];

 CreateOpportunityResponse response = createOpportunity(inputFile);

 client.close();
}

static CreateOpportunityResponse createOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

 Root root = GSON.fromJson(inputString, Root.class);

 List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
 if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
  for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
   NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
     .time(Instant.parse(nextStepsHistoryJson.time))
     .value(nextStepsHistoryJson.value)
                 .build();
   nextStepsHistories.add(nextStepsHistory);
  }
 }

 LifeCycle lifeCycle = null;
 if ( root.lifeCycle != null ) {
  lifeCycle = LifeCycle.builder()
   .closedLostReason(root.lifeCycle.closedLostReason)
   .nextSteps(root.lifeCycle.nextSteps)
   .nextStepsHistory(nextStepsHistories)
   .reviewComments(root.lifeCycle.reviewComments)
   .reviewStatus(root.lifeCycle.reviewStatus)
   .reviewStatusReason(root.lifeCycle.reviewStatusReason)
   .stage(root.lifeCycle.stage)
   .targetCloseDate(root.lifeCycle.targetCloseDate)
   .build();
```

```
 }

 Marketing marketing = null;
 if ( root.marketing != null ) {
  marketing = Marketing.builder()
    .awsFundingUsed(root.marketing.awsFundingUsed)
    .campaignName(root.marketing.campaignName)
    .channels(root.marketing.channels)
    .source(root.marketing.source)
    .useCases(root.marketing.useCases)
    .build();

 }

 Address address = null;
 if ( root.customer != null && root.customer.account != null &&
root.customer.account.address != null ) {
  address = Address.builder()
   .city(root.customer.account.address.city)
               .postalCode(root.customer.account.address.postalCode)
               .stateOrRegion(root.customer.account.address.stateOrRegion)
               .countryCode(root.customer.account.address.countryCode)
               .streetAddress(root.customer.account.address.streetAddress)
               .build();
 }

 Account account = null;
 if ( root.customer != null && root.customer.account!= null) {
  account = Account.builder()
            .address(address)
            .awsAccountId(root.customer.account.awsAccountId)
               .duns(root.customer.account.duns)
               .industry(root.customer.account.industry)
               .otherIndustry(root.customer.account.otherIndustry)
               .companyName(root.customer.account.companyName)
               .websiteUrl(root.customer.account.websiteUrl)
               .build();
 }

 List<Contact> contacts = new ArrayList<Contact>();
 if ( root.customer != null && root.customer.contacts != null) {
  for (org.example.entity.Contact jsonContact : root.customer.contacts) {
   Contact contact = Contact.builder()
                 .email(jsonContact.email)
```

```
                    .firstName(jsonContact.firstName)
                    .lastName(jsonContact.lastName)
                    .phone(jsonContact.phone)
                    .businessTitle(jsonContact.businessTitle)
                    .build();
    contacts.add(contact);
   }
  }

  Customer customer = Customer.builder()
    .account(account)
    .contacts(contacts)
    .build();

  Contact oportunityTeamContact = null;
  if (root.opportunityTeam != null && root.opportunityTeam.get(0) != null ) {
   oportunityTeamContact = Contact.builder()
                .firstName(root.opportunityTeam.get(0).firstName)
                .lastName(root.opportunityTeam.get(0).lastName)
                .email(root.opportunityTeam.get(0).email)
                .phone(root.opportunityTeam.get(0).phone)
                .businessTitle(root.opportunityTeam.get(0).businessTitle)
                .build();
  }

  List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
  if ( root.project != null && root.project.expectedCustomerSpend != null) {
   for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
    ExpectedCustomerSpend expectedCustomerSpend = null;
    expectedCustomerSpend = ExpectedCustomerSpend.builder()
      .amount(expectedCustomerSpendJson.amount)
      .currencyCode(expectedCustomerSpendJson.currencyCode)
      .frequency(expectedCustomerSpendJson.frequency)
      .targetCompany(expectedCustomerSpendJson.targetCompany)
      .build();
    expectedCustomerSpends.add(expectedCustomerSpend);
   }
        }

        Project project = null;
        if ( root.project != null) {
         project = Project.builder()
```

```
                .title(root.project.title)
                .customerBusinessProblem(root.project.customerBusinessProblem)
                .customerUseCase(root.project.customerUseCase)
                .deliveryModels(root.project.deliveryModels)
                .expectedCustomerSpend(expectedCustomerSpends)
                .salesActivities(root.project.salesActivities)
                .competitorName(root.project.competitorName)
                .otherSolutionDescription(root.project.otherSolutionDescription)
                .build();
        }

        SoftwareRevenue softwareRevenue = null;
        if ( root.softwareRevenue != null) {
         MonetaryValue monetaryValue = null;
         if ( root.softwareRevenue.value != null) {
          monetaryValue = MonetaryValue.builder()
             .amount(root.softwareRevenue.value.amount)
             .currencyCode(root.softwareRevenue.value.currencyCode)
             .build();
         }
         softwareRevenue = SoftwareRevenue.builder()
           .deliveryModel(root.softwareRevenue.deliveryModel)
           .effectiveDate(root.softwareRevenue.effectiveDate)
           .expirationDate(root.softwareRevenue.expirationDate)
           .value(monetaryValue)
           .build();
        }

 // Building the Actual CreateOpportunity Request
 CreateOpportunityRequest createOpportunityRequest =
CreateOpportunityRequest.builder()
   .catalog(CATALOG_TO_USE)
   .clientToken(root.clientToken)
   .primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws)
   .opportunityType(root.opportunityType)
   .lifeCycle(lifeCycle)
   .marketing(marketing)
   .nationalSecurity(root.nationalSecurity)
   .origin(root.origin)
   .customer(customer)
   .project(project)
   .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
   .opportunityTeam(oportunityTeamContact)
   .softwareRevenue(softwareRevenue)
```

```
    .build();

  CreateOpportunityResponse response =
 client.createOpportunity(createOpportunityRequest);
  System.out.println("Successfully created: " + response);

  return response;
    }

}
```

- For API details, see CreateOpportunity in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Create an opportunity.

```
#!/usr/bin/env python
import boto3
import logging
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
import utils.helpers as helper
import utils.stringify_details as sd
from botocore.client import ClientError
from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

def create_opportunity(partner_central_client):
    create_opportunity_request = helper.remove_nulls(sd.stringify_json("src/
create_opportunity/createOpportunity.json"))
    try:
        # Perform an API call
        response =
 partner_central_client.create_opportunity(**create_opportunity_request)

        helper.pretty_print_datetime(response)
```

```
        # Retrieve the opportunity details
        get_response = partner_central_client.get_opportunity(
            Identifier=response["Id"],
            Catalog=CATALOG_TO_USE
        )
        helper.pretty_print_datetime(get_response)
        return response
    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

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

    print("-" * 88)
    print("Create Opportunity.")
    print("-" * 88)

    partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
    )

    create_opportunity(partner_central_client)

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

- For API details, see CreateOpportunity in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `DisassociateOpportunity` with an AWS SDK

The following code examples show how to use `DisassociateOpportunity`.

Action examples are code excerpts from larger programs and must be run in context. You can see this action in context in the following code example:

- Update associated entity of an opportunity

Java

SDK for Java 2.x

Remove an existing association between an Opportunity and related entities.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityReque
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRespo

/*
Purpose
PC-API -14 Removing a Solution
PC-API -15 Removing an offer
PC-API -16 Removing a product
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/

public class DisassociateOpportunity {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

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

```
    String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

    String entityType = "Solutions";

    String entityIdentifier = "S-0000000";

    DisassociateOpportunityResponse response = getResponse(opportunityId,
entityType, entityIdentifier );

    ReferenceCodesUtils.formatOutput(response);
   }

static DisassociateOpportunityResponse getResponse(String opportunityId, String
entityType, String entityIdentifier) {

 DisassociateOpportunityRequest disassociateOpportunityRequest =
DisassociateOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .opportunityIdentifier(opportunityId)
         .relatedEntityType(entityType)
         .relatedEntityIdentifier(entityIdentifier)
         .build();

       DisassociateOpportunityResponse response =
client.disassociateOpportunity(disassociateOpportunityRequest);

       return response;
 }
}
```

- For API details, see DisassociateOpportunity in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Remove an existing association between an Opportunity and related entities.

```
#!/usr/bin/env python
```

```
"""
Purpose
PC-API -14 Removing a Solution
PC-API -15 Removing an offer
PC-API -16 Removing a product
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def disassociate_opportunity(entity_type, entity_identifier,
 opportunityIdentifier):
    disassociate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : entity_type,
        "RelatedEntityIdentifier" : entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.disassociate_opportunity(**disassociate_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    #entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
    entity_type = "Solutions"
    entity_identifier = "S-0049999"
    opportunityIdentifier = "O4397574"
```

```
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get updated Opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(disassociate_opportunity(entity_type,
 entity_identifier, opportunityIdentifier))

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

- For API details, see DisassociateOpportunity in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetAwsOpportunitySummary` with an AWS SDK

The following code examples show how to use `GetAwsOpportunitySummary`.

Java

SDK for Java 2.x

Retrieves a summary of an AWS Opportunity.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryRequ
```

```
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryResp

/*
 * Purpose
 * PC-API-25 Retrieves a summary of an AWS Opportunity.
 */

public class GetAwsOpportunitySummary {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetAwsOpportunitySummaryResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetAwsOpportunitySummaryResponse getResponse(String opportunityId)
 {

  GetAwsOpportunitySummaryRequest getOpportunityRequest =
 GetAwsOpportunitySummaryRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .relatedOpportunityIdentifier(opportunityId)
          .build();

  GetAwsOpportunitySummaryResponse response =
 client.getAwsOpportunitySummary(getOpportunityRequest);

        return response;
 }
}
```

- For API details, see GetAwsOpportunitySummary in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Retrieves a summary of an AWS Opportunity.

```
#!/usr/bin/env python

"""
Purpose
PC-API-25 Retrieves a summary of an AWS Opportunity.
 LifeCycle.ReviewStatus=Approved
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "RelatedOpportunityIdentifier": identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_aws_opportunity_summary(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
```

```
    identifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get AWS Opportunity summary.")
    print("-" * 88)

    helper.pretty_print_datetime(get_opportunity(identifier))

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

- For API details, see GetAwsOpportunitySummary in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetEngagementInvitation` with an AWS SDK

The following code examples show how to use `GetEngagementInvitation`.

Java

SDK for Java 2.x

Retrieves the details of an engagement invitation shared by AWS with a partner.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationReque
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRespo

/*
 * Purpose
 * PC-API-22 Get engagement invitation opportunity
 */

public class GetEngagementInvitation {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static GetEngagementInvitationResponse getResponse(String opportunityId) {

  GetEngagementInvitationRequest getOpportunityRequest =
 GetEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

  GetEngagementInvitationResponse response =
 client.getEngagementInvitation(getOpportunityRequest);

        return response;
 }
}
```

- For API details, see GetEngagementInvitation in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Retrieves the details of an engagement invitation shared by AWS with a partner.

```
#!/usr/bin/env python

"""
Purpose
PC-API-22  GetOpportunityEngagementInvitation - Retrieves details of a specific
 engagement invitation.
This operation allows partners to view the invitation and its associated
 information,
such as the customer, project, and lifecycle details.
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity_engagement_invitation(identifier):
    get_opportunity_engagement_invitation_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier
    }
    try:
        # Perform an API call
```

```
        response =
 partner_central_client.get_engagement_invitation(**get_opportunity_engagement_invitation
        return response

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    identifier = "arn:aws:partnercentral-selling:us-east-1:aws:catalog/Sandbox/
engagement-invitation/engi-0000000IS0Qga"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Given the ARN identifier, retrieve details of Opportunity Engagement
 Invitation.")
    print("-" * 88)


 helper.pretty_print_datetime(get_opportunity_engagement_invitation(identifier))

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

- For API details, see GetEngagementInvitation in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `GetOpportunity` with an AWS SDK

The following code examples show how to use `GetOpportunity`.

.NET

SDK for .NET

Get an opportunity.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0

using System;
using Newtonsoft.Json;
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;

namespace AWSExample
{
    class Program
    {
        static readonly string catalogToUse = "AWS";
        static readonly string identifier = "O1111111";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                var client = new
 AmazonPartnerCentralSellingClient(awsCredentials);

                var request = new GetOpportunityRequest
                {
                    Catalog = catalogToUse,
                    Identifier = identifier
                };

                try {
                    var response = await client.GetOpportunityAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    string formattedJson = JsonConvert.SerializeObject(response,
 Formatting.Indented);
                    Console.WriteLine(formattedJson);
                } catch(ValidationException ex) {
                    Console.WriteLine("Validation error: " + ex.Message);
```

```
                } catch (AmazonPartnerCentralSellingException e) {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
        }
    }
}
```

- For API details, see GetOpportunity in *AWS SDK for .NET API Reference*.

Go

SDK for Go V2

Get an opportunity.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"
 "log"

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

func main() {
 config, err := config.LoadDefaultConfig(context.TODO())

 if err != nil {
  log.Fatal(err)
```

```
 }

 config.Region = "us-east-1"

 client := partnercentralselling.NewFromConfig(config)

 output, err := client.GetOpportunity(context.TODO(),
 &partnercentralselling.GetOpportunityInput{
  Identifier: aws.String("O1111111"),
  Catalog:    aws.String("AWS"),
 })

 if err != nil {
  log.Fatal(err)
 }
 log.Println("printing opportuniy...\n")

 jsonOutput, err := json.MarshalIndent(output, "", "    ")

 fmt.Println(string(jsonOutput))
}
```

- For API details, see GetOpportunity in *AWS SDK for Go API Reference*.

Java

SDK for Java 2.x

Get an opportunity.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;

/*
 * Purpose
 * PC-API-08 Get updated Opportunity
 */

public class GetOpportunity {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetOpportunityResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetOpportunityResponse getResponse(String opportunityId) {

        GetOpportunityRequest getOpportunityRequest =
 GetOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

        GetOpportunityResponse response =
 client.getOpportunity(getOpportunityRequest);

        return response;
 }
}
```

- For API details, see GetOpportunity in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Get an opportunity.

```
#!/usr/bin/env python

"""
Purpose
PC-API -08 Get updated Opportunity given opportunity id
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_opportunity(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)
```

```
def usage_demo():
    identifier = "05465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get updated Opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(get_opportunity(identifier))

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

- For API details, see GetOpportunity in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListEngagementInvitations` with an AWS SDK

The following code examples show how to use `ListEngagementInvitations`.

Java

SDK for Java 2.x

Retrieves a list of engagement invitations sent to the partner.

```
package org.example;

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

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsReq
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsRes
import
 software.amazon.awssdk.services.partnercentralselling.model.ParticipantType;
import
 software.amazon.awssdk.services.partnercentralselling.model.EngagementInvitationSummary;

public class ListEngagementInvitations {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     List<EngagementInvitationSummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    static List<EngagementInvitationSummary> getResponse() {

  List<EngagementInvitationSummary> opportunitySummaries = new
 ArrayList<EngagementInvitationSummary>();

  ListEngagementInvitationsRequest listOpportunityRequest =
 ListEngagementInvitationsRequest.builder()
                .catalog(CATALOG_TO_USE)
                .participantType(ParticipantType.RECEIVER)
          .maxResults(5)
          .build();

  ListEngagementInvitationsResponse response =
 client.listEngagementInvitations(listOpportunityRequest);

     opportunitySummaries.addAll(response.engagementInvitationSummaries());

     client.close();
```

```
        return opportunitySummaries;
 }
}
```

- For API details, see ListEngagementInvitations in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Retrieves a list of engagement invitations sent to the partner.

```
#!/usr/bin/env python

"""
Purpose
PC-API-21 ListEngagementInvitations - Retrieves a list of engagement invitations
 based on specified criteria.
This operation allows partners to view all invitations to engagement.
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def list_engagement_invitations():
    list_engagement_invitations_request ={
        "Catalog": CATALOG_TO_USE,
        "MaxResults": 20
    }
    try:
```

```
        # Perform an API call
        response =
 partner_central_client.list_engagement_invitations(**list_engagement_invitations_request
        return response

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

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

    print("-" * 88)
    print("Retrieve list of Engagement Invitations.")
    print("-" * 88)

    helper.pretty_print_datetime(list_engagement_invitations())

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

- For API details, see ListEngagementInvitations in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListOpportunities` with an AWS SDK

The following code examples show how to use `ListOpportunities`.

.NET

SDK for .NET

List opportunities.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// PDX-License-Identifier: Apache-2.0
```

```
using System;
using Newtonsoft.Json;
using Amazon;
using Amazon.Runtime;
using Amazon.PartnerCentralSelling;
using Amazon.PartnerCentralSelling.Model;

namespace AWSExample
{
    class Program
    {
        static readonly string catalogToUse = "Sandbox";
        static async Task Main(string[] args)
        {
            // Initialize credentials from .aws/credentials file
            var credentials = new
 Amazon.Runtime.CredentialManagement.SharedCredentialsFile();
            if (credentials.TryGetProfile("default", out var profile))
            {
                AWSCredentials awsCredentials =
 profile.GetAWSCredentials(credentials);

                //var config = new AmazonPartnerCentralSellingConfig()
                //{
                //    ServiceURL = "https://partnercentral-selling.us-
east-1.api.aws",
                //};
                //var client = new
 AmazonPartnerCentralSellingClient(awsCredentials, config);
                var client = new
 AmazonPartnerCentralSellingClient(awsCredentials);
                var request = new ListOpportunitiesRequest
                {
                    Catalog = catalogToUse,
                    MaxResults = 2
                };

                try {
                    var response = await client.ListOpportunitiesAsync(request);
                    Console.WriteLine(response.HttpStatusCode);
                    foreach (var opportunity in response.OpportunitySummaries)
                    {
                        Console.WriteLine("Opportunity id: " + opportunity.Id);
                    }
```

```
                    string formattedJson =
 JsonConvert.SerializeObject(response.OpportunitySummaries, Formatting.Indented);
                    Console.WriteLine(formattedJson);
                } catch(ValidationException ex) {
                    Console.WriteLine("Validation error: " + ex.Message);
                } catch (AmazonPartnerCentralSellingException e) {
                    Console.WriteLine("Failed:");
                    Console.WriteLine(e.RequestId);
                    Console.WriteLine(e.ErrorCode);
                    Console.WriteLine(e.Message);
                }
            }
            else
            {
                Console.WriteLine("Profile not found.");
            }
        }
    }
}
```

- For API details, see ListOpportunities in *AWS SDK for .NET API Reference*.

Go

SDK for Go V2

List opportunities.

```
package main

import (
 "context"
 "encoding/json"
 "fmt"
 "log"

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

```
func main() {
 config, err := config.LoadDefaultConfig(context.TODO())

 if err != nil {
  log.Fatal(err)
 }

 config.Region = "us-east-1"

 client := partnercentralselling.NewFromConfig(config)

 output, err := client.ListOpportunities(context.TODO(),
 &partnercentralselling.ListOpportunitiesInput{
  MaxResults: aws.Int32(2),
  Catalog:    aws.String("AWS"),
 })

 if err != nil {
  log.Fatal(err)
 }

 jsonOutput, err := json.MarshalIndent(output, "", "    ")
 fmt.Println(string(jsonOutput))
}
```

- For API details, see ListOpportunities in *AWS SDK for Go API Reference*.

Java

SDK for Java 2.x

List opportunities.

```
package org.example;

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

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
```

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.OpportunitySummary;

/*
 * Purpose
 * PC-API-18 Getting list of Opportunities
 */

public class ListOpportunititesPaging {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
           .region(Region.US_EAST_1)
           .credentialsProvider(DefaultCredentialsProvider.create())
           .httpClient(ApacheHttpClient.builder().build())
           .build();

   public static void main(String[] args) {
    List<OpportunitySummary> opportunitySummaries = getResponse();
       ReferenceCodesUtils.formatOutput(opportunitySummaries);
   }

   private static List<OpportunitySummary> getResponse() {
    List<OpportunitySummary> opportunitySummaries = new
 ArrayList<OpportunitySummary>();

  ListOpportunitiesRequest listOpportunityRequest =
 ListOpportunitiesRequest.builder()
               .catalog(CATALOG_TO_USE)
         .maxResults(5)
         .build();

    ListOpportunitiesResponse response =
 client.listOpportunities(listOpportunityRequest);

    opportunitySummaries.addAll(response.opportunitySummaries());
```

```
    while (response.nextToken() != null && response.nextToken().length() > 0) {
     listOpportunityRequest = ListOpportunitiesRequest.builder()
                  .catalog(CATALOG_TO_USE)
             .maxResults(5)
             .nextToken(response.nextToken())
             .build();
     response = client.listOpportunities(listOpportunityRequest);
     opportunitySummaries.addAll(response.opportunitySummaries());
    }

    client.close();

       return opportunitySummaries;
 }
}
```

- For API details, see ListOpportunities in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

List opportunities.

```
#!/usr/bin/env python

"""
Purpose
PC-API -18 Getting list of Opportunities
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
```

```
        service_name=serviceName,
        region_name='us-east-1'
)

def get_list_of_opportunities():

    opportunity_list = []

    list_opportunities_request ={
        "Catalog": CATALOG_TO_USE,
     "MaxResults": 20
    }
    try:
        # Perform an API call
        response =
 partner_central_client.list_opportunities(**list_opportunities_request)
        opportunity_list.extend(response["OpportunitySummaries"])

        while "NextToken" in response and response["NextToken"] is not None:
            list_opportunities_request["NextToken"] = response["NextToken"]
            response =
 partner_central_client.list_opportunities(**list_opportunities_request)
            opportunity_list.extend(response["OpportunitySummaries"])

        return opportunity_list

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

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

    print("-" * 88)
    print("Getting list of Opportunities.")
    print("-" * 88)

    helper.pretty_print_datetime(get_list_of_opportunities())

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

- For API details, see ListOpportunities in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `ListSolutions` with an AWS SDK

The following code examples show how to use `ListSolutions`.

Java

SDK for Java 2.x

Retrieves a list of Partner Solutions that the partner registered on Partner Central.

```
package org.example;

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

import static org.example.utils.Constants.*;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsResponse;
import software.amazon.awssdk.services.partnercentralselling.model.SolutionBase;

/*
 * Purpose
 * PC-API-10 Getting list of solutions
 */

public class ListSolutions {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
```

```
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<SolutionBase> solutionSummaries = getResponse();
        ReferenceCodesUtils.formatOutput(solutionSummaries);
    }

    static List<SolutionBase> getResponse() {
  List<SolutionBase> solutionSummaries = new ArrayList<SolutionBase>();

  ListSolutionsRequest listSolutionsRequest = ListSolutionsRequest.builder()
    .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();

     ListSolutionsResponse response = client.listSolutions(listSolutionsRequest);

     solutionSummaries.addAll(response.solutionSummaries());

        return solutionSummaries;
 }
}
```

- For API details, see ListSolutions in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Retrieves a list of Partner Solutions that the partner registered on Partner Central.

```
#!/usr/bin/env python

"""
Purpose
PC-API-10 Getting list of solutions
"""
import logging
import boto3
```

```
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_list_of_solutions():
    list_solutions_request ={
        "Catalog": CATALOG_TO_USE,
     "MaxResults": 20
    }
    try:
        # Perform an API call
        response =
 partner_central_client.list_solutions(**list_solutions_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

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

    print("-" * 88)
    print("Getting list of solutions.")
    print("-" * 88)

    helper.pretty_print_datetime(get_list_of_solutions())

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

- For API details, see ListSolutions in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `RejectEngagementInvitation` with an AWS SDK

The following code examples show how to use `RejectEngagementInvitation`.

Java

SDK for Java 2.x

Rejects an EngagementInvitation that AWS shared.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationRe
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationRe

/*
 * Purpose
 * PC-API-05 AWS Originated(AO) rejection
 */

public class RejectEngagementInvitation {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
```

```
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

  RejectEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static RejectEngagementInvitationResponse getResponse(String invitationId) {

        RejectEngagementInvitationRequest rejectOpportunityRequest =
 RejectEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .rejectionReason("Unable to support")
          .build();

  RejectEngagementInvitationResponse response =
 client.rejectEngagementInvitation(rejectOpportunityRequest);

        return response;
 }
}
```

- For API details, see [RejectEngagementInvitation](#) in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Rejects an EngagementInvitation that AWS shared.

```
#!/usr/bin/env python

"""
Purpose
PC-API-05 AWS Originated AO rejection - RejectOpportunityEngagementInvitation -
 Rejects a engagement invitation.
```

```
This action indicates that the partner does not wish to participate in the
 engagement and
provides a reason for the rejection.
Upon rejection, a OpportunityEngagementInvitationRejected event is triggered.
Subsequently, the invitation will no longer be available for the partner to act
 on.
"""
import json
import logging
import boto3
import utils.helpers as helper

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def reject_opportunity_engagement_invitation(identifier, reject_reason):
    reject_opportunity_engagement_invitation_request ={
        "Catalog": CATALOG_TO_USE,
     "Identifier": identifier,
        "RejectionReason": reject_reason
    }
    try:
        # Perform an API call
        response =
 partner_central_client.reject_engagement_invitation(**reject_opportunity_engagement_invi
        return response

    except Exception as err:
        # Catch all client exceptions
        print(json.dumps(err.response))

def usage_demo():
    identifier = "arn:aws:partnercentral:us-east-1::catalog/Sandbox/engagement-
invitation/engi-0000002isviga"
    reject_reason = "Customer problem unclear"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    print("-" * 88)
    print("Given the ARN identifier and reject reason, reject the Opportunity
 Engagement Invitation.")
    print("-" * 88)


 helper.pretty_print_datetime(reject_opportunity_engagement_invitation(identifier,
 reject_reason))

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

- For API details, see RejectEngagementInvitation in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `StartEngagementByAcceptingInvitationTask` with an AWS SDK

The following code examples show how to use `StartEngagementByAcceptingInvitationTask`.

Java

SDK for Java 2.x

Starts the engagement by accepting an EngagementInvitation.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingIn
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingIn
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationReque
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRespo
import
 software.amazon.awssdk.services.partnercentralselling.model.InvitationStatus;

/*
Purpose
PC-API-04: Start Engagement By Accepting InvitationTask for AWS Originated(AO)
 opportunity
*/

public class StartEngagementByAcceptingInvitationTask {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 static String clientToken = "test-a30d161";

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     StartEngagementByAcceptingInvitationTaskResponse response =
 getResponse(opportunityId);

     if ( response == null) {
      System.out.println("Opportunity is not AWS Originated.");
     } else {
      ReferenceCodesUtils.formatOutput(response);
     }
    }
```

```
    private static GetEngagementInvitationResponse getInvitation(String
 invitationId) {

     GetEngagementInvitationRequest getRequest =
 GetEngagementInvitationRequest.builder()
          .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .build();

  GetEngagementInvitationResponse response =
 client.getEngagementInvitation(getRequest);

        return response;
 }

 static StartEngagementByAcceptingInvitationTaskResponse getResponse(String
 invitationId) {

  if ( getInvitation(invitationId).status().equals(InvitationStatus.PENDING)) {
   StartEngagementByAcceptingInvitationTaskRequest acceptOpportunityRequest =
     StartEngagementByAcceptingInvitationTaskRequest.builder()
     .catalog(Constants.CATALOG_TO_USE)
           .identifier(invitationId)
           .clientToken(clientToken)
           .build();

   StartEngagementByAcceptingInvitationTaskResponse response =
 client.startEngagementByAcceptingInvitationTask(acceptOpportunityRequest);
         return response;
  }
  return null;
 }
}
```

- For API details, see StartEngagementByAcceptingInvitationTask in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Starts the engagement by accepting an EngagementInvitation.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
     "Identifier": identifier,
        "Catalog": CATALOG_TO_USE
    }
    try:
        # Perform an API call
        response =
 partner_central_client.get_engagement_invitation(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def start_engagement_by_accepting_invitation_task(identifier):

    response = get_opportunity(identifier)

    if ( response['Status'] == 'PENDING') :
```

```
        accept_opportunity_engagement_invitation_request ={
            "Catalog": CATALOG_TO_USE,
         "Identifier" : identifier,
            "ClientToken": "test-123456"
        }
        try:
            # Perform an API call
            response =
 partner_central_client.start_engagement_by_accepting_invitation_task(**accept_opportunit
            return response

        except ClientError as err:
            # Catch all client exceptions
            print(err.response)
            return None
    else:
        return None

def usage_demo():
    identifier = "arn:aws:partnercentral:us-east-1::catalog/Sandbox/engagement-
invitation/engi-0000002isusga"
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Get updated Opportunity.")
    print("-" * 88)


 helper.pretty_print_datetime(start_engagement_by_accepting_invitation_task(identifier))

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

- For API details, see StartEngagementByAcceptingInvitationTask in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `StartEngagementFromOpportunityTask` with an AWS SDK

The following code examples show how to use `StartEngagementFromOpportunityTask`.

Java

SDK for Java 2.x

Initiates the engagement process from an existing opportunity by accepting the engagement invitation and creating a corresponding opportunity in the partner's system.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.AwsSubmission;
import
 software.amazon.awssdk.services.partnercentralselling.model.SalesInvolvementType;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportuni
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportuni
import software.amazon.awssdk.services.partnercentralselling.model.Visibility;

/*
 * Purpose
 * PC-API-01 Partner Originated (PO) opp submission(Start Engagement From
 Opportunity Task for AO Originated Opportunity)
 */

public class StartEngagementFromOpportunityTask {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
```

```
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     StartEngagementFromOpportunityTaskResponse response =
 getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static StartEngagementFromOpportunityTaskResponse getResponse(String
 opportunityId) {

  StartEngagementFromOpportunityTaskRequest submitOpportunityRequest =
 StartEngagementFromOpportunityTaskRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .clientToken("test-annjqwesdsd99")

 .awsSubmission(AwsSubmission.builder().involvementType(SalesInvolvementType.CO_SELL).vis
          .build();

  StartEngagementFromOpportunityTaskResponse response =
 client.startEngagementFromOpportunityTask(submitOpportunityRequest);

        return response;
 }
}
```

- For API details, see StartEngagementFromOpportunityTask in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Initiates the engagement process from an existing opportunity by accepting the engagement invitation and creating a corresponding opportunity in the partner's system.

```
#!/usr/bin/env python

"""
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def start_engagement_from_opportunity_task(identifier):

    start_engagement_from_opportunity_task_request ={
            "AwsSubmission": {
                "InvolvementType": "Co-Sell",
                "Visibility": "Full"
            },
            "Catalog": CATALOG_TO_USE,
         "Identifier" : identifier,
            "ClientToken": "test-annjqwesdsd99"
    }
    try:
            # Perform an API call
            response =
 partner_central_client.start_engagement_from_opportunity_task(**start_engagement_from_op
            return response

    except ClientError as err:
            # Catch all client exceptions
            print(err.response)
```

```
            return None

def usage_demo():
    identifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Start Engagement from Opportunity Task.")
    print("-" * 88)


 helper.pretty_print_datetime(start_engagement_from_opportunity_task(identifier))

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

- For API details, see StartEngagementFromOpportunityTask in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Use `UpdateOpportunity` with an AWS SDK

The following code examples show how to use `UpdateOpportunity`.

Java

SDK for Java 2.x

Update an opportunity.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;
```

```
import org.example.entity.Root;
import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import
 software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import software.amazon.awssdk.services.partnercentralselling.model.ReviewStatus;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityResponse;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

/*
 * Purpose
 * PC-API-02/06 Update opportunity when LifeCycle.ReviewStatus is not Submitted
 or In-Review
 */

public class UpdateOpportunity {
```

```
static final Gson GSON = new GsonBuilder()
  .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
  .registerTypeAdapter(String.class, new StringSerializer())
  .create();

static PartnerCentralSellingClient client =
PartnerCentralSellingClient.builder()
           .region(Region.US_EAST_1)
           .credentialsProvider(DefaultCredentialsProvider.create())
           .httpClient(ApacheHttpClient.builder().build())
           .build();

static String OPPORTUNITY_ORIGIN = ORIGIN_PARTNER_ORIGINATED;

public static void main(String[] args) {

 String inputFile = "updateOpportunity.json";

 if (args.length > 0)
  inputFile = args[0];

 UpdateOpportunityResponse response = updateOpportunity(inputFile);

 client.close();
}

public static GetOpportunityResponse getResponse(String opportunityId) {

       GetOpportunityRequest getOpportunityRequest =
GetOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .identifier(opportunityId)
         .build();

       GetOpportunityResponse response =
client.getOpportunity(getOpportunityRequest);
       System.out.println(opportunityId + ":" + response);
       return response;
}

public static UpdateOpportunityResponse updateOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);
```

```
 Root root = GSON.fromJson(inputString, Root.class);
 GetOpportunityResponse response = getResponse(root.identifier);

 if (response != null
   && response.lifeCycle() != null
   && response.lifeCycle().reviewStatus() != null
   && response.lifeCycle().reviewStatus() != ReviewStatus.SUBMITTED
   && response.lifeCycle().reviewStatus() != ReviewStatus.IN_REVIEW) {

  List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
  if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
   for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
    NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
      .time(Instant.parse(nextStepsHistoryJson.time))
      .value(nextStepsHistoryJson.value)
                  .build();
    nextStepsHistories.add(nextStepsHistory);
   }
  }

  LifeCycle lifeCycle = null;
  if ( root.lifeCycle != null ) {
   lifeCycle = LifeCycle.builder()
    .closedLostReason(root.lifeCycle.closedLostReason)
    .nextSteps(root.lifeCycle.nextSteps)
    .nextStepsHistory(nextStepsHistories)
    .reviewComments(root.lifeCycle.reviewComments)
    .reviewStatus(root.lifeCycle.reviewStatus)
    .reviewStatusReason(root.lifeCycle.reviewStatusReason)
    .stage(root.lifeCycle.stage)
    .targetCloseDate(root.lifeCycle.targetCloseDate)
    .build();
  }

  Marketing marketing = null;
  if ( root.marketing != null ) {
   marketing = Marketing.builder()
     .awsFundingUsed(root.marketing.awsFundingUsed)
     .campaignName(root.marketing.campaignName)
     .channels(root.marketing.channels)
     .source(root.marketing.source)
     .useCases(root.marketing.useCases)
     .build();
```

```
  }

  Address address = null;
  if (root.customer != null && root.customer.account != null &&
root.customer.account.address != null) {
   address =
Address.builder().postalCode(root.customer.account.address.postalCode)
     .stateOrRegion(root.customer.account.address.stateOrRegion)
     .countryCode(root.customer.account.address.countryCode).build();
  }

  Account account = null;
  if (root.customer != null && root.customer.account != null) {
   account = Account.builder().address(address).duns(root.customer.account.duns)

.industry(root.customer.account.industry).companyName(root.customer.account.companyName)
     .websiteUrl(root.customer.account.websiteUrl).build();
  }

  List<Contact> contacts = new ArrayList<Contact>();
  if ( root.customer != null && root.customer.contacts != null) {
   for (org.example.entity.Contact jsonContact : root.customer.contacts) {
    Contact contact = Contact.builder()
                  .email(jsonContact.email)
                  .firstName(jsonContact.firstName)
                  .lastName(jsonContact.lastName)
                  .phone(jsonContact.phone)
                  .businessTitle(jsonContact.businessTitle)
                  .build();
    contacts.add(contact);
   }
  }

  Customer customer =
Customer.builder().account(account).contacts(contacts).build();

  List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
  if ( root.project != null && root.project.expectedCustomerSpend != null) {
   for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
    ExpectedCustomerSpend expectedCustomerSpend = null;
    expectedCustomerSpend = ExpectedCustomerSpend.builder()
```

```
      .amount(expectedCustomerSpendJson.amount)
      .currencyCode(expectedCustomerSpendJson.currencyCode)
      .frequency(expectedCustomerSpendJson.frequency)
      .targetCompany(expectedCustomerSpendJson.targetCompany)
      .build();
    expectedCustomerSpends.add(expectedCustomerSpend);
   }
        }

  Project project = null;
  if (root.project != null) {
   project = Project.builder().title(root.project.title)
     .customerBusinessProblem(root.project.customerBusinessProblem)

.customerUseCase(root.project.customerUseCase).deliveryModels(root.project.deliveryModel
     .expectedCustomerSpend(expectedCustomerSpends)

.salesActivities(root.project.salesActivities).competitorName(root.project.competitorNam
     .otherSolutionDescription(root.project.otherSolutionDescription).build();
  }

  // Building the Actual CreateOpportunity Request
  UpdateOpportunityRequest updateOpportunityRequest =
UpdateOpportunityRequest.builder().catalog(root.catalog)

.identifier(root.identifier).lastModifiedDate(Instant.parse(root.lastModifiedDate))

.primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws).opportunityType(root.opportuni
    .lifeCycle(lifeCycle)
    .customer(customer)
    .project(project)
    .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
    .marketing(marketing)
    .nationalSecurity(root.nationalSecurity)
    .opportunityType(root.opportunityType)
    .build();

  UpdateOpportunityResponse updateResponse =
client.updateOpportunity(updateOpportunityRequest);
  System.out.println("Successfully updated opportunity: " + updateResponse);

  return updateResponse;
 } else {
  System.out.println("Opportunity cannot be updated.");
```

```
   return null;
  }
    }
}
```

- For API details, see UpdateOpportunity in *AWS SDK for Java 2.x API Reference*.

Python

SDK for Python (Boto3)

Update an opportunity.

```
#!/usr/bin/env python

"""
Purpose
PC-API-2  Updating Partner Originated Opportunity
"""
import logging
import boto3
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
import utils.helpers as helper
from botocore.client import ClientError
import utils.stringify_details as sd
from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
        service_name=serviceName,
        region_name='us-east-1'
)

def get_opportunity(identifier):
    get_opportunity_request ={
     "Identifier": identifier,
        "Catalog": CATALOG_TO_USE
    }
```

```
    try:
        # Perform an API call
        response =
 partner_central_client.get_opportunity(**get_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def update_opportunity():
    update_opportunity_request_orig = sd.stringify_json("src/update_opportunity/
update_opportunity_technical_validation.json")
    update_opportunity_request =
 helper.remove_nulls(update_opportunity_request_orig)

    try:
        # Perform an API call
        response =
 partner_central_client.update_opportunity(**update_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def update_opportunity_if_eligible(identifier):
    response = get_opportunity(identifier)
    if response is not None:
        return update_opportunity()
    else:
        print("Failed to retrieve opportunity details")

def usage_demo():
    identifier = "O5465588"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Updating opportunity.")
    print("-" * 88)

    helper.pretty_print_datetime(update_opportunity_if_eligible(identifier))
```

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

- For API details, see UpdateOpportunity in *AWS SDK for Python (Boto3) API Reference*.

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Scenarios for Partner Central using AWS SDKs

The following code examples show you how to implement common scenarios in Partner Central with AWS SDKs. These scenarios show you how to accomplish specific tasks by calling multiple functions within Partner Central or combined with other AWS services. Each scenario includes a link to the complete source code, where you can find instructions on how to set up and run the code.

Scenarios target an intermediate level of experience to help you understand service actions in context.

Examples

- Update associated entity of an opportunity

Update associated entity of an opportunity

The following code examples show how to:

- Disassociate an old entity.
- Associate a new entity.

Java

SDK for Java 2.x

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the Scenarios repository.

Update associated entity of an opportunity

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityReque
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRespo

/*
Purpose
PC-API -17 Replacing a solution
*/

public class ReplaceSolution {

 static PartnerCentralSellingClient client =
 PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";
     String originalEntityIdentifier = "S-0000000";
     String newEntityIdentifier = "S-0011111";
```

```
    disassociateOppornitityResponse(opportunityId, entityType,
originalEntityIdentifier );
    AssociateOpportunityResponse associateOpportunityResponse =
associateOpportunityResponse(opportunityId, entityType, newEntityIdentifier );

    ReferenceCodesUtils.formatOutput(associateOpportunityResponse);
   }

private static AssociateOpportunityResponse associateOpportunityResponse(String
opportunityId, String entityType, String entityIdentifier) {

       AssociateOpportunityRequest associateOpportunityRequest =
AssociateOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .opportunityIdentifier(opportunityId)
         .relatedEntityType(entityType)
         .relatedEntityIdentifier(entityIdentifier)
         .build();

       AssociateOpportunityResponse response =
client.associateOpportunity(associateOpportunityRequest);

       return response;
}

private static DisassociateOpportunityResponse
disassociateOppornitityResponse(String opportunityId, String entityType, String
entityIdentifier) {
 PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
           .region(Region.US_EAST_1)
           .credentialsProvider(DefaultCredentialsProvider.create())
           .httpClient(ApacheHttpClient.builder().build())
           .build();

       DisassociateOpportunityRequest disassociateOpportunityRequest =
DisassociateOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .opportunityIdentifier(opportunityId)
         .relatedEntityType(entityType)
         .relatedEntityIdentifier(entityIdentifier)
         .build();
```

```
        DisassociateOpportunityResponse response =
 client.disassociateOpportunity(disassociateOpportunityRequest);

        return response;
 }
}
```

- For API details, see the following topics in *AWS SDK for Java 2.x API Reference*.
  - AssociateOpportunity
  - DisassociateOpportunity

Python

SDK for Python (Boto3)

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

Update Associated Entity of an opportunity

```
#!/usr/bin/env python

"""
Purpose
PC-API -17 Replacing a solution
"""
import logging
import boto3
import utils.helpers as helper
from botocore.client import ClientError

from utils.constants import CATALOG_TO_USE

serviceName = "partnercentral-selling"

partner_central_client = boto3.client(
```

```
        service_name=serviceName,
        region_name='us-east-1'
)

def replace_solution(original_entity_identifier, new_entity_identifier,
 opportunityIdentifier):
    disassociate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : "Solutions",
        "RelatedEntityIdentifier" : original_entity_identifier
    }

    associate_opportunity_request ={
        "Catalog": CATALOG_TO_USE,
     "OpportunityIdentifier" : opportunityIdentifier,
        "RelatedEntityType" : "Solutions",
        "RelatedEntityIdentifier" : new_entity_identifier
    }
    try:
        # Perform an API call
        response =
 partner_central_client.disassociate_opportunity(**disassociate_opportunity_request)
        response =
 partner_central_client.associate_opportunity(**associate_opportunity_request)
        return response

    except ClientError as err:
        # Catch all client exceptions
        print(err.response)

def usage_demo():
    original_entity_identifier = "S-0049999"
    new_entity_identifier = "S-0050014"
    opportunityIdentifier = "O4397574"

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Replacing a solution.")
    print("-" * 88)

    helper.pretty_print_datetime(replace_solution(original_entity_identifier,
 new_entity_identifier, opportunityIdentifier))
```

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

- For API details, see the following topics in *AWS SDK for Python (Boto3) API Reference*.
  - AssociateOpportunity
  - DisassociateOpportunity

For a complete list of AWS SDK developer guides and code examples, see Using AWS Partner Central API with an AWS SDK. This topic also includes information about getting started and details about previous SDK versions.

Release notes

This page contains a summary of the significant changes to the documentation for the AWS Partner Central API, starting with the most recent update.

Revision history

Change	Description	Date
RC 6.0	• AWS Partner Central added new Partner Revenue Measurement APIs.	2026-06-30
RC 5.3	• ACE Resonate updates: `GetAwsOpportunitySummary` response updated.	2026-06-16
RC 5.2	• EUR currency support: `CreateOpportunity`, `UpdateOpportunity`, and `CreateEngagementInvitation` now accept EUR in addition to USD for the MRR currency code. When the AWS Partition is `aws-eusc` (AWS European Sovereign Cloud), the currency code must be EUR. • EUR in read operations: `GetOpportunity`, `ListOpportunities`, and `GetAwsOpportunitySummary` now return EUR in the `CurrencyCode` field for	2026-03-30

Change	Description	Date
	opportunities in the `aws-eusc` partition.	
RC 5.1	• EUR currency support: `CreateOpportunity`, `UpdateOpportunity`, and `CreateEngagementInvitation` now accept EUR in addition to USD for the MRR currency code. When the AWS Partition is `aws-eusc` (AWS European Sovereign Cloud), the currency code must be EUR. • EUR in read operations: `GetOpportunity`, `ListOpportunities`, and `GetAwsOpportunitySummary` now return EUR in the `CurrencyCode` field for opportunities in the `aws-eusc` partition.	2026-03-30

Change	Description	Date
RC 5	• Introduced Account API: Added a new Account API to AWS Partner Central, enabling partners to manage their account information, registrat ion, profiles, domain management, and partner connections. • Account API reference documentation: Added comprehensive API reference documenta tion for the Account API, including partner registrat ion, profile managemen t, domain verification, and account connection workflows. • Account API logging: Added logging support and documentation for the Account API to help audit and track API usage. • Account API notifications: Added event notification support for the Account API, enabling partners to receive real-time updates on account-related activitie s including partner account connections.	2025-11-30

Change	Description	Date
	• Account API quotas: Added service quotas documentation for the Account API. • Sandbox testing for Account API: Added sandbox testing capabilities for the Account API, allowing partners to test account management workflows in a non-production environment. • Introduced Benefits API: Added a new Benefits API to AWS Partner Central, enabling partners to discover, apply for, and manage benefits across AWS Partner Programs through a centralized interface. • Benefits API reference documentation: Added comprehensive API reference documentation for the Benefits API, covering benefit applications, allocations, and fulfillment processes. • Benefits API logging: Added logging support and documentation for the Benefits API to help audit and track API usage.	

Change	Description	Date
	• Benefits API notifications: Added event notification support for the Benefits API, enabling partners to receive real-time updates on benefit applications and allocations. • Benefits API quotas: Added service quotas documenta tion for the Benefits API. • Sandbox testing for Benefits API: Added sandbox testing capabilit ies for the Benefits API, allowing partners to test benefit management workflows in a non-produ ction environment. • Updated Selling API: Enhanced Selling API engagement actions to support lead management workflow. • Selling API documentation updates: Enhanced Selling API documentation with improved structure and updated workflows for lead management. • Selling API notifications updates: Enhanced event notification documenta tion for Engagement and	

Change	Description	Date
	Engagement Invitation events. • API structure expansion : Expanded documentation structure to support four distinct APIs (Account, Benefits, Selling, and Channel) with dedicated sections for each API's permissions, logging, notifications, quotas, and sandbox testing. • Enhanced API usage documentation: Added comprehensive usage guides for Account and Benefits APIs with detailed workflows and integration patterns. • Supported regions documentation: Added documentation for supported AWS regions for Account and Benefits APIs. • New filter for `ListOpportunities` : Added `TargetCloseDate` filter to enable filtering opportunities by their expected close date using `AfterTargetCloseDate` and `BeforeTargetCloseDate` fields in YYYY-MM-DD format.	

Change	Description	Date
RC 4	• Introduced Channel API: Added a new Channel API to AWS Partner Central, enabling partners to manage channel relations hips and partner-to-partner collaborations. • Channel API reference documentation: Added comprehensive API reference documenta tion for the Channel API, including all available actions and data models. • Channel API permissions: Added IAM permission policies and access control documentation specific to the Channel API. • Channel API logging: Added logging support and documentation for the Channel API to help audit and track API usage. • Channel API notifications: Added event notification support for the Channel API, enabling partners to receive real-time updates on channel-related activitie s.	2025-11-19

Change	Description	Date
	• Channel API quotas: Added service quotas documentation for the Channel API. • Sandbox testing for Channel API: Added sandbox testing capabilities for the Channel API, allowing partners to test channel workflows in a non-production environment. • API structure reorganization: Reorganized documentation to distinguish between the Selling API and Channel API, with dedicated sections for each API's permissions, logging, notifications, and quotas. • Supported regions documentation: Added documentation for supported AWS regions for both Selling and Channel APIs.	

Change	Description	Date
RC 3	• Updated API permission policies: Policies have been updated to reflect the new actions introduced in this release. Ensure your IAM roles and permissions are adjusted accordingly. • Custom header usage details: Added details on how to use the custom header `X-Amzn-User-Agent` to help audit and measure API usage. • Replaced actions with asynchronous actions: Replaced `AcceptOpportunityEngagementInvitation` with the asynchronous action `StartEngagementByAcceptingInvitationTask` and payload format has changed with respect to RC2. • Entity and action renaming: Changed `OpportunityEngagementInvitation` to `EngagementInvitation`. The attributes for these actions have also changed to align with the new entity. The following actions were replaced:	2024-10-17

Change	Description	Date
	• `GetOpportunityEngagementInvitation` is now `GetEngagementInvitation` . Multiple attributes in the invitation have been changed, added, or removed. • `ListOpportunityEngagementInvitations` is now `ListEngagementInvitations` . • `RejectOpportunityEngagementInvitation` is now `RejectEngagementInvitation` . • `AssignOpportunity` parameter update: `AssignOpportunity` now takes `Assignee` instead of `AssigneeEmail` . Update your implementations accordingly. • `SubmitOpportunity` functionality change: `SubmitOpportunity` is now performed by the asynchronous action `StartEngagementFromOpportunityTask` .	

Change	Description	Date
	`SubmitOpportunity` now accepts `InvolvementType` and `Visibility`. • Attribute renaming and expansion: `EstimatedAwsMonthlyRevenue` changed to `ExpectedCustomerSpend`. `ExpectedCustomerSpend` now includes new attributes, such as `Frequency` and `TargetCompany`. • New action for AWS opportunity summary: Opportunity visibility is now provided through a separate action called `GetAwsOpportunitySummary`. • Workflow updates for working with opportunities: The workflow for *Working with your opportunities* was updated to use the new actions introduced in this release. • Workflow updates for working with opportunities: The workflow for *Working with opportunities* was updated to use `EngagementInvitati`	

Change	Description	Date

	ons , which improves handling opportunities provided by AWS.
	• Workflow updates for tracking updates: The workflow to *Track Updates* is now *Working with opportunity updates*, enhancing clarity and efficiency in monitoring opportunity status.
	• New filters for listing actions: New filters are available for `ListSolutions` and `ListOpportunities` actions.
	  • For `ListSolutions` , the new filters are `FilterCategory` , `FilterIdentifier` , and `FilterStatus` .
	  • For `ListOpportunities` , the new filters are `FilterCustomerCompanyName` , `FilterIdentifier` , `FilterLastModifiedDate` , `FilterLifeCycleApprovalStatus` , `FilterLifeCycleStage` , and `FilterOrigin` .	

Change	Description	Date
	• Removal of `AccessControl` subobject: The `AccessControl` subobject in the opportunity model was removed. `NationalSecurity` is now a top-level attribute. • Attributes relocated to `GetAwsOpportunitySummary`: Attributes such as `AWSOpportunityTeam`, insights (`EngagementScore`, `NextBestAction`), origin, and `InvolvementType` are now available through `GetAwsOpportunitySummary` instead of `GetOpportunity`. • Introduction of `InvolvementType` field: Introduced the `InvolvementType` field to differentiate between cosell and visibility-only opportunities. • Event name updates: Events were updated with to reflect the changes. For example, *Opportunity engagement invitation created* is now *Engagement invitation created*.	

Change	Description	Date
	• Response payload updates: All response payloads now include an ID and/or ARN. • Request payload updates: All request payloads now use identifiers as input, which accept either an ID or ARN. • Engagement entity update: The title attribute in the engagement entity was renamed `EngagementTitle`. • Filter prefix removal: All filters no longer include the filter prefix, simplifying the filtering mechanism. • `StartEngagementFromOpportunity` action update: The attribute `AwsInvolvement` was changed to `AwsSubmission`. • Invitation project details update: `Invitation.ProjectDetails.TargetCompletionDate` was renamed `Invitation.ProjectDetails.EstimatedCompletionDate`. • Revenue estimate update: `Invitation.Partner`	

Change	Description	Date
	`RevenueEstimate` is now `ExpectedCustomerSpend` . The data type changed from one object to an array containing one object. • Receiver account update: The field `ReceiverAccount` was renamed `AccountReceiver` . • Engagement invitation new attribute: `SenderContact` attribute was introduced for engagement invitations. • `Customer.Contact` is now an array instead of an object. `OpportunityOwner` , `PartnerAccountManager` are supported values as `BusinessTitle` . • `PartnerOpportunityTeam` is now called `OpportunityTeam` . • `APNPrograms` is now `ApnPrograms` • Engagement invitation statuses were updated. • `ListEngagementInvitations` now requires `ParticipantType` as `RECEIVER`.	

Change	Description	Date
RC 2	• Introduced a new entity called OpportunityEngagementInvitation. • Introduced a new action: ListOpportunityEngagementInvitations. • Introduced a new action: GetOpportunityEngagementInvitation. • Introduced a new event: "Opportunity Engagement Invitation Created". • Replaced AcceptOpportunity with AcceptOpportunityEngagementInvitation. • Replaced RejectOpportunity with RejectOpportunityEngagementInvitation. • Replaced "Opportunity Accepted" with "Opportunity Engagement Invitation Accepted". • Replaced "Opportunity Rejected" with "Opportunity Engagement Invitation Rejected". • `PrimaryNeedsFromAWS` changed to PrimaryNeedsFromAws.	2024-08-07

Change	Description	Date
	• AWSOpportunityTeam changed to AwsOpportunityTeam . • AWSSalesLifeCycle changed to AwsSalesLifeCycle . • PrimaryNeedsFromAWSChangeReason changed to PrimaryNeedsFromAwsChangeReason . • AWSAccountId changed to AwsAccountId . • ExpectedMonthlyAWSRevenue changed to ExpectedMonthlyAwsRevenue . • AWSFundingUsed changed to AwsFundingUsed . • AWSMarketplaceOffers changed to AwsMarketplaceOffers . • Country changed to CountryCode . • PartnerOpportunityContact changed to PartnerOpportunityTeam . • Updated field Contact.Title to Contact.BusinessTitle .	

Change	Description	Date
	• Updated field `ContactInfo` to `OwnerInfo`. • Changed AWS Team from a map to a list. • Added details about the list of accepted Rejection Reasons. • Provided information on how to use Client Token. • Included details on how to use UpdateOpportunity. • Added guidance on using AWS Products and Partner Solutions. • Provided details about the OpportunityEngagementInvitation entity. • Updated StateOrRegion to include a revised list of accepted values. • Implemented multiple bug fixes to improve performance and error messaging.	

Change	Description	Date
RC 1	• [Breaking] Renaming of ApprovalStatus to ReviewStatus: We have renamed the ApprovalStatus field to ReviewStatus to better reflect its purpose. Additionally, we're introducing a new status, Pending Submission, to indicate a draft opportunity. The ReviewStatus will now accept the following values: Pending Submission, Submitted, In Review, Approved, Rejected, and Action Required. The former Draft status will be replaced by Pending Submission. • [Breaking] Introduction of SubmitOpportunity Action: A new action, SubmitOpportunity, has been introduced. Unlike the current behavior where CreateOpportunity also submits the opportunity, you can now create an opportunity in the Pending Submission state without linking a Solution or AWS Product immediately. To complete the submission, use the AssociateOpportuni	2024-06-21

Change	Description	Date
	ty action to link a solution or product, followed by SubmitOpportunity. This separation provides better API response performance and a flexible preparation phase before submission. • [Breaking] Requirement of a Catalog Parameter and Deprecation of Sandbox Endpoint: The 'gamma' endpoint (partnercentral-selling-gamma.us-eas t-1.amazonaws.com) will become unavailable after 7/30/2024. All API actions now require a Catalog parameter to specify whether the operation is performed on the Sandbox or AWS catalog. Notificat ion rules will now filter on catalog instead of environmentName. • [Breaking] Renaming of OpportunityIdentif ier to Identifier: Actions AssignOpportunity, AcceptOpportunity, and RejectOpportunity now take Identifier instead of OpportunityIdentifier to keep consistent with UpdateOpportunity.	

Change	Description	Date
	• Changes from type Enum to String for APNProgram CustomerUseCase, and UseCase: We will be converting Project.APNPrograms, Marketing.ActivityUseCases, and Project.CustomerUseCase from enumeration type to string type to reduce the risks associated with changing values. These fields will accept values only from predefined lists but not natively available in the SDKs. • Enhanced Error Handling: Error handling within the APIs has been significantly improved to facilitate easier debugging and quicker resolution of issues. The new error handling structures errors in two stages: 1. Request Validation Failed: Checks for data types and format, or 2. Business Validation Failed: All issues in a payload will be listed in one response, specifying the field that has errors. This consolidated feedback allows you to troubleshoot and correct errors more efficiently.	

Change	Description	Date
	Errors codes and formats have been standardized. • Client Source Identifier: Ability for partners include the X-Amzn-User-Agent header in their requests, which helps identify the source of the traffic. Use the format [Solution Name including solution provider] | [Version] For Example: AWS Partner CRM Connector | v2.0.1 • Bug fix, already deployed, no change to SDK] Opportunity Creation and AWS Account ID: Partners can now launch an opportunity while changing the AWS Account ID from null to a valid value without encounter ing errors. Previously, a bug prevented partners from performing these actions separately. • Bug fix, already deployed, no change to SDK] Solution Association with New Opportunities: Partners can associate a solution with a newly created opportuni ty. Before, when partners created a new opportuni	

Change	Description	Date
	ty and simultaneously associated a solution, the solution information was being lost. This issue has been resolved. • Bug fix, already deployed, no change to SDK] Opportunity Owner's Details Display: The issue where some opportuni ties were displaying the owner's details in a combined "LastName" field instead of the expected "Email", "FirstName", and "LastName" fields has been corrected. • Bug fix, already deployed, no change to SDK] Customer.Account.W ebsiteUrl Field Optional: The Customer.Account.W ebsiteUrl field has been made optional when AccessControls.Nat ionalSecurity field is "Yes" thus aligning the API behavior with the existing UI behavior. If the AccessControls.Nat ionalSecurity field is "No" or null, the WebsiteUrl field will be required as a business validation.	

Change	Description	Date
	• Bug fix, already deployed, no change to SDK] AWS Account ID Requirement for Consulting Partners: The inconsistency in the UI that showed the AWS Account ID field as optional for Consulting Partners has been resolved. It is now a conditional mandatory field, as per the documenta tion. • Bug fix, To be deployed on 6/19, no change to SDK] Stage of a Co-sell Opportunity to Closed Lost: Partners can now successfu lly update the Stage of a Co-sell Opportunity to Closed Lost, as long as a Closed Lost Reason is provided. This resolves the previous error stating "Missing Required Field: Closed Lost Reason is required when closing the opportunity." • Documentation] Improvements to documentation: Made several improvements to documentation to reflect the changes.	

Change	Description	Date

	• Known issues with a pending fix and list of upcoming features:
	  • While performing AssociateOpportunity action with Solutions or Products, the error messages from Offers are shown incorrectly.
	  • Ability to remotely create opportunities and simulate AWS validation process in Sandbox
	  • Postal Code returns null when it is of the format XXXXX-XXXX for US.
	  • Next Steps and Next Step History are not available for opportunities that are marked as "Do not need support from AWS."
	  • Unable to get historica l opportunities with Customer Business Problem descripti ons longer than 255 characters.
	  • Ability to delete opportunities that are pending submission.
	  • LeadSource is not available in the data model.	

Change	Description	Date
Beta 5	• Changed LifeCycle.ApprovalStatus to LifeCycle.ReviewStatus • Changed Insights.ReviewComments to LifeCycle.ReviewComments • Added LifeCycle.ReviewStatusReason (new attribute) • Added New Value "Pending" to PartnerAcceptance.Status • Released SDKs for dotnet, java, javascript, python, ruby. • Updated API Reference Guide to reflect the above changes.	2024-04-19

Change	Description	Date
Beta 4	• Added Sandbox Testing: Ability to test Opportuni ty Event notification in Sandbox. • Added Documentation: Introduced Opportunity Event notification sample rules. • Added SDK Releases: Released SDKs for Java (V2), Javascript (V2), DotNet (V3), and Python (V3). • Updated Date Format: Updated to follow ISO standards.	2024-03-04

Change	Description	Date
Beta 3	• Added Change Log File: Introduced a change log file for better tracking of updates and changes. • Updated API Reference Guide (PDF): Released a new version of the API Reference Guide in PDF format. • Updated API Reference Guide (Web Version): Updated the web version of the API Reference Guide. • Added DotNet SDK: Released the DotNet SDK, expanding our support for different programming environments. • Updated Python Boto 3 SDK (V2): Released Version 2 of the Python Boto 3 SDK, introducing new features and improveme nts.	2024-01-24
Beta 2	• Added API Reference Guide (Web Version): Released a new version of the API Reference Guide (Web Version). Note: This version requires a local webserver to be run after extraction for proper functionality.	2024-01-07

Change	Description	Date
Beta 1	• Added Boto3 (Initial Release): Launched the initial release of Boto3, a Python SDK for our services. • Added Beta Program Guide: Introduced a guide for participants in our Beta testing program. • Added API Reference Guide (PDF): Uploaded the first version of our API Reference Guide in PDF format, providing comprehensive documenta tion for our API.	2023-12-15
Beta 0	This initial release provides a suite of functionalities to manage and optimize partner engagements and opportuni ties in AWS Partner Central.	2023-11-15

Document history

The following is a list of documentation releases for this API reference.

Change	Description	Date
Added documentation for Partner Revenue Measurement APIs	AWS Partner central added new APIs for Partner Revenue Measurement (PRM).	June 30, 2026
Updated documentation for ACE Resonate	AWS Partner central updated API response for GetAwsOpportunitySummary action to support Cosell Motion.	June 16, 2026
Added documentation for Partner Central MCP Server	AWS Partner central agents MCP server provides partner central tools through the Model Context Protocol (MCP).	March 16, 2026
RC 5 release	Candidate 5 version of the AWS Partner Central API released. Introduced a new Account API to AWS Partner Central, enabling partners to manage their account information, registration, profiles, domain management, and partner connections. Introduced a new Benefits API to AWS Partner Central, enabling partners to discover, apply for, and manage benefits across AWS Partner Programs through a centralized interface. Enhanced Selling	November 30, 2025

	API engagement actions to support lead management workflow. Added comprehen sive usage guides and API documentation for Account, Benefits APIs with detailed workflows and integration patterns. Enhanced Selling API documentation for Engagement management with Lead context.	
RC 4 release	Candidate 4 version of the AWS Partner Central API released. Introduced Channel API for managing channel relationships and partner-to-partner collabora tions, enabling partners to qualify for program benefits when transacting with end customers using Billing Transfer. Documentation was re-organized to support multiple AWS Partner Central APIs.	November 19, 2025
Added documentation for multi-partner opportunities	Multi-partner opportunities (in preview) is an integrate d experience in AWS Partner Central that allows AWS Partners to find and connect with other Partners and co-sell together on customer deals.	December 4, 2024

RC 3 release	Candidate 3 version of the AWS Partner Central API released	October 17, 2024
RC 2 release	Candidate 2 version of the AWS Partner Central API released	August 7, 2024
RC 1 release	Candidate 1 version of the AWS Partner Central API released	June 21, 2024
Beta 5 release	Beta 5 of the AWS Partner Central API released	April 19, 2024
Beta 4 release	Beta 4 of the AWS Partner Central API released	March 4, 2024
Beta 3 release	Beta 3 of the AWS Partner Central API released	January 24, 2024
Beta 2 release	Beta 2 of the AWS Partner Central API released	January 7, 2024
Beta 1 release	Beta 1 of the AWS Partner Central API released	December 15, 2023
Beta 0 release	AWS Partner Central API initial release	November 15, 2023