User Guide

AWS for SAP MCP Server

AWS for SAP MCP Server: User Guide

Table of Contents

What is the AWS for SAP MCP Server?

With AWS for SAP MCP Server ("MCP Server"), running on Amazon Bedrock AgentCore ("AgentCore"), you can give your AI agents structured, protocol-driven access to SAP S/4HANA and SAP ECC OData (Open Data Protocol) V2 services through the Model Context Protocol (MCP). The MCP Server surfaces SAP operations as discoverable MCP tools, providing you with a secure, standardized interface for AI-driven SAP interactions.

Key capabilities

- **Service Discovery** — You can query the SAP API service catalog to find available OData services, with support for pagination and filtering by service type.
- **Metadata Inspection** — You can fetch OData service metadata to map entity types, properties, and relationships for contextual understanding.
- **OData Read Operations** — You can perform read queries against SAP OData entity sets, including filtering, field selection, and record count.
- **OData Write Operations** — You can create, update, and delete entity records when explicitly enabled by the operator.
- **Authentication** — You can configure inbound access control for connecting to the MCP Server and outbound authentication for SAP system access.
- **Function Import Execution** — You can invoke SAP OData function imports (consult SAP Documentation for feature specifics).
- **Custom Catalog Support** — You can augment or replace the default SAP service catalog by using a user-defined catalog stored in Amazon S3.
- **API Allowlisting** — You can use prefixes (or full API names) to allowlist one or multiple APIs to restrict the scope of MCP Server access.
- **Service Hints** — You can provide targeted usage guidance for SAP OData services through an Amazon S3-hosted hints file, helping agents optimize interactions.

Architecture

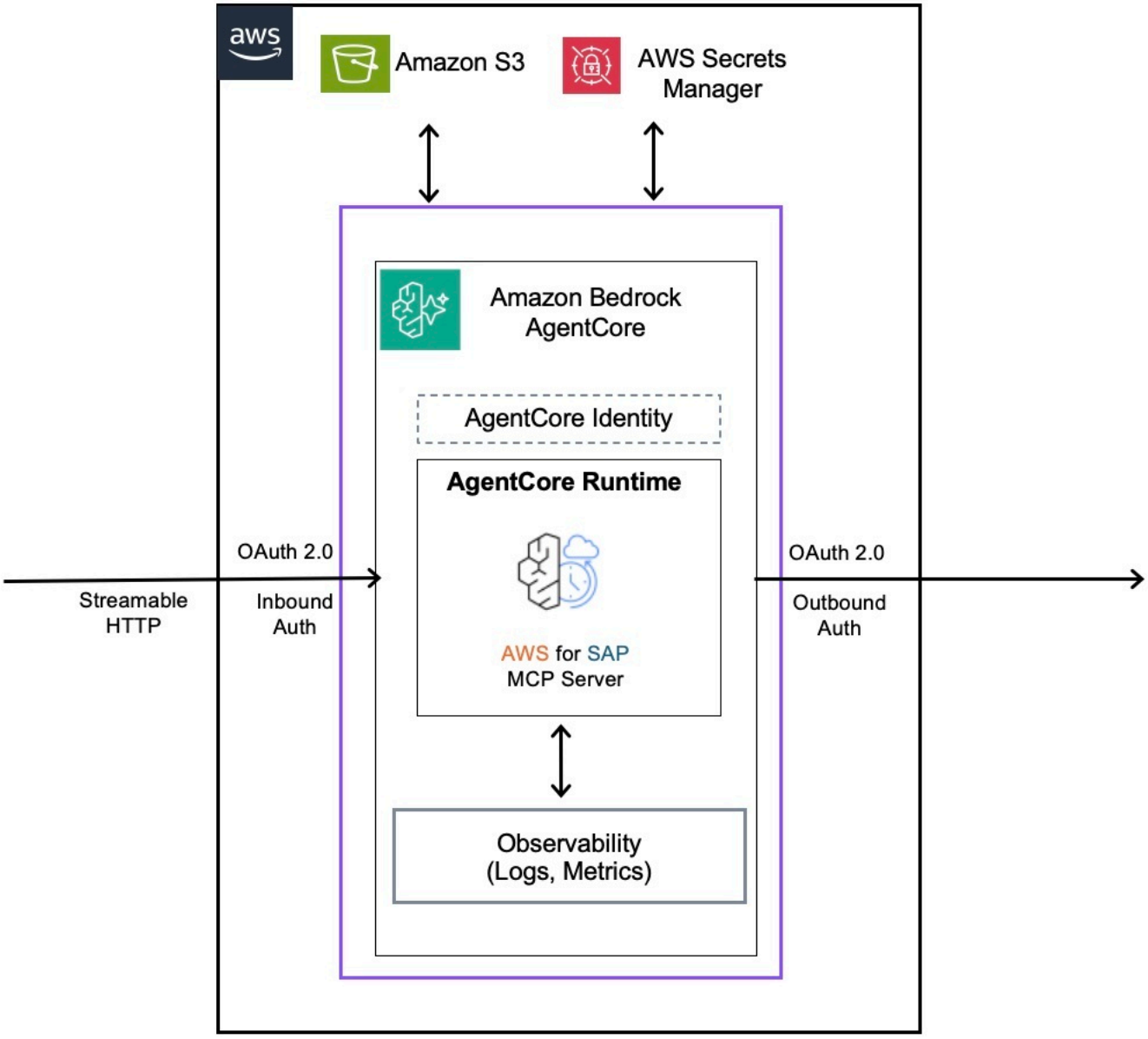
With the MCP Server, your AI agents (Clients) communicate through Streamable HTTP on the dedicated `/mcp` endpoint. Each interaction uses a stateless session. The server container is

available through Amazon Elastic Container Registry (Amazon ECR) and is deployed in Amazon Bedrock AgentCore Runtime.

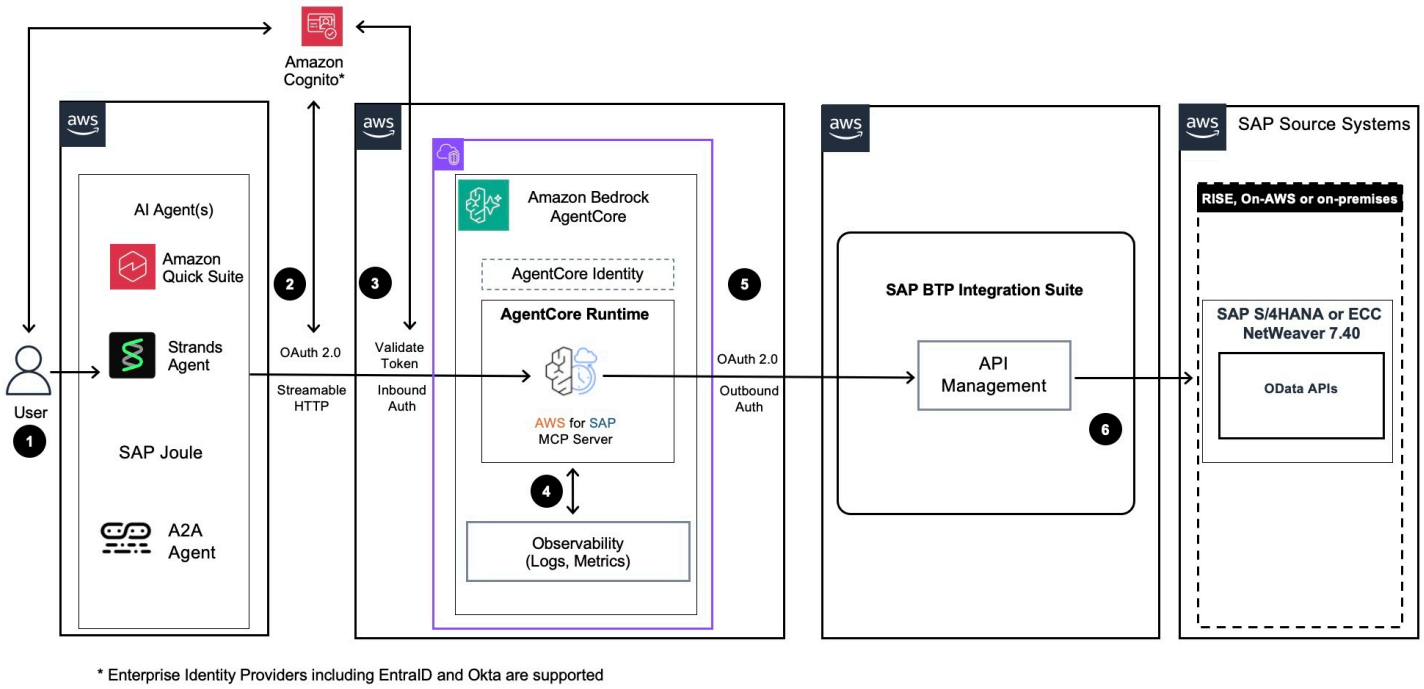
Through this Runtime, the MCP Server securely connects to SAP systems to run operations such as service discovery, metadata retrieval, CRUD (Create, Read, Update, Delete) actions, and function imports. The MCP Server integrates with the following AWS services:

- Amazon Bedrock AgentCore Identity — For authenticated access through OAuth token flows.
- AWS Secrets Manager — For secure credential storage.
- Amazon S3 — To host custom catalogs and service hints files.

The following diagram shows the AWS for SAP MCP Server architecture:



The following is a sample architecture pattern showing how you can use the AWS for SAP MCP Server with SAP Business Technology Platform (SAP BTP) API Management. We recommend enabling your SAP OData APIs through the SAP BTP API Management layer. The AWS for SAP MCP Server connects to those APIs securely over HTTPS with OAuth 2.0 authentication. Data transmitted to these API endpoints is encrypted in transit by using TLS.



The following steps describe the flow:

1. **User initiation** — A business user interacts with an MCP-compatible AI agent (for example, Amazon Quick).
2. **Authentication initiation** — The AI agent initiates an authentication request to the AWS for SAP MCP Server, which runs on Amazon Bedrock AgentCore Runtime. The agent communicates over Streamable HTTP to discover and invoke available tools on behalf of the user.
3. **Inbound authentication** — The inbound authorizer within Amazon Bedrock AgentCore Identity validates the token issued by the configured identity provider before allowing access to the AWS for SAP MCP Server.
4. **Observability** — You can monitor tool invocations and server activity through the logs and metrics that AgentCore Runtime emits to Amazon CloudWatch.
5. **Outbound authentication** — The AWS for SAP MCP Server uses outbound authentication to securely connect to SAP BTP Integration Suite API Management through OAuth 2.0.
6. **SAP backend data access** — The SAP BTP Destination Service routes the request to SAP S/4HANA or SAP ECC to retrieve data through OData APIs.

Supported AWS Regions

The AWS for SAP Model Context Protocol (MCP) Server runs on Amazon Bedrock AgentCore Runtime. You can deploy the MCP Server in any AWS Region where AgentCore Runtime is available and where the AWS for SAP MCP Server container image is published.

Region availability

The following table shows the AWS Regions where the AWS for SAP MCP Server is supported.

AWS Region name	Region code	Support
US East (N. Virginia)	`us-east-1`	✓
US East (Ohio)	`us-east-2`	✓
US West (Oregon)	`us-west-2`	✓
Europe (Frankfurt)	`eu-central-1`	✓
Europe (Ireland)	`eu-west-1`	✓
Europe (London)	`eu-west-2`	✓
Europe (Paris)	`eu-west-3`	✓
Europe (Stockholm)	`eu-north-1`	✓
Asia Pacific (Mumbai)	`ap-south-1`	✓
Asia Pacific (Singapore)	`ap-southeast-1`	✓
Asia Pacific (Sydney)	`ap-southeast-2`	✓
Asia Pacific (Tokyo)	`ap-northeast-1`	✓
Asia Pacific (Seoul)	`ap-northeast-2`	✓
Canada (Central)	`ca-central-1`	✓

> **Note**
>
> The AWS for SAP MCP Server requires Amazon Bedrock AgentCore Runtime and Amazon Bedrock AgentCore Identity. Verify that both services are available in your target Region before deployment. For the latest AgentCore region availability, see Supported AWS Regions in the Amazon Bedrock AgentCore documentation.

Region considerations

When you choose a Region for your AWS for SAP MCP Server deployment, consider the following:

- **Proximity to your SAP system** — Deploy the MCP Server in the same Region (or closest Region) to your SAP system to minimize network latency for OData (Open Data Protocol) requests.
- **Data residency requirements** — Choose a Region that meets your organization's data residency and compliance requirements for SAP data processing.
- **Dependent AWS services** — The MCP Server uses AWS Secrets Manager, Amazon S3, and Amazon Bedrock AgentCore Identity. These services must be available in your chosen Region.
- **Network connectivity** — Ensure that the VPC and subnets in your chosen Region have network connectivity to your SAP system, whether through AWS Direct Connect, VPN, or internal routing.

Available MCP tools

With the AWS for SAP Model Context Protocol (MCP) Server, your AI agents can access tools and prompts for interacting with SAP systems. These tools are split into two categories: Read tools, which are available by default, and Write tools, which you must explicitly enable before they can be used.

Tools reference

Tool	Description	Classification
`find_sap_services`	Discovers available SAP OData services from the standard or custom service catalog. Supports pagination (`top`/`skip`).	Read
`get_metadata`	Retrieves OData service metadata.	Read
`odata_read`	Reads data from SAP OData entity sets.	Read
`odata_count`	Returns the count of records in an OData entity set, with optional filtering. Use this before `odata_read` to understand data volume.	Read
`odata_create`	Creates a new entity record in an SAP OData entity set. Requires a JSON payload with entity data.	Write
`odata_update`	Updates an existing entity record identified by key fields.	Write
`odata_delete`	Deletes an entity record identified by key fields.	Write
`odata_function_import`	Enables execution of custom backend logic that does not fit standard CRUD operations.	Write

Tool	Description	Classification
`get_service_hints`	Returns usage guidance and hints for a specific SAP OData service from the configured service hints file.	Read

Important

By default, the AWS for SAP MCP Server runs in **read-only mode**. Your AI agents can perform non-mutating tasks such as querying service catalogs, inspecting metadata, and running read operations. Write actions — including create, read, update, and delete (CRUD) operations and function imports — are disabled unless you set both the global `WriteEnabled` configuration and the operation specific flag to `true`.

Identity and Authentication

The AWS for SAP MCP Server uses two layers of authentication: inbound authentication controls access to the Model Context Protocol (MCP) Server, and outbound authentication controls access to SAP.

Inbound Authentication

Inbound authentication determines which clients are allowed to invoke the MCP server. The AgentCore runtime validates incoming JSON Web Tokens (JWT) before requests reach the server. Only requests with a valid token from a trusted identity provider are accepted.

The CloudFormation template supports two inbound authentication options:

1. **Amazon Cognito**

   When Amazon Cognito is selected as the inbound authentication provider, the CloudFormation template automatically creates and configures a Cognito user pool, client, resource server, and domain. MCP Clients authenticate by requesting a token from the Cognito token endpoint using client credentials. The Cognito user pool client ID, client secret, and token endpoint are available in the CloudFormation stack outputs after deployment.

2. **External Identity Provider**

   When an external identity provider such as Microsoft Entra ID is selected, the CloudFormation template configures the AgentCore runtime to validate tokens issued by that provider.

Outbound Authentication

You can choose from three authentication flows to connect to SAP systems. You select the authentication flow through the CloudFormation template.

Choosing your outbound authentication setup

Scenario	Auth Flow	Protocol	Identity Provider	SAP Configuration
Direct SAP credentials, no IdP	BASIC	Basic Auth	None	SAP System User
SAP as Authorization Server with OAuth2	M2M	OAuth2	SAP	SAP OAuth2 Client
External IdP with OIDC	M2M	OIDC	Entra ID	SAP OIDC trust
SAP as Authorization Server with OAuth2 and SAML IdP redirect	User Federation	OAuth2 + SAML	Entra ID or other SAML IdP	SAP OAuth2 client + SAML trusted provider
External IdP with OIDC	User Federation	OIDC	Entra ID	SAP OIDC trust

Basic Authentication

The Basic authentication flow retrieves SAP username and password credentials from AWS Secrets Manager at runtime. This is the simplest flow, suitable for development, testing, and environments where direct SAP credentials are acceptable.

> **Warning**
>
> Not recommended for production deployments.

How it works:

1. At startup, the server validates that the secret specified in the CloudFormation template exists in AWS Secrets Manager. If the secret is not found, the server fails to start.
2. On each SAP OData request, the server calls AWS Secrets Manager to retrieve the username and password from the secret.

Key characteristics:

- Credentials are retrieved from AWS Secrets Manager per request - they are never persisted in AgentCore Runtime.
- No user interaction is required.
- No dependency on Bedrock AgentCore Identity.

Machine to Machine (2-Legged OAuth)

The machine-to-machine (M2M) authentication flow uses Bedrock AgentCore Identity to exchange credentials for an SAP OData (Open Data Protocol) OAuth token without any user interaction. This flow is designed for automated and headless deployments. M2M uses either OAuth2 or OIDC depending on your choice of IdP.

How it works:

1. At startup, the server validates the OAuth provider specified by the CloudFormation template against Bedrock AgentCore Identity. If the provider does not exist, the server fails to start.
2. Bedrock AgentCore Identity performs the OAuth token exchange and returns an access token.

Key characteristics:

- No user interaction required - fully automated token exchange.
- Token is never persisted in MCP Server.
- Requires a pre-configured OAuth provider in Bedrock AgentCore Identity.

User Federation (3-Legged OAuth)

User Federation authentication flow uses Bedrock AgentCore Identity with a callback URL to perform interactive OAuth token exchange. This flow is designed for scenarios where user-specific

access is required, and the user must authorize access via a browser. This flow uses either OAuth2, OIDC, or SAML2 depending on your choice of IdP.

How it works:

1. At startup, the server validates the OAuth provider against Bedrock AgentCore Identity. If the provider does not exist, the server fails to start.
2. Bedrock AgentCore Identity returns an authorization URL, which the server passes back to the MCP client for the user to open in a browser.
3. The user authorizes access in the browser, which redirects to the AgentCore callback URL.
4. AgentCore then redirects the user's browser to the client application callback URL to signal the flow is complete.
5. After authorization completes, Bedrock AgentCore Identity issues an access token.

> **Note**
>
> This flow uses two callback URLs: * The AgentCore callback URL (auto-generated, must be registered with your IdP). * The client application callback URL (configured via CloudFormation template).

Getting Started

Before you deploy the AWS for SAP Model Context Protocol (MCP) Server by using the CloudFormation template, verify that you meet the following prerequisites across your AWS environment, SAP system, and network.

AWS Prerequisites

- An AWS account with access to AWS Bedrock AgentCore for hosting the AWS for SAP MCP Server.
- You (or the IAM role you use for deployment) must have the following IAM permissions.

IAM permissions required for CloudFormation deployment

The AWS for SAP MCP Server can be deployed via a CloudFormation template (see *Deployment*). The following IAM permissions are required for the user or role deploying the CloudFormation template. These permissions allow CloudFormation to create and manage the resources defined in the template.

AWS Service	Actions	Resources
AWS CloudFormation	`cloudformation:CreateStack ,cloudformation:UpdateStack ,cloudformation:DeleteStack ,cloudformation:DescribeStacks ,cloudformation:DescribeStackEvents ,cloudformation:GetTemplate ,cloudformation:CreateChangeSet`	`arn:aws:cloudformation:*:*:stack/*`
AWS Secrets Manager	`secretsmanager:GetSecretValue ,secretsmanager:DescribeSecr`	`arn:aws:secretsmanager:*:*:secret:<your-secret-name>*`

AWS Service	Actions	Resources
	et ,secretsmanager:CreateSecret ,secretsmanager:DeleteSecret	
Amazon Cognito	cognito-idp:CreateUserPool ,cognito-idp:CreateUserPoolClient ,cognito-idp:CreateUserPoolDomain ,cognito-idp:CreateResourceServer ,cognito-idp:DeleteUserPool ,cognito-idp:DeleteUserPoolClient ,cognito-idp:DeleteUserPoolDomain ,cognito-idp:DeleteResourceServer ,cognito-idp:DescribeUserPool ,cognito-idp:DescribeUserPoolClient ,cognito-idp:DescribeResourceServer ,cognito-idp:DescribeUserPoolDomain	*
AWS IAM	iam:CreateRole ,iam:DeleteRole ,iam:GetRole ,iam:PutRolePolicy ,iam:DeleteRolePolicy ,iam:AttachRolePolicy ,iam:DetachRolePolicy ,iam:PassRole ,iam:ListRolePolicies ,iam:ListAttachedRolePolicies ,iam:GetRolePolicy	arn:aws:iam::*:role/*

AWS Service	Actions	Resources
AWS Lambda	`lambda:CreateFunction ,lambda:DeleteFunction ,lambda:GetFunction ,lambda:GetFunctionConfiguration ,lambda:InvokeFunction ,lambda:UpdateFunctionCode ,lambda:UpdateFunctionConfiguration ,lambda:AddPermission ,lambda:RemovePermission`	`arn:aws:lambda:*:*:function:*`
Amazon Bedrock AgentCore	`bedrock-agentcore:CreateRuntime ,bedrock-agentcore:UpdateRuntime ,bedrock-agentcore:DeleteRuntime ,bedrock-agentcore:GetRuntime ,bedrock-agentcore:CreateOauth2CredentialProvider ,bedrock-agentcore:GetOauth2CredentialProvider ,bedrock-agentcore:DeleteOauth2CredentialProvider ,bedrock-agentcore:CreateTokenVault`	*
Amazon S3	`s3:GetObject`	CloudFormation template S3 URI
Amazon ECR	`ecr:BatchGetImage ,ecr:GetDownloadUrlForLayer ,ecr:GetAuthorizationToken`	*

AWS Service	Actions	Resources
Amazon CloudWatch Logs	`logs:CreateLogGroup` , `logs:DescribeLogGroups` , `logs:DescribeLogStreams` , `logs:CreateLogStream` , `logs:PutLogEvents`	`arn:aws:logs:*:*:log-group:/aws/bedrock-agentcore/runtimes/*`
Amazon CloudWatch	`cloudwatch:PutMetricData`	*
AWS X-Ray	`xray:PutTraceSegments` , `xray:PutTelemetryRecords` , `xray:GetSamplingRules` , `xray:GetSamplingTargets`	*

SAP Prerequisites

- SAP S/4HANA or SAP ERP Central Component (ECC) system with OData (Open Data Protocol) enabled. SAP Gateway supports OData V2 from SAP Application Server ABAP (AS ABAP) 7.00. For more information, see SAP documentation.
- SAP OData Service Activation via OData Service in SAP Gateway Hub.
- Enable the OData API to use service type `WEB_API`. This setting is recommended and applies only to SAP S/4HANA.
- SAP's OData Service Catalog `IWFND/CATALOGSERVICE;v=2` must be available on the SAP system for service discovery. This is required only for the use of Standard Catalog.
- Valid SAP credentials (System User / OAuth) for your chosen authentication flow.

Network Prerequisites

- Allow outbound HTTPS access from the Amazon Bedrock AgentCore Runtime elastic network interface (ENI) to the SAP system. If traffic flows through an Application Load Balancer (ALB) or Network Load Balancer (NLB), allow outbound HTTPS access to that ALB or NLB instead.
- Allow outbound HTTPS access from Amazon Bedrock AgentCore Runtime to AWS services such as AWS Secrets Manager, Amazon S3, and Amazon Bedrock AgentCore Identity. Because these

service endpoints are public, the private subnet where Amazon Bedrock AgentCore Runtime runs must provide internet access through a NAT Gateway. As an alternative, you can use VPC endpoints to access these services privately.

- Inbound connectivity on HTTPS (443) port from the MCP client (AI agent) to Amazon Bedrock AgentCore Runtime.
- For detailed prerequisites, refer to the Amazon Bedrock AgentCore documentation:
  - AgentCore Runtime
  - AgentCore Identity
  - AgentCore VPC configuration

Authentication Prerequisites

The prerequisites for credential setup depend on which authentication flow you plan to use.

Inbound authentication

Inbound authentication requires a JSON Web Token (JWT) compatible identity provider. The CloudFormation template supports two options:

- **Amazon Cognito:** No prerequisites.
- **External Identity Provider (Entra ID):** If you use an external identity provider (IdP) for inbound authentication, you must configure your IdP before deployment and provide the following through the CloudFormation template parameters:
  - **Discovery URL:** Your identity provider's well-known configuration endpoint, used by AgentCore Identity to fetch token validation keys and issuer information.
  - **Allowed audiences:** The audience values that AgentCore Identity accepts when validating incoming tokens.

Outbound Authentication

AWS for SAP MCP Server supports Basic Auth and OAuth2 (M2M and User Federation).

Basic Authentication

AWS Secrets Manager secret containing SAP username and password is required in the following Key-Value Pair format.

```
username: <sap_username>
password: <sap_password>
```

OAuth 2.0 Authentication

AWS Secrets Manager secret containing IdP clientId and clientSecret is required in the following Key-Value Pair format, for both M2M and User Federation.

```
clientId: <oauth_client_id>
clientSecret: <oauth_client_secret>
```

Additionally, SAP and IdP related OAuth2/OIDC/SAML configuration are required as outlined for different patterns.

Machine to Machine (2-Legged OAuth)

Pattern 1: SAP as Authorization Server with OAuth2

- • SAP Application Server (Gateway), when hosted inside your VPC without public internet exposure will require AgentCore Identity to have private connectivity in order to reach the IdP's token/Auth endpoints. In these cases, configure a Private Endpoint on the outbound OAuth credential provider (SAP in this case) using VPC Lattice (managed or self-managed). See Connect to private identity providers for the required VPC, subnet, security group, and IAM setup.
- OAuth Client Setup via OAuth client configuration.
- Scopes correspond to SAP OData service names (for example, `ZAPI_SALES_ORDER_SRV_0001`).

Pattern 2: External IdP (Entra ID) with OIDC

- Your identity provider must be configured and accessible before deployment. You will need the authorization URL, token URL, and scope from your IdP for the CloudFormation template parameters.
- OIDC Trust Setup via SOIDC configuration.
- This is applicable only to SAP S/4HANA.

User Federation (3-Legged OAuth)

Pattern 1: SAP as Authorization Server with OAuth2

- Same as M2M Pattern 1 prerequisites, plus:
  - The AgentCore callback URL must be registered with SAP as a redirect URI in the OAuth2 client configuration. This URL is auto-generated during deployment.

Pattern 2: SAP as Authorization Server with OAuth2 + SAML IdP (Entra ID)

- Same as M2M Pattern 1 prerequisites, plus:
  - A SAML identity provider (Entra ID) configured as a trusted provider in SAP (SAML2 transaction).
  - The AgentCore callback URL must be registered with SAP as a redirect URI in the OAuth2 client configuration. This URL is auto-generated during deployment.

Pattern 3: External IdP (Entra ID) with OIDC

- Same as External IdP M2M OIDC prerequisites, plus:
  - The AgentCore callback URL must be registered as a redirect URI in your identity provider's application configuration. This URL is auto-generated during deployment.
  - This is applicable only to SAP S/4HANA.

Deployment via AWS CloudFormation

You deploy the AWS for SAP Model Context Protocol (MCP) Server onto Amazon Bedrock AgentCore Runtime by using an AWS CloudFormation template. The template creates all required resources for you, including the AgentCore runtime, Identity and Access Management (IAM) roles, and networking configuration.

Before you begin, ensure you have completed the necessary setup detailed in the *Getting Started* section.

Summary of prerequisites

- **IAM Roles** with sufficient privileges to create CloudFormation stacks and the resources defined within the stack.
- **VPC** with private subnets and security groups already configured in your target AWS Region.
- **Network connectivity** from selected subnets to your SAP system (via Direct Connect, VPN, or internal routing).
- **AWS Secrets Manager secret** containing SAP or OAuth credentials.
- **Private IdP** if your VPC-hosted SAP system (Application Server/Gateway) is used as an IdP, you must choose an Amazon VPC Lattice connectivity mode (managed or self-managed). For self-managed mode, pre-create a VPC Lattice resource configuration before deployment. For more information, see Connect to private identity providers.

Deployment Steps

Step 1: Launch the CloudFormation stack

1. Sign in to the {aws-management-console}.
2. Navigate to **CloudFormation** → **Stacks** → **Create stack** → **With new resources (standard)**.
3. Under **Specify template**, select **Amazon S3 URL** and enter:

```
https://awsforsap-mcp-server-setup-{region}.s3.{region}.amazonaws.com/cfn-launch-template/latest/AwsForSapMcpServerStack.template.json
```

Example:

```
https://awsforsap-mcp-server-setup-us-east-1.s3.us-east-1.amazonaws.com/cfn-launch-template/latest/AwsForSapMcpServerStack.template.json
```

4. Choose **Next**.

Step 2: Specify stack details

Provide a **Stack name** following these rules:

- Must start with a letter (a–z, A–Z).
- Can contain letters, numbers, and hyphens only.
- Maximum 128 characters.

Example: `aws-for-sap-mcp-server`

Step 3: Configure Parameters

General configuration

Parameter	Parameter Label	Description	Example
`UniqueId`	Unique Identifier	Short unique identifier for stack resources. Max 8 characters, lowercase alphanumeric. Used to namespace all resources within the stack.	`mcp01`

SAP system configuration

Parameter	Parameter Label	Description	Example
`SapBaseUrl`	SAP Base OData Endpoint	Base URL of the SAP OData endpoint.	`https://host:port/`

Parameter	Parameter Label	Description	Example
			`sap/opu/odata/sap/`

Authentication configuration

The MCP Server supports three authentication flows for connecting to SAP. Choose the one that matches your SAP system setup.

Parameter	Parameter Label	Description	BASIC	M2M	USER_FEDERATION
`AuthFlow`	Authentication Flow	Authentication flow to use: `BASIC`, `M2M`, or `USER_FEDERATION` .	✓	✓	✓
`SapCredentialsSecret`	Auth Credentials Secret Name	AWS Secrets Manager secret name containing SAP credentials (BASIC) or OAuth client credentials (M2M).	Required	Required	Required
`SapAuthorizeUrl`	Authorization Endpoint	SAP OAuth2 authorization URL.	—	Required	Required
`SapTokenUrl`	Token Endpoint	SAP OAuth2 token URL.	—	Required	Required
`OauthScopes`	Scope(s)	OAuth scopes for SAP access.	—	Required	Required

Inbound Authentication Configuration

Parameter	Parameter Label	Description	Notes
`InboundAuthProvider`	Inbound Authentication Provider	Identity provider for validating incoming requests to the MCP Server.	Default: Cognito (or Entra ID)
`DiscoveryUrl`	Discovery Url	URL to fetch authorization server metadata for JWT validation.	Required for Entra ID; not needed for Cognito
`AllowedAudiences`	Allowed Audiences	Audience values validated in incoming JWT tokens.	Required for Entra ID; not needed for Cognito

Note

`DiscoveryUrl` and `AllowedAudiences` are only required when using an external identity provider (for example, Entra ID). Leave these fields empty if using Cognito.

MCP Server Configuration

Parameter	Parameter Label	Description	Allowed Values	Default
`McpServerLogLevel`	MCP Server Log Level	Server log level.	`DEBUG`, `INFO`, `WARNING`, `ERROR`	`INFO`

MCP Server Permissions

Control which operations the MCP Server is permitted to perform against your SAP system. Start with the minimum permissions required.

Parameter	Parameter Label	Default	Dependency
McpServerReadEnabled	Enable Read Access	TRUE	—
McpServerWriteEnabled	Enable Write Access	FALSE	—
McpServerCreateEnabled	Enable Create Access	FALSE	Requires McpServerWriteEnabled=true
McpServerUpdateEnabled	Enable Update Access	FALSE	Requires McpServerWriteEnabled=true
McpServerDeleteEnabled	Enable Delete Access	FALSE	Requires McpServerWriteEnabled=true
McpServerFunctionImportEnabled	Enable Function Import	FALSE	—

Note

McpServerCreateEnabled, McpServerUpdateEnabled, and McpServerDeleteEnabled have no effect unless McpServerWriteEnabled is set to true.

Network Configuration

Parameter	Parameter Label	Description
`McpServer VpcSecurityGroup`	VPC Security Groups	Comma-separated list of VPC security group IDs. The security group must allow outbound traffic to your SAP system on the `SapBaseUrl` port. If traffic is routed through a load balancer (ALB/NLB), allow traffic on the associated listener port instead.
`McpServer NetworkSubnets`	VPC Subnets	Comma-separated list of private subnet IDs where AgentCore launches its ENIs. These subnets must have network connectivity to the SAP system.

Subnet selection guidance:

- The subnets for the AWS for SAP MCP Server must use an Availability Zone that is supported by AgentCore. See AgentCore supported Availability Zones.
- If your SAP system runs in a multi-AZ high-availability setup, ensure the network route from the selected subnets to the SAP virtual IP always resolves to the active system.
- If traffic is routed through an Elastic Load Balancer (ALB/NLB), ensure the subnets can reach the ELB's resolved IPs.

Step 4: Configure stack options

On the **Configure stack options** page you can optionally add tags, set IAM permissions, and configure stack failure behavior. For most deployments, the defaults are sufficient.

Choose **Next**.

Step 5: Review and Deploy

1. Review all parameters on the summary page.
2. If prompted, acknowledge the IAM capabilities checkbox: **"I acknowledge that AWS CloudFormation might create IAM resources"**.
3. Choose **Submit**.

Step 6: Monitor Deployment

1. Navigate to **CloudFormation** → **Stacks** and select your stack.
2. Open the **Events** tab to monitor progress.
3. When the status shows `CREATE_COMPLETE`, the MCP Server is deployed and ready.

> **Note**
>
> If the stack reaches `ROLLBACK_COMPLETE`, check the Events tab for the root cause error, correct the parameter values, and redeploy.

Example: Basic Authentication Deployment

The following command deploys the AWS for SAP MCP Server with Basic Authentication. Replace the placeholder values with your actual configuration:

```
aws cloudformation create-stack \
  --stack-name <your-stack-name> \
  --template-urlhttps://awsforsap-mcp-server-setup-<region>.s3.<region>.amazonaws.com/
cfn-launch-template/latest/AwsForSapMcpServerStack.template.json \
  --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM \
  --parameters \
    ParameterKey=UniqueId,ParameterValue=<your-unique-id> \
    ParameterKey=SapBaseUrl,ParameterValue=<your-sap-base-url> \
    ParameterKey=AuthFlow,ParameterValue=BASIC \
    ParameterKey=SapCredentialsSecret,ParameterValue=<your-secret-name> \
    ParameterKey=McpServerLogLevel,ParameterValue=INFO \
    ParameterKey=McpServerReadEnabled,ParameterValue=true \
    ParameterKey=McpServerWriteEnabled,ParameterValue=false \
    ParameterKey=McpServerCreateEnabled,ParameterValue=false \
    ParameterKey=McpServerUpdateEnabled,ParameterValue=false \
    ParameterKey=McpServerDeleteEnabled,ParameterValue=false \
    ParameterKey=McpServerVpcSecurityGroup,ParameterValue="sg-1234567" \
    ParameterKey=McpServerNetworkSubnets,ParameterValue="subnet-1234567"
```

Note

For Basic Authentication with Cognito as the inbound provider, `DiscoveryUrl` and `AllowedAudiences` are not required.

Example: Machine-to-Machine (M2M) Authentication Deployment

For machine-to-machine OAuth authentication, include the additional OAuth parameters:

```
aws cloudformation create-stack \
  --stack-name <your-stack-name> \
  --template-url https://awsforsap-mcp-server-setup-<region>.s3.<region>.amazonaws.com/
cfn-launch-template/latest/AwsForSapMcpServerStack.template.json \
  --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM \
  --parameters \
    ParameterKey=UniqueId,ParameterValue=<your-unique-id> \
    ParameterKey=SapBaseUrl,ParameterValue=<your-sap-base-url> \
    ParameterKey=SapSystemType,ParameterValue=S4HANA \
    ParameterKey=SapClientNumber,ParameterValue=<your-client-number> \
    ParameterKey=InboundAuthProvider,ParameterValue=<your-idp-provider> \
    ParameterKey=DiscoveryUrl,ParameterValue=<your-discovery-url> \
    ParameterKey=AllowedAudiences,ParameterValue=<your-allowed-audiences> \
    ParameterKey=AuthFlow,ParameterValue=M2M \
    ParameterKey=SapCredentialsSecret,ParameterValue=<your-secret-name> \
    ParameterKey=SapAuthorizeUrl,ParameterValue=<your-sap-authorize-url> \
    ParameterKey=SapTokenUrl,ParameterValue=<your-sap-token-url> \
    ParameterKey=OauthScopes,ParameterValue=<your-oauth-scopes> \
    ParameterKey=McpServerLogLevel,ParameterValue=INFO \
    ParameterKey=McpServerReadEnabled,ParameterValue=true \
    ParameterKey=McpServerWriteEnabled,ParameterValue=false \
    ParameterKey=McpServerCreateEnabled,ParameterValue=false \
    ParameterKey=McpServerUpdateEnabled,ParameterValue=false \
    ParameterKey=McpServerDeleteEnabled,ParameterValue=false \
    ParameterKey=McpServerFunctionImportEnabled,ParameterValue=false \
    ParameterKey=UseSapCatalog,ParameterValue=true \
    ParameterKey=McpServerCustomCatalogBucketUri,ParameterValue=None \
    ParameterKey=McpServerServiceHintsS3Uri,ParameterValue=None \
    ParameterKey=AllowedServicePrefixes,ParameterValue=None \
    ParameterKey=McpServerVpcSecurityGroup,ParameterValue=<your-security-group-id> \
```

```
    ParameterKey=McpServerNetworkSubnets,ParameterValue=<your-subnet-ids>
```

> **Note**
>
> `DiscoveryUrl` and `AllowedAudiences` are required when using any inbound auth provider other than Cognito.

Example: User Federation Authentication Deployment

```
aws cloudformation create-stack \
  --stack-name <your-stack-name> \
  --template-url https://awsforsap-mcp-server-setup-<region>.s3.<region>.amazonaws.com/
cfn-launch-template/latest/AwsForSapMcpServerStack.template.json \
  --capabilities CAPABILITY_IAM CAPABILITY_NAMED_IAM \
  --parameters \
    ParameterKey=UniqueId,ParameterValue=<your-unique-id> \
    ParameterKey=SapBaseUrl,ParameterValue=<your-sap-base-url> \
    ParameterKey=SapSystemType,ParameterValue=S4HANA \
    ParameterKey=SapClientNumber,ParameterValue=<your-client-number> \
    ParameterKey=InboundAuthProvider,ParameterValue=<your-idp-provider> \
    ParameterKey=DiscoveryUrl,ParameterValue=<your-discovery-url> \
    ParameterKey=AllowedAudiences,ParameterValue=<your-allowed-audiences> \
    ParameterKey=AuthFlow,ParameterValue=USER_FEDERATION \
    ParameterKey=SapAuthorizeUrl,ParameterValue=<your-sap-authorize-url> \
    ParameterKey=SapTokenUrl,ParameterValue=<your-sap-token-url> \
    ParameterKey=OauthScopes,ParameterValue=<your-oauth-scopes> \
    ParameterKey=AppCallbackEndpoint,ParameterValue=<your-callback-url> \
    ParameterKey=McpServerLogLevel,ParameterValue=INFO \
    ParameterKey=McpServerReadEnabled,ParameterValue=true \
    ParameterKey=McpServerWriteEnabled,ParameterValue=false \
    ParameterKey=McpServerCreateEnabled,ParameterValue=false \
    ParameterKey=McpServerUpdateEnabled,ParameterValue=false \
    ParameterKey=McpServerDeleteEnabled,ParameterValue=false \
    ParameterKey=McpServerFunctionImportEnabled,ParameterValue=false \
    ParameterKey=UseSapCatalog,ParameterValue=true \
    ParameterKey=McpServerCustomCatalogBucketUri,ParameterValue=None \
    ParameterKey=McpServerServiceHintsS3Uri,ParameterValue=None \
    ParameterKey=AllowedServicePrefixes,ParameterValue=None \
    ParameterKey=McpServerVpcSecurityGroup,ParameterValue=<your-security-group-id> \
    ParameterKey=McpServerNetworkSubnets,ParameterValue=<your-subnet-ids>
```

> **Note**
>
> USER_FEDERATION does not require SapCredentialsSecret. AppCallbackEndpoint is required for this flow only.

What to expect

After a successful deployment, you have a Bedrock AgentCore runtime with AWS for SAP MCP Server installed. Your MCP clients (AI agents) can communicate with the server over Streamable HTTP through the AgentCore invocation endpoint. You can find the invocation endpoint in the CloudFormation stack **Outputs** tab after the stack reaches CREATE_COMPLETE.

Troubleshooting deployment failures

If the CloudFormation stack creation fails (status CREATE_FAILED or ROLLBACK_COMPLETE), open the AWS CloudFormation console, select your stack, and choose the **Events** tab. The events list shows each resource creation attempt in chronological order. Look for the first event with a CREATE_FAILED status to identify the root cause. Common failure reasons include invalid parameter values or insufficient IAM permissions.

Private IdP connectivity failure: If the stack fails while creating the OAuth credential provider or the Runtime, check that the specified subnets can reach the IdP's discovery and token endpoints over HTTPS. Verify that the security groups permit outbound traffic on the IdP's port. For managed Lattice, also verify that the deployer has iam:CreateServiceLinkedRole permission. For more information, see *Troubleshooting*.

Configuration Reference

You configure the AWS for SAP Model Context Protocol (MCP) Server by using environment variables that begin with the prefix `MCP_SERVER_`. These variables fall into two groups: mandatory variables that must be set for the server to start, and optional variables that come with default values. When the server starts, it validates all configurations and returns a descriptive error if any validation rule is violated. After deployment, you can edit these configuration variables directly in the Amazon Bedrock AgentCore Runtime console.

As an example, to modify one of the configuration variable values:

1. Open the **Amazon Bedrock AgentCore console**.
2. In the navigation pane, choose **Runtime**.
3. Under **Build**, choose the AWS for SAP MCP Server that you would like to update from the **Runtime Resources** panel.
4. On the chosen MCP server page, choose **Update Hosting**.
5. Expand the **Advanced Configurations** panel. You will see all the environment variables (for example, `MCP_SERVER_SAP_OAUTH_FLOW`) listed with their corresponding values.
6. Change the desired MCP server configuration variable value and choose **Host agent/tool** for the changes to take effect.

Required environment variables

The following variables must be set for the server to start. Some are conditionally required based on the chosen authentication flow.

Variable	Description	Example
`MCP_SERVER_SAP_BASE_URL`	Base URL of the SAP OData endpoint.	`https://sap.example.com:44301/sap/opu/odata/sap/`

Variable	Description	Example
MCP_SERVER_SAP_OAUTH_FLOW	Authentication flow type. Determines which credential mechanism the server uses.	M2M
MCP_SERVER_BASIC_AUTH_SECRET_NAME	AWS Secrets Manager secret name containing SAP username and password.	my-sap-credentials
MCP_SERVER_OAUTH_PROVIDER	Bedrock AgentCore Identity Provider Name. Required when MCP_SERVER_SAP_OAUTH_FLOW is set to M2M or USER_FEDERATION .	sap-oauth-provider
MCP_SERVER_SAP_OAUTH_SCOPES	OAuth scopes for SAP access. Required when MCP_SERVER_SAP_OAUTH_FLOW is set to M2M or USER_FEDERATION .	ZAPI_SALES_ORDER_SRV_0001
MCP_SERVER_APP_CALLBACK_URL	Callback URL for the interactive OAuth flow. Required when MCP_SERVER_SAP_OAUTH_FLOW is set to USER_FEDERATION .	https://app.example.com/auth

Optional environment variables

The following variables have default values and can be overridden to customize server behavior.

Variable	Default	Description
MCP_SERVER_NAME	SAP-MCP-Server	Name of the MCP server instance.
MCP_SERVER_LOG_LEVEL	INFO	Log verbosity. One of DEBUG, INFO, WARNING, ERROR.
MCP_SERVER_REGION	$AWS_REGION or us-west-2	AWS Region for AWS Secrets Manager, Amazon S3, and AgentCore Identity calls.

Variable	Default	Description
		(When deploying with the CFN template, this value is automatically set to the deployment region `$AWS_REGION` . For custom automation workflows, this parameter must be explicitly set in AgentCore to avoid deployment failures.)
`MCP_SERVER_SAP_SYSTEM`	`S4HANA`	SAP system type. One of `S4HANA`, `ECC`.
`MCP_SERVER_SAP_CLIENT_NUMBER`	None	SAP client number. Must be exactly 3 digits and cannot be `000`.
`MCP_SERVER_READ_ENABLED`	`true`	Enable read tools (`find_sap_services` , `get_metadata` , `odata_read` , `odata_count`).
`MCP_SERVER_WRITE_ENABLED`	`false`	Master switch for all write tools. Must be `true` before any per-operation flag takes effect.
`MCP_SERVER_CREATE_ENABLED`	`false`	Enable the `odata_create` tool. Requires `WRITE_ENABLED=true` .
`MCP_SERVER_UPDATE_ENABLED`	`false`	Enable the `odata_update` tool. Requires `WRITE_ENABLED=true` .
`MCP_SERVER_DELETE_ENABLED`	`false`	Enable the `odata_delete` tool. Requires `WRITE_ENABLED=true` .

Variable	Default	Description
MCP_SERVER_FUNCTION_IMPORT_ENABLED	false	Enables the odata_function_import tool. Requires WRITE_ENABLED=true .
MCP_SERVER_CUSTOM_CATALOG_BUCKET	None	Amazon S3 bucket name for a custom service catalog. The S3 bucket name must start with awsforsap-mcp-server- .
MCP_SERVER_USE_SAP_CATALOG	true	Fetch the service catalog from SAP. If false, CUSTOM_CATALOG_BUCKET is required.
MCP_SERVER_ALLOWED_SERVICE_PREFIXES	*	Comma-separated list of service prefixes for filtering. * means all services.

Enabling Write Operations

The write tools (odata_create, odata_update, odata_delete, and odata_function_import) are disabled by default. Enabling them requires two levels of opt-in:

1. **Master switch** — Set MCP_SERVER_WRITE_ENABLED=true to unlock write capabilities at the MCP Server level.
2. **Per-operation flag** — Enable each write operation individually:

Operation	Environment variable
odata_create	MCP_SERVER_CREATE_ENABLED=true
odata_update	MCP_SERVER_UPDATE_ENABLED=true
odata_delete	MCP_SERVER_DELETE_ENABLED=true

Operation	Environment variable
odata_function_import	MCP_SERVER_FUNCTION_IMPORT_ENABLED=true

Both the master switch and the corresponding per-operation flag must be `true` for a write tool to be available to agents. If `MCP_SERVER_WRITE_ENABLED=true` is set but no per-operation flag is enabled, the server will fail configuration validation at startup.

Custom catalog configuration

With the custom catalog feature, you can extend or replace the SAP service catalog with a catalog that you define and host on Amazon S3. This is useful when your SAP system does not expose all services through `IWFND/CATALOGSERVICE;v=2`, or when you want a curated set of services available to AI agents.

Prerequisites

- An Amazon S3 bucket whose name starts with `awsforsap-mcp-server-` (for example, `awsforsap-mcp-server-mycatalog`).
- Appropriate IAM permissions to read from that bucket.
- Access to set environment variables on the AWS for SAP MCP Server.

Create the catalog file

Create a file named `catalog.json`. The file must use this exact name. Each entry requires two fields:

- **Description** — A human-readable description of the service.
- **ServiceUrl** — The full URL to the SAP OData service endpoint.

Example catalog.json:

```
{
  "SapServices": [
```

```
    {
      "Description": "Custom Business Partner API",
      "ServiceUrl": "https://my-sap-system.example.com/sap/opu/odata/sap/
API_BUSINESS_PARTNER"
    },
    {
      "Description": "Custom Sales Order API with version",
      "ServiceUrl": "https://my-sap-system.example.com/sap/opu/odata/sap/
API_SALES_ORDER;v=0002"
    },
    {
      "Description": "Custom Inventory Service",
      "ServiceUrl": "https://my-sap-system.example.com/sap/opu/odata/custom/
Z_INVENTORY_SRV"
    }
  ]
}
```

Constraints:

- The root object must contain an `SapServices` array.
- The catalog supports a maximum of 1024 entries.
- Both `Description` and `ServiceUrl` are required and must be non-empty.
- If duplicate service names exist within your custom catalog, the server keeps the last occurrence.

Upload the catalog to Amazon S3

Upload `catalog.json` to the root of your bucket, or into a subfolder:

```
# Root of bucket
aws s3 cp catalog.json s3://awsforsap-mcp-server-mycatalog/catalog.json

# Or into a subfolder
aws s3 cp catalog.json s3://awsforsap-mcp-server-mycatalog/my-environment/catalog.json
```

Configure the MCP server environment variables

Two environment variables control the custom catalog feature:

Variable	Default	Description
MCP_SERVER_CUSTOM_CATALOG_BUCKET	None	Amazon S3 bucket name (and optional sub-path) for the custom catalog. Must start with awsforsap-mcp-server- .
MCP_SERVER_USE_SAP_CATALOG	true	When true, the server fetches the live SAP catalog. When false, the server skips all SAP catalog network requests.

Choose one of the three configuration options below based on your use case.

Merged mode — SAP catalog + custom catalog (most common)

Use this when you want to supplement or override specific entries in the live SAP catalog with your own definitions. The server fetches the SAP catalog at runtime and merges your custom entries on top. Custom entries override SAP entries that share the same service name; new custom entries are added.

```
MCP_SERVER_CUSTOM_CATALOG_BUCKET="awsforsap-mcp-server-mycatalog"
# MCP_SERVER_USE_SAP_CATALOG defaults to true, no need to set it

# If your catalog.json is in a subfolder:
MCP_SERVER_CUSTOM_CATALOG_BUCKET="awsforsap-mcp-server-mycatalog/my-environment"
```

Custom-only mode — no SAP catalog

Use this when you want to disable all SAP catalog network requests and serve only the services you have explicitly defined. This is the right choice for air-gapped environments, testing without a live SAP system, or scenarios where the SAP catalog is unreliable.

```
MCP_SERVER_USE_SAP_CATALOG="false"
MCP_SERVER_CUSTOM_CATALOG_BUCKET="awsforsap-mcp-server-mycatalog"
```

Both variables are required in this mode. If you set MCP_SERVER_USE_SAP_CATALOG to false without setting MCP_SERVER_CUSTOM_CATALOG_BUCKET, the server fails at startup because it would have no service catalog available.

SAP catalog only — no custom catalog (default behavior)

If you do not set `MCP_SERVER_CUSTOM_CATALOG_BUCKET`, the server uses only the live SAP catalog. This is the default behavior and requires no configuration changes.

Deploy the MCP server

Deploy (or redeploy) the AWS for SAP MCP Server after setting the environment variables. During startup, check the logs to confirm your catalog loaded correctly:

- A successful custom catalog load logs the number of entries loaded.
- If the bucket name does not start with `awsforsap-mcp-server-`, the server rejects the configuration and logs an error, then falls back to the SAP catalog only.
- If `catalog.json` is missing, malformed, or exceeds 1024 entries, the server logs a warning or error and continues with an empty custom catalog.

Verify your services

Use the `find_sap_services` tool to search for your custom services. The tool searches across all catalog entries (SAP and custom) with no distinction between sources. Existing search and filter logic applies to all entries regardless of origin.

Service hints configuration

With service hints, you can provide custom natural-language guidance for different SAP services. AI agents can look up these hints by using the `get_service_hints` tool.

To enable custom service hints, set the following environment variable:

```
MCP_SERVER_SERVICE_HINTS_S3_URL=s3://awsforsap-mcp-server-service-hints-bucket/path/to/
file.json
```

After you configure service hints, the `get_service_hints` tool becomes available to AI agents. The tool returns usage guidance for a requested service from the hints file.

Hints file schema

The service hints file must follow the JSON schema below. The two required top-level fields are `version` and `hints`.

```
{
```

```
  "version": "1.0",
  "_description": "Optional human-readable description of the hints file",
  "hints": [
    {
      "pattern": "API_BUSINESS_PARTNER",
      "priority": 10,
      "service_type": "Business Partner API",
      "known_issues": [
        "Pagination may return inconsistent results when filters change between pages"
      ],
      "workarounds": [
        "Use $skiptoken instead of $skip for stable pagination"
      ],
      "notes": [
        "Always include AddressData in $expand for complete partner records"
      ],
      "field_hints": {
        "BusinessPartner": {
          "type": "Edm.String",
          "format": "10-digit numeric string",
          "example": "0001000000",
          "description": "Unique business partner identifier",
          "constraints": {
            "required": true,
            "maxLength": 10,
            "pattern": "^[0-9]{10}$"
          }
        }
      },
      "tags": ["master-data", "business-partner"]
    }
  ]
}
```

Schema field reference

Top-level fields:

Field	Type	Required	Description
`version`	String	Yes	Schema version identifier (for example, `1.0`).

Field	Type	Required	Description
`hints`	Array	Yes	Array of service-specific hint configurations.
`_description`	String	No	Human-readable description of the hints file purpose.
`_note`	String	No	Additional notes about how hints are processed.

Hint entry fields:

Field	Type	Required	Description
`pattern`	String	Yes	URL pattern or exact service name to match. Supports wildcards (*).
`priority`	Integer	No	Priority level for hint matching. Higher values override lower ones. Default: `10`.
`service_type`	String	No	Human-readable service type or name.
`known_issues`	Array of strings	No	List of known issues with this service.
`workarounds`	Array of strings	No	List of workarounds for known issues.
`notes`	Array of strings	No	Additional notes and guidance for using this service.
`field_hints`	Object	No	Field-specific hints and metadata. Keys are field names, values are field hint objects.

Field	Type	Required	Description
`tags`	Array of strings	No	Optional tags for categorizing or filtering hints. Must be unique.
`metadata`	Object	No	Additional metadata for extensibi lity.

Field hint fields:

Field	Type	Description
`type`	String	Data type of the field (for example, `Edm.String` , `Edm.Int32`).
`format`	String	Format or pattern description for the field value.
`example`	String, number, boolean, or array	Example value for the field.
`description`	String	Detailed description of the field and its usage.
`constraints`	Object	Field constraints including `required`, `minLength` , `maxLength` , `pattern`, and `enum`.
`notes`	Array of strings	Additional notes about the field.

Service Prefixes

Service prefix filtering restricts which SAP OData services are discoverable by AI agents. When enabled, the `find_sap_services` tool returns only services whose technical name starts with one of the configured prefixes.

```
# Allow only services starting with ZAPI_ or ZCUSTOM_
MCP_SERVER_ALLOWED_SERVICE_PREFIXES=ZAPI_,ZCUSTOM_
```

```
# Allow all services
MCP_SERVER_ALLOWED_SERVICE_PREFIXES=*
```

Cross-validation Rules

The server enforces the following cross-validation rules at startup. If any rule is violated, the server logs a descriptive error and exits.

1. **Write operations require at least one operation enabled.** If `MCP_SERVER_WRITE_ENABLED=true`, at least one of `MCP_SERVER_CREATE_ENABLED`, `MCP_SERVER_UPDATE_ENABLED`, `MCP_SERVER_DELETE_ENABLED`, or `MCP_SERVER_FUNCTION_IMPORT_ENABLED` must also be `true`.
2. **Custom catalog required when SAP catalog is disabled.** If `MCP_SERVER_USE_SAP_CATALOG=false`, then `MCP_SERVER_CUSTOM_CATALOG_BUCKET` must be set.
3. **OAuth provider validated at startup.** When the authentication flow is `M2M` or `USER_FEDERATION`, the server validates the `MCP_SERVER_SAP_OAUTH_PROVIDER` value against Bedrock AgentCore Identity during startup.
4. **Basic auth secret validated at startup.** When the authentication flow is `BASIC`, the server validates that the secret specified in `MCP_SERVER_BASIC_AUTH_SECRET_NAME` exists in AWS Secrets Manager.

Troubleshooting

Use this section to diagnose and resolve common issues with the AWS for SAP Model Context Protocol (MCP) Server. You can find CloudWatch Logs Insights queries for log analysis, common error scenarios with resolutions, and guidance on contacting AWS Support.

CloudWatch Logs Insights queries

The AWS for SAP MCP Server emits structured logs to CloudWatch Logs. You can use the following queries to diagnose common issues. Select the log group for your MCP Server container and set the desired time range.

All errors in time range

```
fields @timestamp, @message
| filter @message like /ERROR/
| sort @timestamp desc
| limit 100
```

Session trace by ID

```
fields @timestamp, @message
| filter @message like /\[session:<session-id>\]/
| sort @timestamp asc
```

Authentication failures

```
fields @timestamp, @message
| filter @message like /ODataAuthError|AuthenticationError|401|403/
| sort @timestamp desc
| limit 50
```

Tool execution failures

```
fields @timestamp, @message
| filter @message like /TOOL CALL END \(FAILURE\)/
| sort @timestamp desc
| limit 50
```

Tool execution latency

```
fields @timestamp, @message
| filter @message like /Execution Time:/
| parse @message "Execution Time: *s" as exec_time
| stats avg(exec_time), max(exec_time), min(exec_time) by bin(5m)
```

Configuration validation errors at startup

```
fields @timestamp, @message
| filter @message like /ValidationError|Invalid configuration|ValueError/
| sort @timestamp desc
| limit 50
```

Function import audit log

```
fields @timestamp, @message
| filter @message like /\[AUDIT\]/
| sort @timestamp desc
| limit 100
```

Common error scenarios

Scenario	Symptoms	Resolution
SAP system unreachable	Connection timeout errors, `Error fetching SAP services` in logs.	Verify `SAP_BASE_URL` is correct and the SAP system is running. Check network connectivity and security group configuration.
Invalid credentials	`ODataAuthError`, HTTP 401 status code.	For BASIC auth: verify the AWS Secrets Manager secret contains the correct username and password. For M2M/USER_FEDERATION: verify the OAuth provider name and scopes.
CSRF token failures	HTTP 403 status code on write operations.	Retry the operation — the server fetches a fresh CSRF token on each request. If the issue

Scenario	Symptoms	Resolution
		persists, verify that the SAP system's CSRF token endpoint is accessible.
Service catalog empty	`No services found` in logs.	Verify that `IWFND/CATALOGSERVICE;v=2` is accessible on the SAP system. Check SAP user authorization. If using a custom catalog, verify the Amazon S3 bucket name and JSON file validity.
Configuration validation failures	Server fails to start, `ValidationError` in logs.	Review environment variables against the validation rules in *Configuration Reference*.
Custom catalog: server rejects config at startup	Error logged at startup, server falls back to SAP catalog.	Verify the Amazon S3 bucket name starts with `awsforsap-mcp-server-` and that the bucket exists in the correct AWS account and Region.
Custom catalog: server fails with error at startup	`MCP_SERVER_USE_SAP_CATALOG=false` without `MCP_SERVER_CUSTOM_CATALOG_BUCKET` set.	Set `MCP_SERVER_CUSTOM_CATALOG_BUCKET` or re-enable the SAP catalog.
Custom catalog: services not appearing	`catalog.json` not found, access denied, or invalid JSON.	Verify the file is named exactly `catalog.json` and is in the correct bucket/path. Check IAM permissions. Check server logs for specific validation errors.
Custom catalog: individual entry missing	Title derivation failed for that entry.	Verify the `ServiceUrl` follows the `/sap/opu/odata/{namespace}/{SERVICE_NAME}` pattern.

Scenario	Symptoms	Resolution
Custom catalog: exceeds entry limit	Catalog has more than 1024 entries.	Reduce the number of entries to 1024 or fewer.
Custom catalog: changes not reflected	Catalog is loaded at startup only.	Restart the MCP server to pick up changes to `catalog.json` .
Private IdP unreachable	OAuth token exchange fails with connection timeout or DNS resolution errors. Server logs show failures reaching the SAP/IdP token endpoint.	If the SAP authorization server or external IdP is hosted inside a VPC, confirm that a `privateEndpoint` is configured on the AgentCore OAuth credential provider. Verify that the specified subnets and security groups allow HTTPS to the IdP. For more information, see Connect to private identity providers.

Security

Learn about the security features of the Model Context Protocol (MCP) Server, including its network configuration, create, read, update, and delete (CRUD) default read-only mode, credential management, and authentication and authorization mechanisms.

Network security

The AWS for SAP MCP Server runs in virtual private cloud (VPC) mode on Amazon Bedrock AgentCore Runtime. Network access is controlled through VPC security groups and subnets that you configure during deployment. You can manage network access to Amazon Bedrock AgentCore by using resource-based policies.

Create, Read, Update, Delete (CRUD) operations

The MCP Server runs in read-only mode by default. All write-related configuration flags are set to `false` out of the box. No write tool is registered with the MCP Server unless you explicitly opt in. This granular model helps you grant only the specific write capabilities that your use case requires.

Authentication

The server enforces access control at two layers: inbound and outbound.

- **Inbound access control:** Amazon Bedrock AgentCore runtime validates all incoming requests before they reach the MCP Server. When configured with JSON Web Token (JWT) based authentication, the runtime verifies the token signature, issuer, audience, and expiration against the configured identity provider's discovery endpoint. Requests with missing, expired, or invalid tokens are rejected with a 401 response.
- **Outbound access control:** The MCP Server authenticates to SAP using the configured authentication flow (Basic, M2M, or User Federation). Credentials are never hardcoded in the MCP server or its configuration. For Basic authentication, credentials are retrieved from AWS Secrets Manager on each request. For OAuth flows, tokens are obtained via Amazon Bedrock AgentCore Identity at runtime. The MCP Server does not store or cache outbound credentials.

Authorization

When you use OAuth authentication flows, the OAuth scopes that you configure through the CloudFormation template determine which SAP OData services the server is authorized to access. Configure scopes to grant access only to the specific services required, following the principle of least privilege. Assign the SAP user associated with the authentication flow only the minimum necessary roles and authorizations in SAP.

Credential management

The MCP Server is designed to never store credentials on disk. Sensitive values, such as authentication or cross-site request forgery (CSRF) tokens, are automatically redacted from logs.

- **OAuth 2.0** — Access tokens are held in-memory only via the TokenStore dataclass. Tokens are not persisted to disk or written to any external store.
- **BASIC** — When using the BASIC authentication flow, SAP system credentials are retrieved from AWS Secrets Manager at runtime on each request. The credentials exist only in memory for the duration of the request.

Version management

You access the AWS for SAP Model Context Protocol (MCP) Server as a container image distributed through AgentCore Runtime. There are two ways to consume the image, each offering a different trade-off between convenience and control.

Image tags

`latest` tag (auto-updating)

If your container configuration references `aws-sap-mcp:latest`, you automatically receive the newest version the next time your container runtime pulls the image. No action is required — the `latest` tag is re-pointed atomically to the new image during each deployment.

Version-specific tags (pinned, immutable)

Each deployment also produces a version-specific tag derived from the server version (for example, `1.0.0-200`). Version-specific tags are immutable — once pushed, they cannot be overwritten. Use version-specific tags when you need deterministic deployments or want to control exactly when version changes are applied.

To update to a newer version, update the image URI in your AgentCore Runtime configuration to reference the desired version tag and update the runtime hosting.

Rollback procedure

To roll back to a previous version, change your Amazon Bedrock AgentCore Runtime to reference an older version-specific tag. Then, create a new version of the AgentCore Runtime endpoint. For instructions on creating a new version of the AgentCore Runtime endpoint, see the AgentCore Runtime documentation.

Document history

The following table describes important changes to the AWS for SAP MCP Server User Guide.

Change	Description	Date
Initial release	Initial release of the AWS for SAP MCP Server User Guide.	April 30, 2026