



Home    Developer guides    API reference    No-code tools    More ▾

Betas overview  
Beta specifications  
MCP Tools (beta)

Overview  
**Get Started**  
Resources  
Account registration (beta)  
Brand Stores (beta)  
Events (beta)  
Geographic Insights and Activation (beta)  
Linked accounts (beta)  
Reporting API (beta)

Overview  
Quickstart guide  
Helpful concepts  
New features  
Reports resource  
Additional topics  
Dimensions  
Metrics  
Amazon Marketing Stream (closed beta)

# Connecting to the Amazon Ads MCP Server

## Getting Started:

**Step 1:** Use an existing LwA application and Amazon Developer account with access to the Amazon Ads API complete the onboarding steps ↗

**Step 2:** Complete an authorization grant ↗ and retrieve **access** and **refresh** tokens. Save the access and refresh tokens - you'll need them to connect to the MCP server later.

feedback

## Account Identifier Context Modes

While the Amazon Ads API requires you to input specific accountIds in headers or fields, like inserting a profileId into the Amazon-Advertising-API-Scope header, the MCP server handles this differently.

By default, the MCP server will use a **Dynamic Account Context** mode. In this mode, the LLM will ask you provide the account identifiers needed to invoke each selected tool. You can use the `query_advertiser_accounts` tool to get all accounts the user has access to. Once provided to the LLM this account may be reused for subsequent requests or until new account identifiers are provided to the LLM. The MCP server requires the account identifiers to be passed as a parameter in the request body.

- **profileId** : The identifier of a profile associated with the advertiser account.
- **managerAccountId** : The manager account id.
- **advertiserAccountId** : The identifier can be either a DSP advertiser account, global account id or AMC instance id depending on the tool you are trying to call.

If you would instead like to support an account identifier context which behaves more closely to the Amazon Ads APIs, we also support a **Fixed Account Context** mode. In this mode, you can set your account identifier as statically-defined headers. Doing so will indicate to our server to not prompt your LLM for account information, and instead will use the pre-set headers for this information. Please note that for this mode, these headers should correspond with a single account.

## Required Headers

In order to connect to the Amazon Ads MCP server, each request to the Amazon Ads MCP server requires the following headers:

- **Amazon-Ads-ClientId** : The client identifier of an LwA application authorized to access the MCP Server.
- **Authorization** : The string **Bearer** prepended to an **access token** representing that application's permission to access data and services for a given Amazon user.

**Fixed Account Context** requires the **additional headers** defined below: profile ID header, account ID header or manager account ID header, and a header to indicate fixed mode. Dynamic Account Context however only needs the two basic headers defined above (Amazon-Ads-ClientId and Authorization).

- **Amazon-Ads-AI-Account-Selection-Mode** : The header that indicates to the MCP server that the client wants to operate in fixed account scope mode.
  - Currently, we only support "Amazon-Ads-AI-Account-Selection-Mode: FIXED"
  - If you prefer to use dynamically-set account identifiers, simply do not include this header
- **Amazon-Advertising-API-Scope** : The identifier of a profile associated with the advertiser account.
- **Amazon-Ads-AccountID** : The identifier can be either a DSP advertiser account, global account id or AMC instance id depending on the tool you are trying to call.
- **Amazon-Ads-Manager-AccountID** : Alternatively to profile id or account id, you could pass in the manager account id. Please note that not all tools work on manager account scope.

Please note that not all three account identifier headers (Amazon-Advertising-API-Scope, Amazon-Ads-Advertiser-Account-ID, and Amazon-Ads-Manager-Account-ID) are required for every tool.

**amazon ads** Advanced tools center 🔍

---

- Betas overview
- Beta specifications
- MCP Tools (beta)**
  - Overview
  - Get Started**
  - Resources
  - Account registration (beta)
  - Brand Stores (beta)
  - Events (beta)
  - Geographic Insights and Activation (beta)
  - Linked accounts (beta)
- Reporting API (beta)**
  - Overview
  - Quickstart guide
  - Helpful concepts
  - New features
  - Reports resource
  - Additional topics
  - Dimensions
  - Metrics
- Amazon Marketing Stream (closed beta)

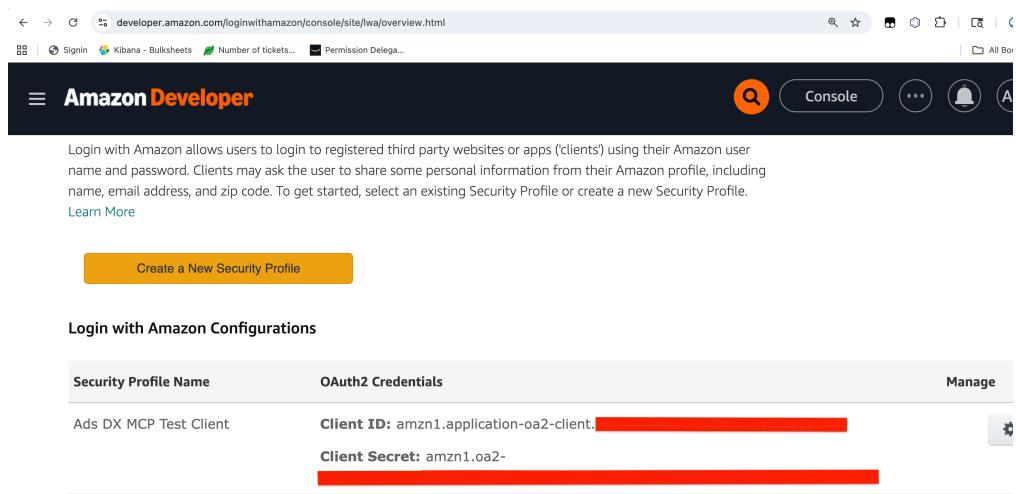
These URLs are not region agnostic. For connecting to the server in the NA region, please use the NA url

Region	Endpoint URL
NA (North America)	<a href="https://advertising-ai.amazon.com/mcp">https://advertising-ai.amazon.com/mcp</a>
EU (Europe)	<a href="https://advertising-ai-eu.amazon.com/mcp">https://advertising-ai-eu.amazon.com/mcp</a>
FE (Far East)	<a href="https://advertising-ai-fe.amazon.com/mcp">https://advertising-ai-fe.amazon.com/mcp</a>

## Connecting to the MCP server using Kiro CLI

**Step 1: Fetch your application's `clientId` as well as `access token` and `refresh token`**

Sign-in to <https://developer.amazon.com/> with your Amazon **developer** account credentials, and then navigate to <https://developer.amazon.com/loginwithamazon/console/site/lwa/overview.html>. You must be able to retrieve your `clientId` from that page as shown below:



**Step 2: Connect to the MCP Server via Kiro CLI by adding the `mcp server config` to `~/.kiro/settings/mcp.json`.**

**Note**  
Ensure that you fill in your `clientId` and `access token` in the `Amazon-Ads-ClientId` and `Authorization` headers.

**Fixed Account Context:**

```
{
  "mcpServers": {
    "amzn-ads-mcp": {
      "url": "https://advertising-ai.amazon.com/mcp",
      "headers": {
        "Authorization": "Bearer Atza|<your_access_token>",
        "Amazon-Ads-ClientId": "<your_client_id>",
        "Amazon-Ads-AI-Account-Selection-Mode": "FIXED",
        "Amazon-Advertising-API-Scope": "<your_profile_id>",
        "Accept": "application/json, text/event-stream"
      },
      "timeout": 60000,
      "disabled": false
    }
  }
}
```

[https://advertising.amazon.com/API/docs/en-us/mcp/get-started#tag/Target/operation/update\\_target\\_bid](https://advertising.amazon.com/API/docs/en-us/mcp/get-started#tag/Target/operation/update_target_bid)

2/9



Betas overview

Beta specifications

## ▼ MCP Tools (beta)

Overview

**Get Started**

Resources

Account registration (beta)

Brand Stores (beta)

Events (beta)

Geographic Insights and Activation (beta)

Linked accounts (beta)

## ▼ Reporting API (beta)

Overview

Quickstart guide

Helpful concepts

New features

Reports resource

Additional topics

Dimensions

Metrics

Amazon Marketing Stream (closed beta)

Amazon-Ads-AccountID or Amazon-Ads-Manager-AccountID can also be used above in addition to or instead of Amazon-Advertising-API-Scope.

**Dynamic Account Context:**

```
{
  "mcpServers": {
    "amzn-ads-mcp": {
      "url": "https://advertising-ai.amazon.com/mcp",
      "headers": {
        "Authorization": "Bearer Atza|<your_access_token>",
        "Amazon-Ads-ClientId": "<your_client_id>",
        "Accept": "application/json, text/event-stream"
      },
      "timeout": 60000,
      "disabled": false
    }
  }
}
```

**Step 3 (optional): Setup tool filters through Kiro config**

If you would like to disable any of the available tools, you can do so through your Kiro config by adding `disabledTools`. The below example disables the `deletecampaign` and `deletead` tools.

```
{
  "mcpServers": {
    "amzn-ads-mcp": {
      "url": "https://advertising-ai.amazon.com/mcp",
      "headers": {
        "Authorization": "Bearer Atza|<your_access_token>",
        "Amazon-Ads-ClientId": "<your_client_id>",
        "Accept": "application/json, text/event-stream"
      },
      "timeout": 60000,
      "disabled": false,
      "disabledTools": ["campaign_management-delete_campaign", "campaign_ma
    }
  }
}
```

**Step 4: Run Kiro CLI**

Once your MCP configuration is properly set up:

1. Start Kiro CLI by running `kiro-cli`
2. Look for a tick mark (✓) next to your MCP server name during startup - this indicates a successful connection
3. To verify the connection, use the `/mcp` command to see the list of connected MCP servers
4. To see available tools, use the `/tools` command to list available tools.

This confirms your Amazon Ads MCP server is properly connected and ready to use.

```
% kiro-cli
```

**amazon ads** Advanced tools center

Betas overview  
Beta specifications  
**MCP Tools (beta)**  
Overview  
**Get Started**  
Resources  
Account registration (beta)  
Brand Stores (beta)  
Events (beta)  
Geographic Insights and Activation (beta)  
Linked accounts (beta)  
**Reporting API (beta)**  
Overview  
Quickstart guide  
Helpful concepts  
New features  
Reports resource  
Additional topics  
Dimensions  
Metrics  
Amazon Marketing Stream (closed beta)

**Did you know?**  
Run `/prompts` to learn how to build & run repeatable workflows

Model: Auto (`/model` to change)

> /tools

Tool	Permission
<b>Built-in</b>	
- code	trust read-only operations
- shell	not trusted
- read	trust working directory
- write	not trusted
- glob	trust working directory
- grep	trust working directory
- introspect	<b>trusted</b>
- report	not trusted
- aws	trust read-only commands
- subagent	not trusted
- web_fetch	not trusted
- web_search	not trusted
<b>amzn-ads-mcp (MCP)</b>	
- account_management-query_account_link	not trusted
- account_management-query_advertiser_account	not trusted
- account_management-update_account_currency	not trusted
- account_management-update_account_name	not trusted
- account_management-update_account_timezone	not trusted
- account_management-update_advertiser_account	not trusted
- billing-list_invoices	not trusted
- billing-query_billing_notifications	not trusted
- campaign_management-add_country_campaign	not trusted
- campaign_management-create_ad	not trusted
- campaign_management-create_ad_association	not trusted
- campaign_management-create_ad_group	not trusted
- campaign_management-create_campaign	not trusted
- campaign_management-create_campaign_harvest_targets	not trusted
- campaign_management-create_singleshot_sp_campaign	not trusted
- campaign_management-create_target	not trusted
- campaign_management-delete_ad	not trusted
- campaign_management-delete_ad_association	not trusted
- campaign_management-delete_ad_group	not trusted
- campaign_management-delete_campaign	not trusted
- campaign_management-delete_target	not trusted
- campaign_management-query_ad	not trusted
- campaign_management-query_ad_association	not trusted

**Step 5: Utilize the Amazon Ads MCP Server**

In the **fixed account context**, you can start directly by creating campaigns or performing other operations

Example of next steps:

Create an SP campaign

> Can you create an SP Campaign for me? Name it "Valentines Day Feb 2026" with a Start Date of Feb 1st and a budget of \$



- Betas overview
- Beta specifications
- MCP Tools (beta)**
  - Overview
  - Get Started**
  - Resources
- Account registration (beta)
- Brand Stores (beta)
- Events (beta)
- Geographic Insights and Activation (beta)
- Linked accounts (beta)
- Reporting API (beta)**
  - Overview
  - Quickstart guide
  - Helpful concepts
  - New features
  - Reports resource
  - Additional topics
  - Dimensions
  - Metrics

```
... "adProduct": "SPONSORED_PRODUCTS",
... "autoCreationSettings": {
...   "autoCreateTargets": false
... },
... "budgets": [
...   {
...     "budgetType": "MONETARY",
...     "budgetValue": {
...       "monetaryBudgetValue": {
...         "monetaryBudget": {
...           "value": 100
...         }
...       }
...     },
...     "recurrenceTimePeriod": "DAILY"
...   ],
...   "marketplaceScope": "SINGLE_MARKETPLACE",
...   "marketplaces": [
...     "US"
...   ],
...   "name": "Valentines Day Feb 2026",
...   "startDateTime": "2026-02-01T00:00:00Z",
...   "state": "ENABLED"
... ]
... ]
```

Allow this action? Use 't' to trust (always allow) this tool for the session. [y/n/t]:

Allow this action? Use 't' to trust (always allow) this tool for the session. [y/n/t]:

> y

- Completed in 1.554s

> Campaign created successfully! Here are the details:

- \*\*Campaign Name\*\*: Valentines Day Feb 2026
- \*\*Campaign ID\*\*: 300378891364541
- \*\*Ad Product\*\*: Sponsored Products
- \*\*Start Date\*\*: February 1, 2026
- \*\*Daily Budget\*\*: \$100 USD
- \*\*Marketplace\*\*: US
- \*\*State\*\*: Enabled
- \*\*Bid Strategy\*\*: Sales Down Only (default)

The campaign is now active and will start running on February 1st, 2026.

> Time: 12s

The Amazon Ads MCP server will use the header provided from the **fixed account context** to create the campaign.

In the **dynamic account context**, before creating campaigns or performing other operations, you must first identify your available advertiser accounts.

#### Note

For clients with multiple marketplaces: First select your marketplace, then select your profile. Following this sequence ensures operations execute correctly.

1. Ask Kiro to list your accounts with this prompt: Can you give me the advertiser accounts that I have access to?
2. This will trigger the account\_management--query\_advertiser\_account tool and return a list all advertiser accounts which can be accessed with the current context.
3. Note the account IDs from the results - you'll need to specify these in subsequent commands.

Example of next steps:

Create an SP campaign using the profile id for amzn1.ads-account.g.id

The account ID retrieved becomes the profile/account identifier for all future campaign management operations.

> Can you show me which accounts I have access to?

**amazon ads** Advanced tools center

Betas overview  
Beta specifications  
**MCP Tools (beta)**

- Overview
- Get Started**
- Resources
- Account registration (beta)
- Brand Stores (beta)
- Events (beta)
- Geographic Insights and Activation (beta)
- Linked accounts (beta)

**Reporting API (beta)**

- Overview
- Quickstart guide
- Helpful concepts
- New features
- Reports resource
- Additional topics
- Dimensions
- Metrics

Amazon Marketing Stream (closed beta)

Allow this action? Use 't' to trust (always allow) this tool for the session. [y/n/t]:  
**> y**  
- Completed in 4.653s

> You have access to 12 advertiser accounts. Here's the breakdown:

**US Accounts (9):**

1. 3P\_US\_VENDOR\_
2. 3P\_US\_VENDOR\_
3. 3P\_US\_VENDOR\_
4. 3P\_US\_AUTHOR\_
5. pMOAnDQg8ENC5
6. 3P\_US\_VENDOR\_
7. 3P\_US\_VENDOR\_
8. 3P\_US\_VENDOR\_
9. 3P\_US\_VENDOR\_

- ID: amzn1.ads-account.g.  
- ID: amzn1.ads-account.g.  
- ID: amzn1.ads-account.g.  
- ID: amzn1.ads-account.g.

**CA Accounts (2):**

10. 3P\_CA\_VENDOR\_TESTACCOUNT
11. 3P\_CA\_VENDOR\_TESTACCOUNT

- ID: amzn1.ads-account.g.  
- ID: amzn1.ads-account.g.

> Can you create an SP Campaign for account 'amzn1.ads-account.g.'? Name it "Winter Flash Sale 2026" with a Start Date of Feb 1st and a Budget of \$100?  
Running tool `campaign.management-create_campaign` with the param (from mcp server: amzn-ads-mcp)

```
{
  "body": {
    "accessRequestedAccount": {
      "advertiserAccountId": "amzn1.ads-account.g."
    },
    "campaigns": [
      {
        "adProduct": "SPONSORED_PRODUCTS",
        "autoCreationSettings": {
          "autoCreateTargets": false
        },
        "budgets": [
          {
            "budgetType": "MONETARY",
            "budgetValue": {
              "monetaryBudget": {
                "value": 100
              }
            },
            "recurrenceTimePeriod": "DAILY"
          }
        ],
        "marketplaceScope": "SINGLE_MARKETPLACE",
        "marketplaces": [
          "US"
        ],
        "name": "Winter Flash Sale 2026",
        "startTime": "2026-02-01T00:00:00Z",
        "state": "ENABLED"
      }
    ]
  }
}
```

Allow this action? Use 't' to trust (always allow) this tool for the session. [y/n/t]:  
**> y**  
- Completed in 2.132s

> Campaign created successfully!

**Campaign Details:**

- **Name:** Winter Flash Sale 2026
- **Campaign ID:** 447936506865282
- **Type:** Sponsored Products
- **Marketplace:** US
- **Start Date:** February 1, 2026
- **Daily Budget:** \$100 USD
- **State:** Enabled
- **Bid Strategy:** Sales Down Only

The campaign is ready and will start running on February 1st, 2026.  
**> Time: 13s**

## Sample Python Client Library & Conversational Agent

If you are interested in connecting to the Amazon Ads MCP Server programmatically through Python, we provide a sample client library implementation and a working demo that you can copy and customize for your needs. The library includes a fluent builder interface for easy configuration, support for all regions (NA, EU, FE), optional tool filtering, and automatic handling of statically-defined account identifier headers if preferred.

**Prerequisites:** You will need to install the following dependencies which are used for initializing your client.



Betas overview
Beta specifications
<b>▼ MCP Tools (beta)</b>
Overview
<b>Get Started</b>
Resources
Account registration (beta)
Brand Stores (beta)
Events (beta)
Geographic Insights and Activation (beta)
Linked accounts (beta)
<b>▼ Reporting API (beta)</b>
Overview
Quickstart guide
Helpful concepts
New features
Reports resource
Additional topics
Dimensions
Metrics
Amazon Marketing Stream (closed beta)

to connect your agent to an LLM.

For more on foundational model access with AWS, see here:

<https://docs.aws.amazon.com/bedrock/latest/userguide/model-access.html>

First, install the AWS CLI if it isn't already installed. After that, you can run `aws configure` to connect your local environment to an AWS account that has access to your desired model. See [here](#) for more on using `aws configure`.

Finally, fetch your application's **clientId**, **access token**, and **refresh token**. Sign-in to <https://developer.amazon.com/> with your Amazon developer account credentials, and then navigate to <https://developer.amazon.com/loginwithamazon/console/site/lwa/overview.htm> to retrieve the required id and tokens.

### Sample Client + Conversational Playground

Create a Python file in your local environment and paste in the following code snippet:

```
from mcp.client.streamable_http import streamablehttp_client
from strands.tools.mcp import MCPClient
import sys
import io
from enum import Enum

class Region(Enum):
    NA = "na"
    EU = "eu"
    FE = "fe"

def _derive_server_url(region: str) -> str:
    if region == "na":
        return "https://advertising-ai.amazon.com/mcp"
    elif region == "eu":
        return "https://advertising-ai-eu.amazon.com/mcp"
    elif region == "fe":
        return "https://advertising-ai-fe.amazon.com/mcp"
    raise ValueError(f"Unsupported region: {region}")

class AmazonAdsMCPClient:
    def __init__(self, mcp_client: MCPClient):
        self._client = mcp_client

    def __enter__(self):
        self._client.__enter__()
        return self

    def __exit__(self, exc_type, exc_val, exc_tb):
        old_stderr = sys.stderr
        sys.stderr = io.StringIO()
        try:
            self._client.__exit__(exc_type, exc_val, exc_tb)
        except Exception:
            pass
        finally:
            sys.stderr = old_stderr

    def get_tools(self, disabled_tools=None):
        """
        Get available tools from the MCP server, excluding disabled tools.
        """
        Args:
            disabled_tools: List of tool names to exclude

        Returns:
            List of available MCP tools
        """
        all_tools = self._client.list_tools_sync()
        if not disabled_tools:
            return all_tools

        disabled_set = set(disabled_tools)
        return [tool for tool in all_tools if tool.tool_name not in disabled_
```

```
class MCPClientBuilder:
    def __init__(self):
```



Betas overview
Beta specifications
<b>▼ MCP Tools (beta)</b>
Overview
<b>Get Started</b>
Resources
Account registration (beta)
Brand Stores (beta)
Events (beta)
Geographic Insights and Activation (beta)
Linked accounts (beta)
<b>▼ Reporting API (beta)</b>
Overview
Quickstart guide
Helpful concepts
New features
Reports resource
Additional topics
Dimensions
Metrics
Amazon Marketing Stream (closed beta)

```

        self._manager_account_id = None

    def region(self, region: Region):
        self._region = region.value
        return self

    def client_id(self, client_id: str):
        self._client_id = client_id
        return self

    def auth_token(self, auth_token: str):
        self._auth_token = auth_token
        return self

    def account_id(self, account_id: str):
        self._account_id = account_id
        return self

    def profile_id(self, profile_id: str):
        self._profile_id = profile_id
        return self

    def manager_account_id(self, manager_account_id: str):
        self._manager_account_id = manager_account_id
        return self

    def build(self) -> AmazonAdsMCPClient:
        missing = []
        if not self._client_id:
            missing.append("client_id")
        if not self._auth_token:
            missing.append("auth_token")
        if missing:
            raise ValueError(f"Missing required parameters: {', '.join(missing)}")

        url = _derive_server_url(self._region)
        headers = {
            "Amazon-Ads-ClientId": self._client_id,
            "Authorization": self._auth_token
        }
        # Populate fixed account headers if present
        has_fixed_account = False
        if self._account_id:
            headers["Amazon-Ads-AccountId"] = self._account_id
            has_fixed_account = True
        if self._profile_id:
            headers["Amazon-Advertising-Api-Scope"] = self._profile_id
            has_fixed_account = True
        if self._manager_account_id:
            headers["Amazon-Ads-Manager-AccountId"] = self._manager_account_id
            has_fixed_account = True

        if has_fixed_account:
            headers["Amazon-Ads-AI-Account-Selection-Mode"] = "FIXED"

        mcp_client = MCPClient(lambda: streamablehttp_client(url=url, headers=headers))
        return AmazonAdsMCPClient(mcp_client)

    # Sample Playground Client using Strands
    if __name__ == "__main__":
        from strands import Agent

        try:
            client = MCPClientBuilder() \
                .region(Region.NA) \
                .client_id("<your_client_id_here>") \
                .auth_token("Bearer Atza[<your_lwa_token_here>]") \
                .build()

            with client:
                tools = client.get_tools()
                agent = Agent(tools=tools, model="us.anthropic.claude-3-5-sonnet")
                print(f"\nAgent created with {len(tools)} tools")

                print("\nStarting conversation loop. Type 'quit' to exit.")
                while True:
                    try:
                        user_input = input("\n\033[94mYou: \033[0m").strip()
                        if user_input.lower() in ['quit', 'exit', 'q']:
                            break
                        if not user_input:
                            continue
                    except KeyboardInterrupt:
                        print("\n\nQuitting...\n")
                        break
        except Exception as e:
            print(f"\nAn error occurred: {e}")

```



Betas overview  
Beta specifications

#### ▼ MCP Tools (beta)

Overview  
**Get Started**  
Resources  
Account registration (beta)  
Brand Stores (beta)  
Events (beta)  
Geographic Insights and Activation (beta)  
Linked accounts (beta)

#### ▼ Reporting API (beta)

Overview  
Quickstart guide  
Helpful concepts  
New features  
Reports resource  
Additional topics  
Dimensions  
Metrics

Amazon Marketing Stream (closed beta)

```
print("\nExiting...")  
break  
except Exception as e:  
    print(f"Error: {e}")
```

```
sys.exit(0)
```

```
except Exception as e:  
    print(f"x Error: {e}")  
    sys.exit(1)
```

Now, you can start the conversational agent by simply running your Python script.

Feel free to edit this snippet or use it as reference point to create your own Ads agent.

#### ❗ Important

This snippet is for demo purposes only. Use caution when adapting it for production applications.

If you prefer to use statically-defined account identifiers (Fixed Context Mode), you can add them to your client builder:

```
client = MCPClientBuilder() \  
.region(Region.NA) \  
.client_id("<your_client_id_here>") \  
.auth_token("Bearer Atza!<your_lwa_token_here>") \  
.profile_id("your_profile_id") #Corresponds to Amazon-Advertising  
.build()
```

We support the following account identifier headers. Please only set what is appropriate for your use case:

```
.profile_id("your_profile_id") #Corresponds to Amazon-Advertising-Api-Scope  
.account_id("your_account_id") #Corresponds to Amazon-Ads-AccountId  
.manager_account_id("your_manager_account_id") #Corresponds to Amazon-Ads-Man
```

If you prefer using dynamically-set account identifiers provided by your LLM, simply do not include any of these three values when initializing your client.

#### Troubleshooting:

**Error: the client initialization failed** Most likely, this means that your LwA token has expired. Follow the prerequisite steps above to fetch a new one, update your client initialization with the fresh header, and restart the code snippet execution to create a new session.

**Error: An error occurred (ExpiredTokenException) when calling the ConverseStream operation: The security token included in the request is expired** This indicates that your application does not have access to your AWS credentials, or your credential access has expired. Typically, this error will occur when you try prompting the agent, but your agent does not have access to your selected Lambda function. Follow the prerequisite steps and run aws configure to automatically grant your local environment access to your AWS account.

Was this page helpful?

Yes

No

GitHub RSS release notes

