aws

开发人员指南

Amazon Managed Blockchain 查询

Amazon Managed Blockchain 查询: 开发人员指南

Table of Contents

什么是亚马逊托管区块链 (AMB) 查询？

Amazon Managed Blockchain (AMB) 是一项完全托管的服务，旨在帮助您在公共和私有区块链上构建弹性的 Web3 应用程序。使用 AMB Access 对多个区块链进行即时和无服务器访问。无需部署专门的区块链基础设施并保持与区块链网络的连接，即可构建支持 Web3 的应用程序。借助 AMB Query，您可以使用开发人员友好的 API 操作来访问来自多个区块链的实时和历史数据。标准化的区块链数据可以与 AWS 服务集成，无需专门的区块链基础设施或 ETL（提取、转换和加载）。所有 AMB 功能均可安全扩展，适用于机构级和主流消费类应用程序构建。

Amazon Managed Blockchain (AMB) 查询提供对标准化、多区块链数据集的无服务器访问以及对开发人员友好的 API 操作。您可以使用 AMB Query 快速发布需要来自一个或多个公共区块链数据的应用程序，而无需支付解析区块链数据、跟踪合约和维护专门的索引基础设施的开销。无论您是分析可替代代币还是不可替代代币的历史代币余额（NFTs），查看给定钱包地址的交易历史记录，还是对以太坊等原生加密货币的分布进行数据分析，AMB Query都允许您访问区块链数据。

您是首次使用 AMB Query 的用户吗？

如果您是首次使用 AMB Query 的用户，我们建议您先阅读以下章节：

- 关键概念：亚马逊托管区块链 (AMB) 查询
- 设置亚马逊托管区块链 (AMB) 查询
- 亚马逊托管区块链 (AMB) 查询入门
- 亚马逊托管区块链 (AMB) 查询用例

关键概念：亚马逊托管区块链 (AMB) 查询

> **Note**
>
> 本指南假设您熟悉基本的区块链概念。这些概念包括去中心化、代币、合约、交易 proof-of-work、、钱包、公钥和私钥、质押、采矿、减半等。

Amazon Managed Blockchain (AMB) 查询使您可以方便地访问多区块链网络数据，这使您可以更轻松地提取与区块链活动相关的上下文数据。您可以使用 AMB Query 从公共区块链网络（例如比特币主网和以太坊主网）读取数据。您还可以获取信息，例如地址的当前和历史余额，或者您可以获取给定时间段内的区块链交易列表。此外，您还可以获取给定事务的详细信息，例如交易事件，您可以进一步分析这些细节，或者将其用于应用程序的业务逻辑中。

使用亚马逊托管区块链 (AMB) 查询的注意事项和限制

使用 AMB 查询时，请考虑以下几点：

- 可用区域

  美国东部（弗吉尼亚北部）`us-east-1`区域支持 AMB 查询。

- 服务终端节点

  可以使用以下端点访问 AMB 查询：

  `https://managedblockchain-query.us-east-1.amazonaws.com`.

- 支持的区块链网络

  AMB Query 支持以下公共区块链网络：

  - 比特币主网 — 通过 proof-of-work共识保护的公共比特币区块链网络，比特币（BTC）加密货币是在该网络上发行和交易的。主网上的交易具有实际价值（也就是说，它们会产生实际成本），并记录在公共区块链上。
  - 比特币测试网 — 比特币主网的测试网。该网络上的比特币（BTC）与主网比特币是分开的，并且通常没有任何价值。

- 以太坊主网 — 公共以太坊区块链 proof-of-stake的主网络。主网上的交易具有实际价值（也就是说，它们会产生实际成本），并记录在分布式账本上。
- Sepolia 测试网 — 以太坊主网的测试网。该网络上的以太币（ETH）与主网 ETH 是分开的，并且通常没有任何价值。

- 支持的区块链代币和合约

  AMB Query 支持以下原生和标准以太坊合约代币。

  - 公共区块链原生代币
    - 比特币（BTC）— 这是比特币相关区块链的原生代币。
    - 以太币（ETH）— 这是以太坊相关区块链的原生代币。

  - 以太坊合约标准
    - ERC-20 代币标准 — ERC-20 是可替代代币的标准。它有一个属性，可以使每个 ERC-20 代币与铸造的另一个 ERC-20 代币完全相同（在类型和值上），这意味着一个代币现在和将来都等于所有其他代币。欲了解更多信息，请参阅 Ethereum.org 上的 ERC-20 代币标准。
    - ERC-721 不可替代代币标准 — ERC-721 是不可替代代币的标准 ()。NFTs这种类型的代币是独一无二的，其价值可能与同一合约中的另一种代币不同，这可能是由于其年龄、稀有度或其他属性所致。欲了解更多信息，请参阅 Ethereum.org 上的 ERC-721 代币标准。

      ERC-1155 多代币标准 — ERC-1155 是一个创建合约接口的标准，该接口可以表示和控制任意数量的可替代和不可替代的代币类型。通过这种方式，ERC-1155 代币的功能可以与 ERC-20 和 ERC-721 代币相同，甚至可以同时发挥两者的作用。ERC-1155 代币改进了 ERC-20 和 ERC-721 标准的功能，使其更加高效，同时纠正了明显的实现错误。欲了解更多信息，请参阅 Ethereum.org 上的 ERC-1155 代币标准。
- 终局性

  在区块链中，最终性意味着有效的交易不太可能被撤销。对于比特币主网，AMB Query认为交易在6个区块后最终完成。对于比特币测试网，它认为交易在6个区块或60分钟后完成，以先到者为准。对于支持的以太坊网络，AMB Query 认为交易在 64 个区块后最终完成。

  AMB Query 的代币余额和合约 API 操作仅返回已完成的数据。但是，AMB Query 的交易和交易事件 API 操作可以返回区块链网络上已确认的交易的数据，即使这些交易尚未最终确定。

- 不支持空地址

  AMB 查询不支持 NULL (0x0000000000000000000000000000000000000000) 地址。

- 签名版本 4 对 API 调用进行签名

  调用 AMB 查询时 APIs，您可以通过使用签名版本 4 签名流程进行身份验证的 HTTPS 连接进行调用。这意味着只有 AWS 账户中获得授权的 IAM 委托人才能调用 AMB 查询 API。为此，必须在呼叫中提供 AWS 证书（访问密钥 ID 和私有访问密钥）。

  > **Important**
  >
  > 不要在面向用户的应用程序中嵌入客户端凭据。

- AMB Query 支持比特币交易标识符和交易哈希

  对于比特币网络，AMB Query API 操作同时支持交易标识符 (`transactionId`) 和交易哈希 (`transactionHash`)。`transactionId`是交易的双 SHA 哈希值，不包括见证人数据。`transactionHash`是交易的双 SHA 哈希值，包括见证人数据（也称为见证人交易 ID）。

  在为比特币网络调用`GetTransaction`或 `ListTransactionEvents`API 操作时，您可以指定`transactionId`或。`transactionHash`此外，比特币网络上所有返回 a `transactionId` 或 a 的 AMB Query 操作都`transactionHash`将同时包含这两个值作为响应的一部分。

设置亚马逊托管区块链 (AMB) 查询

在您首次使用亚马逊托管区块链 (AMB) 查询之前，请按照本节中的步骤创建 AWS 账户。以下部分讨论如何开始使用 AMB 查询。

先决条件和注意事项

在首次使用 Amazon Web Services 之前，您必须拥有一个 AWS 账户。

报名参加 AWS

当您注册 Amazon Web Services (AWS) 时，您的 AWS 账户将自动注册所有账户 AWS 服务，包括亚马逊托管区块链 (AMB) 查询。您只需为使用的服务付费。

如果您 AWS 账户 已经有，请转到下一步。如果您还没有 AWS 账户，请使用以下步骤创建。

创建 AWS 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明操作。

   在注册时，将接到电话或收到短信，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为最佳安全实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

创建具有适当权限的 IAM 用户

要创建和使用 AMB Query，您必须创建一个 AWS Identity and Access Management (IAM) 委托人（用户或群组），其权限允许必要的托管区块链操作。

只有 IAM 委托人可以发出 AMB 查询 API 请求。调用 AMB 查询时 APIs，您可以通过使用签名版本 4 签名流程进行身份验证的 HTTPS 连接进行调用。这意味着只有 AWS 账户中获得授权的 IAM 委托人才能调用 AMB 查询 API。为此，必须在呼叫中提供 AWS 证书（访问密钥 ID 和私有访问密钥）。

有关如何创建 IAM 用户的信息，请参阅在您的 AWS 账户中创建 IAM 用户。有关如何向用户关联权限策略的更多信息，请参阅更改 IAM 用户的权限。有关可用于向用户授予使用 AMB Query 的权限策略的示例，请参阅亚马逊托管区块链 (AMB) 查询的基于身份的策略示例。

安装和配置 AWS Command Line Interface

如果您尚未这样做，请安装最新的 AWS 命令行界面 (CLI) 以使用来自终端的 AWS 资源。有关更多信息，请参阅安装或更新 AWS CLI的最新版本。

> **Note**
>
> 要进行 CLI 访问，您需要访问密钥 ID 和秘密访问密钥。如果可能，请使用临时凭证代替长期访问密钥。临时凭证包括访问密钥 ID、秘密访问密钥，以及一个指示凭证何时到期的安全令牌。有关更多信息，请参阅 IAM 用户指南中的将临时证书与 AWS 资源配合使用。

使用使用亚马逊托管区块链 (AMB) 查询查询区块链 AWS 管理控制台

您可以使用 Amazon Managed Blockchain (AMB) 查询，并在支持的区块链网络上 AWS 管理控制台进行查询。以下步骤显示了如何执行此操作：

1. 打开亚马逊区块链管理控制台，网址为https://console.aws.amazon.com/managedblockchain/。
2. 从 “查询” 部分中选择 “查询编辑器”。
3. 从支持的区块链网络中选择一个。
4. 选择要运行的查询类型。
5. 输入所选查询类型的相关参数，然后运行查询。

AMB Query 将运行您的查询，您将在查询结果窗口中看到结果。

亚马逊托管区块链 (AMB) 查询入门

使用本节中的 step-by-step教程来学习如何使用亚马逊托管区块链 (AMB) 查询来执行任务。这些过程需要一些先决条件。如果您不熟悉 AMB Query，可以查看本指南的设置部分。有关更多信息，请参阅设置亚马逊托管区块链 (AMB) 查询。

> **Note**
>
> 这些示例中的一些变量是故意混淆的。在运行这些示例之前，请将它们替换为您自己的有效版本。

主题

- 创建用于访问 AMB 查询 API 操作的 IAM 策略
- 使用 Go 发出亚马逊托管区块链 (AMB) 查询 API 请求
- 使用 Node.js 发出亚马逊托管区块链 (AMB) 查询 API 请求
- 使用 Python 发出亚马逊托管区块链 (AMB) 查询 API 请求
- 使用上的 Amazon Managed Blockchain (AMB) 查询 AWS 管理控制台 来运行操作 GetTokenBalance

创建用于访问 AMB 查询 API 操作的 IAM 策略

要发出 AMB 查询 API 请求，您必须使用对亚马逊托管区块链 (AMB) 查询具有相应 IAM 权限的用户证书（AWS_ACCESS_KEY_ID 和 AWS_SECRET _ACCESS_KEY）。在 AWS CLI 安装了的终端中，运行以下命令创建用于访问 AMB Query API 操作的 IAM 策略：

```
cat <<EOT > ~/amb-query-access-policy.json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid" : "AMBQueryAccessPolicy",
            "Effect": "Allow",
            "Action": [
                "managedblockchain-query:*"
            ],
            "Resource": "*"
```

```
        }
    ]
}
EOT
aws iam create-policy --policy-name AmazonManagedBlockchainQueryAccess --policy-
document file://$HOME/amb-query-access-policy.json
```

创建策略后，将该策略附加到 IAM 用户的角色以使其生效。在中 AWS 管理控制台，导航到 IAM 服务，并将策略附加AmazonManagedBlockchainQueryAccess到分配给将使用该服务的 IAM 用户的角色。有关更多信息，请参阅创建角色并分配给 IAM 用户。

> **Note**
>
> AWS 建议您授予对特定 API 操作的访问权限，而不是使用通配符。*有关更多信息，请参阅访问特定的亚马逊托管区块链 (AMB) 查询 API 操作。

使用 Go 发出亚马逊托管区块链 (AMB) 查询 API 请求

借助 Amazon Managed Blockchain (AMB) 查询，即使区块链数据尚未最终确定，您也可以构建依赖于即时访问区块链数据的应用程序。AMB Query 支持多种用例，例如填充钱包的交易历史记录、根据交易哈希提供有关交易的上下文信息，或者获取原生代币以及 ERC-721、ERC-1155 和 ERC-20 代币的余额。

以下示例使用 Go 语言创建，并使用 AMB 查询 API 操作。有关 Go 的更多信息，请参阅 Go 文档。有关 AMB 查询 API 的更多信息，请参阅亚马逊托管区块链 (AMB) 查询 API 参考文档。

以下示例使用ListTransactions和 GetTransaction API 操作首先获取以太坊主网上给定外部拥有地址 (EOA) 的所有交易的列表，然后下一个示例从列表中检索单笔交易的交易详细信息。

Example— 使用 Go 进行 ListTransactions API 操作

将以下代码复制到ListTransactions目录listTransactions.go中名为的文件中。

```
package main

import (
    "fmt"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/managedblockchainquery"
```

```
    "time"
)

func main() {

    // Set up a session
    ambQuerySession := session.Must(session.NewSessionWithOptions(session.Options{
        Config: aws.Config{
            Region: aws.String("us-east-1"),
        },
    }))
    client := managedblockchainquery.New(ambQuerySession)

    // Inputs for ListTransactions API
    ownerAddress := "0x00000bf26964af9d7eed9e03e53415d********"
    network := managedblockchainquery.QueryNetworkEthereumMainnet
    sortOrder := managedblockchainquery.SortOrderAscending
    fromTime := time.Date(1971, 1, 1, 1, 1, 1, 1, time.UTC)
    toTime := time.Now()
    nonFinal := "NONFINAL"
    // Call ListTransactions API. Transactions that have reached finality are always
 returned
    listTransactionRequest, listTransactionResponse :=
 client.ListTransactionsRequest(&managedblockchainquery.ListTransactionsInput{
        Address: &ownerAddress,
        Network: &network,
        Sort: &managedblockchainquery.ListTransactionsSort{
            SortOrder: &sortOrder,
        },
        FromBlockchainInstant: &managedblockchainquery.BlockchainInstant{
            Time: &fromTime,
        },
        ToBlockchainInstant: &managedblockchainquery.BlockchainInstant{
            Time: &toTime,
        },

        ConfirmationStatusFilter: &managedblockchainquery.ConfirmationStatusFilter{
            Include: []*string{&nonFinal},
          },
    })
    errors := listTransactionRequest.Send()

    if errors == nil {
        // handle API response
```

```
        fmt.Println(listTransactionResponse)
    } else {
        // handle API errors
        fmt.Println(errors)
    }
}
```

保存文件后，在ListTransactions目录中使用以下命令运行代码：`go run listTransactions.go`。

以下输出类似于以下内容：

```
{
  Transactions: [
    {
      ConfirmationStatus: "FINAL",
      Network: "ETHEREUM_MAINNET",
      TransactionHash:
 "0x12345ea404b45323c0cf458ac755ecc45985fbf2b18e2996af3c8e8693354321",
      TransactionTimestamp: 2020-06-01 01:59:11 +0000 UTC
    },
    {
      ConfirmationStatus: "FINAL",
      Network: "ETHEREUM_MAINNET",
      TransactionHash:
 "0x1234547c65675d867ebd2935bb7ebe0996e9ec8e432a579a4516c7113bf54321",
      TransactionTimestamp: 2021-09-01 20:06:59 +0000 UTC
    },
     {
      ConfirmationStatus: "NONFINAL",
      Network: "ETHEREUM_MAINNET",
      TransactionHash:
 "0x123459df7c1cd42336cd1c444cae0eb660ccf13ef3a159f05061232a24954321",
      TransactionTimestamp: 2024-01-23 17:10:11 +0000 UTC
    }
  ]
}
```

Example— 使用 Go 进行 GetTransaction API 操作

此示例使用先前输出中的交易哈希。将以下代码复制到GetTransaction目录`GetTransaction.go`中名为的文件中。

```
package main
```

```
import (
    "fmt"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/managedblockchainquery"
)

func main() {

    // Set up a session
    ambQuerySession := session.Must(session.NewSessionWithOptions(session.Options{
        Config: aws.Config{
            Region: aws.String("us-east-1"),
        },
    }))
    client := managedblockchainquery.New(ambQuerySession)

    // inputs for GetTransaction API
    transactionHash :=
 "0x123452695a82868950d9db8f64dfb2f6f0ad79284a6c461d115ede8930754321"
    network := managedblockchainquery.QueryNetworkEthereumMainnet

    // Call GetTransaction API. This operation will return transaction details for all
    // transactions that are con#rmed on the blockchain, even if they have not
    // reached #nality.
    getTransactionRequest, getTransactionResponse :=
 client.GetTransactionRequest(&managedblockchainquery.GetTransactionInput{
        Network:         &network,
        TransactionHash: &transactionHash,
    })

    errors := getTransactionRequest.Send()
    if errors == nil {
        // handle API response
        fmt.Println(getTransactionResponse)
    } else {
        // handle API errors
        fmt.Println(errors)
    }
}
```

保存文件后，在GetTransaction目录中使用以下命令运行代码：`go run GetTransaction.go`。

以下输出类似于以下内容：

```
{
  Transaction: {
    BlockHash: "0x000005c6a71d1afbc005a652b6ceca71cd516d97b0fc514c2a1d0f2ca3912345",
    BlockNumber: "11111111",
    CumulativeGasUsed: "5555555",
    EffectiveGasPrice: "44444444444",
    From: "0x9157f4de39ab4c657ad22b9f19997536********",
    GasUsed: "22222",
    Network: "ETHEREUM_MAINNET",
    NumberOfTransactions: 111,
    SignatureR: "0x99999894fd2df2d039b3555dab80df66753f84be475069dfaf6c6103********",
    SignatureS: "0x77777a101e7f37dd2dd0bf878b39080d5ecf3bf082c9bd4f40de783e********",
    SignatureV: 0,
    ConfirmationStatus: "FINAL",
    ExecutionStatus: "SUCCEEDED",
    To: "0x5555564f282bf135d62168c1e513280d********",
    TransactionHash:
 "0x123452695a82868950d9db8f64dfb2f6f0ad79284a6c461d115ede8930754321",
    TransactionIndex: 11,
    TransactionTimestamp: 2022-02-02 01:01:59 +0000 UTC
  }
}
```

该 GetTokenBalance API为您提供了一种获取原生代币（ETH和BTC）余额的方法，该余额可用于获取某个时间点外部拥有的账户（EOA）的当前余额。

Example— 使用 GetTokenBalance API 操作获取 Go 中原生代币的余额

在以下示例中，您使用 GetTokenBalance API 在以太坊主网上获取地址以太坊 (ETH) 余额。将以下代码复制到GetTokenBalance目录GetTokenBalanceEth.go中名为的文件中。

```
package main

import (
    "fmt"
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/managedblockchainquery"
)

func main() {
```

```
    // Set up a session
    ambQuerySession := session.Must(session.NewSessionWithOptions(session.Options{
        Config: aws.Config{
            Region: aws.String("us-east-1"),
        },
    }))
    client := managedblockchainquery.New(ambQuerySession)

    // inputs for GetTokenBalance API
    ownerAddress := "0xBeE510AF9804F3B459C0419826b6f225********"
    network := managedblockchainquery.QueryNetworkEthereumMainnet
    nativeTokenId  := "eth" //Ether on Ethereum mainnet

    // call GetTokenBalance API
    getTokenBalanceRequest, getTokenBalanceResponse :=
 client.GetTokenBalanceRequest(&managedblockchainquery.GetTokenBalanceInput{
        TokenIdentifier: &managedblockchainquery.TokenIdentifier{
            Network:         &network,
            TokenId: &nativeTokenId,
        },
        OwnerIdentifier: &managedblockchainquery.OwnerIdentifier{
            Address: &ownerAddress,
        },
    })
    errors := getTokenBalanceRequest.Send()

    if errors == nil {
        // process API response
        fmt.Println(getTokenBalanceResponse)
    } else {
        // process API errors
        fmt.Println(errors)
    }
}
```

保存文件后，在GetTokenBalance目录中使用以下命令运行代码：`go run GetTokenBalanceEth.go`。

以下输出类似于以下内容：

```
{
  AtBlockchainInstant: {
    Time: 2020-12-05 11:51:01 +0000 UTC
```

```
  },
  Balance: "4343260710",
  LastTransactionHash:
 "0x00000ce94398e56641888f94a7d586d51664eb9271bf2b3c48297a50a0711111",
  LastTransactionTime: 2023-03-14 18:33:59 +0000 UTC,
  OwnerIdentifier: {
    Address: "0x12345d31750D727E6A3a7B534255BADd********"
  },
  TokenIdentifier: {
    Network: "ETHEREUM_MAINNET",
    TokenId: "eth"
  }
}
```

使用 Node.js 发出亚马逊托管区块链 (AMB) 查询 API 请求

要运行这些 Node 示例，需要满足以下先决条件：

1. 您的计算机上必须安装节点版本管理器 (nvm) 和 Node.js。您可以在此处找到操作系统的安装说明。
2. 使用`node --version`命令并确认您使用的是 Node 版本 14 或更高版本。如果需要，您可以使用`nvm install 14`命令和`nvm use 14`命令安装版本 14。
3. 环境变量`AWS_ACCESS_KEY_ID`和`AWS_SECRET_ACCESS_KEY`必须包含与账户关联的凭证。

   使用以下命令将这些变量作为字符串导出到客户端。将以下突出显示的值替换为 IAM 用户账户中的相应值。

   ```
   export AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
   export AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
   ```

> **Note**
>
> - 完成所有先决条件后，您可以通过 HTTPS 提交签名的请求以访问亚马逊托管区块链 (AMB) 查询 API 操作并使用 Node.js 中的原生 https 模块提出请求，也可以使用 AXIO S 等第三方库并从 AMB 查询中检索数据。
> - 这些示例使用第三方 HTTP 客户端 Node.js，但您也可以使用 AWS JavaScript 软件开发工具包向 AMB Query 发出请求。
> - 以下示例向您展示了如何使用 Axios 和 Sigv4 的 AWS SDK 模块发出 AMB 查询 API 请求。

将以下package.json文件复制到本地环境的工作目录中：

```
{
  "name": "amb-query-examples",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@aws-crypto/sha256-js": "^4.0.0",
    "@aws-sdk/credential-provider-node": "^3.360.0",
    "@aws-sdk/protocol-http": "^3.357.0",
    "@aws-sdk/signature-v4": "^3.357.0",
    "axios": "^1.4.0"
  }
}
```

Example— 使用 AMB 查询 API 从特定的外部所有地址 (EOA) 检索历史代币余额 GetTokenBalance

您可以使用 GetTokenBalance API 获取各种代币（例如 ERC20 ERC721、和 ERC1155）和原生币（例如，ETH 和 BTC）的余额，您可以使用这些余额根据历史timestamp（Unix 时间戳-秒）获取外部拥有的账户 (EOA) 的当前余额。在此示例中，您使用 GetTokenBalanceAPI 在以太坊主网上获取 ERC20 代币 USDC 的地址余额。

要测试 GetTokenBalance API，请将以下代码复制到名为的文件中token-balance.js，然后将该文件保存到同一个工作目录中：

```
const axios = require('axios').default;
const SHA256 = require('@aws-crypto/sha256-js').Sha256
const defaultProvider = require('@aws-sdk/credential-provider-node').defaultProvider
const HttpRequest = require('@aws-sdk/protocol-http').HttpRequest
const SignatureV4 = require('@aws-sdk/signature-v4').SignatureV4

// define a signer object with AWS service name, credentials, and region
const signer = new SignatureV4({
  credentials: defaultProvider(),
  service: 'managedblockchain-query',
  region: 'us-east-1',
  sha256: SHA256,
```

```
});

const queryRequest = async (path, data) => {
  //query endpoint
  let queryEndpoint = `https://managedblockchain-query.us-east-1.amazonaws.com/
${path}`;

  // parse the URL into its component parts (e.g. host, path)
  const url = new URL(queryEndpoint);

  // create an HTTP Request object
  const req = new HttpRequest({
    hostname: url.hostname.toString(),
    path: url.pathname.toString(),
    body: JSON.stringify(data),
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
      'Accept-Encoding': 'gzip',
      host: url.hostname,
    }
  });


  // use AWS SignatureV4 utility to sign the request, extract headers and body
  const signedRequest = await signer.sign(req, { signingDate: new Date() });

  try {
    //make the request using axios
    const response = await axios({...signedRequest, url: queryEndpoint, data: data})

    console.log(response.data)
  } catch (error) {
    console.error('Something went wrong: ', error)
    throw error
  }


}


let methodArg = 'get-token-balance';

let dataArg = {
```

```
  " atBlockchainInstant": {
    "time": 1688071493
  },
  "ownerIdentifier": {
      "address": "0xf3B0073E3a7F747C7A38B36B805247B2********" // externally owned
 address
  },
  "tokenIdentifier": {
      "contractAddress":"0xA0b86991c6218b36c1d19D4a2e9Eb0cE********", //USDC contract
 address
      "network":"ETHEREUM_MAINNET"
  }
}

//Run the query request.
queryRequest(methodArg, dataArg);
```

要运行代码，请在文件所在目录下打开终端，然后运行以下命令：

```
npm i
node token-balance.js
```

此命令运行脚本，传入代码中定义的参数，以请求以太坊主网上列出的EOA的 ERC20 USDC余额。响应类似于以下内容：

```
{
 atBlockchainInstant: { time: 1688076218 },
 balance: '140386693440144',
 lastUpdatedTime: { time: 1688074727 },
 ownerIdentifier: { address: '0xf3b0073e3a7f747c7a38b36b805247b2********' },
 tokenIdentifier: {
   contractAddress: '0xa0b86991c6218b36c1d19d4a2e9eb0ce********',
   network: 'ETHEREUM_MAINNET'
 }
```

使用 Python 发出亚马逊托管区块链 (AMB) 查询 API 请求

要运行这些 Python 示例，需要满足以下先决条件：

1. 你的计算机上必须安装了 Python。您可以在此处找到操作系统的安装说明。
2. 安装适用于 Python 的 AWS 开发工具包 (Boto3)。

3. 安装AWS 命令行界面并运行命令aws configure为Access Key IDSecret Access Key、和设置变量Region。

完成所有先决条件后，您可以通过 HTTPS 使用 AWS 适用于 Python 的软件开发工具包来发出亚马逊托管区块链 (AMB) 查询 API 请求。

以下 Python 示例使用 boto3 中的模块向 AMB 查询 API 操作发送附有必要 Sigv4 标头的请求。ListTransactionEvents此示例检索以太坊主网上给定交易发出的事件列表。

将以下list-transaction-events.py文件复制到本地环境的工作目录中：

```
import json
from botocore.auth import SigV4Auth
from botocore.awsrequest import AWSRequest
from botocore.session import Session
from botocore.httpsession import URLLib3Session

def signed_request(url, method,  params, service, region):

    session = Session()
    sigv4 = SigV4Auth(session.get_credentials(), service, region)
    data = json.dumps(params)
    request = AWSRequest(method, url, data=data)
    sigv4.add_auth(request)
    http_session = URLLib3Session()
    response = http_session.send(request.prepare())

    return(response)

url = 'https://managedblockchain-query.us-east-1.amazonaws.com/list-transaction-events'
method = 'POST'
params = {
'network': 'ETHEREUM_MAINNET',
'transactionHash': '0x125714bb4db48757007fff2671b37637bbfd6d47b3a4757ebbd0c5222984f905'
}
service = 'managedblockchain-query'
region = 'us-east-1'

# Call the listTransactionEvents operation. This operation will return transaction
 details for
# all transactions that are con#rmed on the blockchain, even if they have not reached
# #nality.
```

```
listTransactionEvents = signed_request(url, method, params, service, region)

print(json.loads(listTransactionEvents.content.decode('utf-8')))
```

要将示例代码运行到ListTransactionEvents，请将文件保存在工作目录中，然后运行该命令python3 list-transaction-events.py。此命令运行脚本，传入代码中定义的参数，以请求以太坊主网上与给定交易哈希相关的事件。响应类似于以下内容：

```
{
 'events':
 [
  {
      'contractAddress': '0x95ad61b0a150d79219dcf64e1e6cc01f********',
      'eventType': 'ERC20_TRANSFER',
      'from': '0xab5801a7d398351b8be11c439e05c5b3********',
      'network': 'ETHEREUM_MAINNET',
      'to': '0xdead000000000000000420694206942********',
      'transactionHash':
 '0x125714bb4db48757007fff2671b37637bbfd6d47b3a4757ebbd0c522********',
      'value': '4102419967718718947718261747554 64'
  }
 ]
}
```

使用上的 Amazon Managed Blockchain (AMB) 查询 AWS 管理控制台 来运行操作 GetTokenBalance

以下示例展示了如何使用亚马逊托管区块链 (AMB) 查询在以太坊主网上获取代币余额 AWS 管理控制台

Example

1. 打开亚马逊区块链管理控制台，网址为https://console.aws.amazon.com/managedblockchain/。
2. 从 “查询” 部分中选择 “查询编辑器”。
3. 选择以太坊主网作为区块链网络。
4. 选择GetTokenBalance作为查询类型。
5. 输入代币的区块链地址。
6. 输入代币的合约地址。

7. 输入令牌的可选令牌 ID。
8. 选择代币余额的截止日期。
9. 为代币余额输入可选的 A t tim e。
10. 选择运行查询。

AMB Query 将运行您的查询，您将在查询结果窗口中看到结果。

亚马逊托管区块链 (AMB) 查询用例

本主题提供了 AMB Query 用例列表。

主题

查询当前和历史代币余额

GetTokenBalanceAPI 使用外部拥有的账户的通用时间戳（Unix 时间戳ERC20， ERC721以秒为单位）获取支持的代币（、、 ）和原生硬币（ETH、BTC）的余额，以获取当前或历史余额（ ）。ERC1155 EOAs例如，您可以使用 GetTokenBalance API 操作在以太坊主网上获取 ERC20 代币 USDC 的地址余额。您还可以使用 BatchGetTokenBalance API 操作批量检索代币和原生币的余额。

有关更多信息，请参阅亚马逊托管区块链 (AMB) 查询参考指南。

检索历史交易数据

借助 Amazon Managed Blockchain (AMB) 查询，您可以从以太坊和比特币等公共区块链中检索历史数据。此功能支持多种用例，例如检索区块链钱包上的交易历史记录或根据交易哈希提供有关交易的上下文信息。您可以使用 ListTransactionsAPI 操作在以太坊主网上获取给定外部拥有地址 (EOA) 的交易列表，然后可以使用 GetTransactionAPI 操作从列表中检索单笔交易的交易详细信息。

有关更多信息，请参阅亚马逊托管区块链 (AMB) 查询参考指南。

获取给定地址的所有代币余额

您可以使用 ListTokenBalancesAPI 操作来获取钱包、用户界面、web3 实用程序等的余额。此 API 操作使用单个 API 操作返回给定公共区块链上代币（ERC20 ERC721、、 ERC1155）和原生币

（ETH、BTC）的所有余额列表。例如，您可以提供外部拥有的地址 (EOA) 和网络（以太坊主网），并且可以在响应中收到代币和原生币余额的列表。

有关更多信息，请参阅亚马逊托管区块链 (AMB) 查询参考指南。

列出为交易发出的事件

您可以使用 ListTransactionEventsAPI 操作来检索因给定交易而发出的合约事件列表，这些事件由其哈希（交易标识符）标识。例如，您可以使用ListTransactionEvents检索调用以太坊区块链上 ERC20 代币合约函数的交易的结果事件，例如合 ERC20 约中的转账事件或提款事件。

有关更多信息，请参阅亚马逊托管区块链 (AMB) 查询参考指南。

获取合约铸造的所有代币

当传递合约地址作为输入时，您可以使用 ListTokenBalancesAPI 操作返回合约铸造的所有支持的代币（ERC20 ERC721、、 ERC1155）的列表。例如，您可以使用 API 操作检索与以太坊区块链上 ERC721 合约标准铸造的不可替代代币 (NFTs) 相关的信息。ListTokenBalances

有关更多信息，请参阅亚马逊托管区块链 (AMB) 查询参考指南。

列出合约并获取合同信息

您可以使用 ListAssetContractsAPI 操作列出由给定地址部署的 ERC-721、ERC-1155 或 ERC-20 合约。此外，如果您有合约地址，则可以使用 GetAssetContractAPI 操作来检索合约的属性，例如合约类型部署者地址和相关的代币元数据。

有关更多信息，请参阅亚马逊托管区块链 (AMB) 查询参考指南。

亚马逊托管区块链 (AMB) 查询 API 参考

亚马逊托管区块链 (AMB) 查询提供用于查询支持的区块链的 API 操作。这包括 APIs 查询代币、交易和合约。有关更多信息，请参阅 AMB 查询 API 参考。

亚马逊托管区块链 (AMB) 查询中的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方 AWS 的共同责任。责任共担模型将其描述为既是云的安全性，也是云端的安全：

- 云安全 — AWS 负责保护在云中运行 AWS 服务的基础架构 AWS 云。 AWS 还为您提供可以安全使用的服务。作为 AWS 合规性计划的一部分，第三方审核人员将定期测试和验证安全性的有效性。要了解适用于亚马逊托管区块链 (AMB) 查询的合规计划，请参阅合规计划范围内的AWS 服务。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

为了提供数据保护、身份验证和访问控制，Amazon Managed Blockchain 使用了在托管区块链中运行的开源框架的功能和 AWS 功能。

本文档可帮助您了解在使用 AMB Query 时如何应用责任共担模型。以下主题向您介绍如何配置 AMB Query 以满足您的安全和合规性目标。您还可以学习如何使用其他 AWS 服务来帮助您监控和保护您的 AMB Query 资源。

主题

数据加密

数据加密有助于防止未经授权的用户从区块链网络和相关的数据存储系统读取数据。这包括在网络中传输时可能被拦截的数据，即传输中的数据。

传输中加密

默认情况下，托管区块链使用 HTTPS/TLS 连接来加密从 AWS CLI 客户端传输到 AWS 服务端点的所有数据。

亚马逊托管区块链 (AMB) 查询的身份和访问管理

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（有权限）使用 AMB Query 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

受众

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅亚马逊托管区块链 (AMB) 疑难解答查询身份和访问权限）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅亚马逊托管区块链 (AMB) 查询如何与 IAM 配合使用）
- IAM 管理员：编写用于管理访问权限的策略（请参阅亚马逊托管区块链 (AMB) 查询的基于身份的策略示例）

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center （例如（IAM Identity Center）、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录 用户指南》中的如何登录您的 AWS 账户。

对于编程访问， AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的AWS 签名版本 4。

AWS 账户 root 用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关需要根用户凭证的任务，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户使用与身份提供商的联合身份验证才能 AWS 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Directory Service ，或者 AWS 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

要集中管理访问权限，建议使用。 AWS IAM Identity Center有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

IAM 用户和群组

IAM 用户是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问。

IAM 组指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户使用案例。

IAM 角色

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon EC2 上运行的应用程序非常有用。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略定义了与身份或资源关联时的权限。 AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。IAM 策略定义权限，与执行操作所用的方法无关。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。您必须在基于资源的策略中指定主体。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

其他策略类型

AWS 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，请参阅《 IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs)-在中指定组织或组织单位的最大权限 AWS Organizations。有关更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略 – 在为角色或联合用户创建临时会话时，作为参数传递的高级策略。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

亚马逊托管区块链 (AMB) 查询如何与 IAM 配合使用

在使用 IAM 管理对 AMB 查询的访问权限之前，请先了解有哪些 IAM 功能可用于 AMB 查询。

您可以在亚马逊托管区块链 (AMB) 查询中使用的 IAM 功能

IAM 功能	AMB 查询支持
基于身份的策略	是
基于资源的策略	否
策略操作	是
策略资源	否
策略条件密钥	否
ACLs	否
ABAC（策略中的标签）	否
临时凭证	是
主体权限	是
服务角色	否
服务关联角色	否

要全面了解 AMB Query 和其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的AWS 服务。

AMB 查询的基于身份的策略

支持基于身份的策略：是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

AMB 查询的基于身份的策略示例

要查看 AMB Query 基于身份的策略示例，请参阅。亚马逊托管区块链 (AMB) 查询的基于身份的策略示例

AMB Query 中基于资源的策略

支持基于资源的策略：否

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

AMB 查询的策略操作

支持策略操作：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。在策略中包含操作以授予执行关联操作的权限。

要查看 AMB 查询操作列表，请参阅《服务授权参考》中的 Amazon Managed Blockchain (AMB) 查询定义的操作。

AMB Query 中的策略操作在操作前使用以下前缀：

```
managedblockchain-query:
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "managedblockchain-query::ListTransaction",
      "managedblockchain-query::GetTransaction"
         ]
```

要查看 AMB Query 基于身份的策略示例，请参阅。亚马逊托管区块链 (AMB) 查询的基于身份的策略示例

AMB 查询的策略资源

支持策略资源：否

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

要查看 AMB 查询资源类型及其列表 ARNs，请参阅《服务授权参考》中的 Amazon Managed Blockchain (AMB) 查询定义的资源。要了解您可以使用哪些操作来指定每种资源的 ARN，请参阅 Amazon Managed Blockchain (AMB) 查询定义的操作。

要查看 AMB Query 基于身份的策略示例，请参阅。亚马逊托管区块链 (AMB) 查询的基于身份的策略示例

AMB 查询的策略条件密钥

支持特定于服务的策略条件键：否

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

Condition 元素根据定义的条件指定语句何时执行。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

要查看 AMB 查询条件键列表，请参阅《服务授权参考》中的 Amazon Managed Blockchain (AMB) 查询的条件密钥。要了解您可以使用条件键的操作和资源，请参阅 Amazon Managed Blockchain (AMB) 查询定义的操作。

要查看 AMB Query 基于身份的策略示例，请参阅。亚马逊托管区块链 (AMB) 查询的基于身份的策略示例

ACLs 在 AMB 查询中

支持 ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

带有 AMB 查询功能的 ABAC

支持 ABAC（策略中的标签）：否

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于称为标签的属性来定义权限。您可以将标签附加到 IAM 实体和 AWS 资源，然后设计 ABAC 策略以允许在委托人的标签与资源上的标签匹配时进行操作。

要基于标签控制访问，您需要使用 aws:ResourceTag/*key-name*aws:RequestTag/*key-name* 或 aws:TagKeys 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的使用 ABAC 授权定义权限。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

在 AMB 查询中使用临时证书

支持临时凭证：是

临时证书提供对 AWS 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的临时安全凭证和使用 IAM 的。AWS 服务

AMB 查询的跨服务主体权限

支持转发访问会话（FAS）：是

转发访问会话 (FAS) 使用调用主体的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

AMB 查询的服务角色

支持服务角色：否

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会中断 AMB 查询功能。仅当 AMB Query 提供相关指导时才编辑服务角色。

AMB 查询的服务相关角色

支持服务相关角色：否

服务相关角色是一种链接到的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务关联角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅能够与 IAM 搭配使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

亚马逊托管区块链 (AMB) 查询的基于身份的策略示例

默认情况下，用户和角色无权创建或修改 AMB Query 资源。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略（控制台）。

有关 AMB Query 定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《 ARNs 服务授权参考》中的 Amazon Managed Blockchain (AMB) 查询的操作、资源和条件密钥。

主题

策略最佳实践

基于身份的策略决定了某人是否可以在您的账户中创建、访问或删除 AMB Query 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的AWS 托管策略或工作职能的AWS 托管策略。
- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性：IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言（JSON）和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 验证策略。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

访问特定的亚马逊托管区块链 (AMB) 查询 API 操作

> **Note**
>
> 要访问 AMB 查询以发出 API 调用，您需要具有 AMB 查询相应 IAM 权限的用户证书（AWS_ACCESS_KEY_ID和AWS_SECRET_ACCESS_KEY）。

Example访问所有亚马逊托管区块链 (AMB) 查询的 IAM 政策 APIs

此示例授予您中的 IAM 用户 AWS 账户 访问所有 AMB 查询 APIs的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AccessAllAMBQueryAPIs",
            "Effect": "Allow",
            "Action": [
                "managedblockchain-query:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Example访问亚马逊托管区块链 (AMB) 查询ListTransactions的 IAM 政策和 GetTransaction APIs

此示例授予您中的 IAM 用户 AWS 账户 访问 AMB 查询`ListTransaction`的权限，以及 `GetTransaction` APIs

> **Note**
>
> 您可以将示例 APIs 中的替换或添加为其他 APIs ，以允许访问其他或更多 APIs。有关 AMB 查询的列表 APIs，请参阅亚马逊托管区块链 (AMB) 查询 API 参考指南。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AccessAMBQueryAPIs",
            "Effect": "Allow",
```

```
            "Action": [
                "managedblockchain-query:ListTransactions",
                "managedblockchain-query:GetTransaction"
            ],
            "Resource": "*"
        }
    ]
}
```

亚马逊托管区块链 (AMB) 疑难解答查询身份和访问权限

使用以下信息来帮助您诊断和修复在使用 AMB Query 和 IAM 时可能遇到的常见问题。

主题

- 我无权在 AMB Query 中执行操作

我无权在 AMB Query 中执行操作

如果您收到错误提示，指明您无权执行某个操作，则必须更新策略以允许执行该操作。

当 mateojackson IAM 用户尝试使用控制台查看有关虚构 *my-example-widget* 资源的详细信息，但不拥有虚构 managedblockchain-query::*GetWidget* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 managedblockchain-query::GetWidget on resource: my-example-widget
```

在此情况下，必须更新 mateojackson 用户的策略，以允许使用 managedblockchain-query::*GetWidget* 操作访问 *my-example-widget* 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

亚马逊托管区块链 (AMB) 在亚马逊上查询 API 使用指标 CloudWatch

亚马逊上的 API 使用率指标 CloudWatch

发布的 API 使用指标与亚马逊托管区块链 (AMB) 查询服务配额 CloudWatch 相对应。您可以配置警报，以便在使用量接近服务配额时提醒您。有关与服务配额 CloudWatch集成的更多信息，请参阅 Amazon CloudWatch 用户指南中的 AWS 使用量指标。

AMB Query 在AWS/Usage命名空间中发布以下 API 指标以及Amazon Managed Blockchain Query服务名称。

指标	说明
CallCount	在 AMB 查询中对某个 API 发出的调用总数。SUM 表示在指定时间段内对 API 的调用总数。

Amazon Managed Blockchain (AMB) 查询将使用量指标发布到AWS/Usage命名空间，其维度如下。

维度	描述
服务	包含资源的 AWS 服务的名称。 Amazon Managed Blockchain Query 将始终是该维度的值。
Type	被举报的实体的类型。 API将始终是该维度的值。
资源	要报告的资源类型。所使用的 AMB 查询 API 操作的名称将是该维度的值。
类	正在报告的资源的类别。 None将始终是该维度的值。

AMB 查询用户指南的文档历史记录

下表介绍了 AMB Query 的文档版本。

变更	说明	日期
AMB Query 支持比特币交易标识符和交易哈希	对于比特币网络，AMB Query API 操作同时支持交易标识符 (`transactionId`) 和交易哈希 (`transactionHash`)。	2024 年 3 月 21 日
支持 Amazon 上的 API 使用量指标 CloudWatch	AMB Query 在上添加了对 API 使用量指标的 CloudWatch支持。这些使用量指标与 AMB Query 服务配额相对应。	2024 年 2 月 8 日
Support 支持尚未完成的交易	AMB Query 增加了对尚未完成的交易的支持。它还会从`GetTransaction` 操作的响应中删除对该`status`属性的支持。相反，您将使用`confirmationStatus` 和`executionStatus` 属性来确定事务的状态。	2024 年 2 月 1 日
已在交易数据类型中弃用该`status`属性	Amazon Managed Blockchain (AMB) 查询已弃用交易数据类型中的`status`属性。必须使用`confirmationStatus` 和`executionStatus` 字段来确定交易`status`的是否为`FINAL`或`FAILED`。	2023 年 12 月 20 日
对 Sepolia 测试网的支持	Amazon Managed Blockchain (AMB) 查询现在支持在以太坊 Sepolia 测试网上进行查询。	2023 年 10 月 19 日

Support 对资产合约的支持	您可以使用 ListAssetContracts API 操作列出按给定地址部署的部署。此外，如果您有合同地址，则可以使用 GetAssetContract API 操作来检索合约的详细信息。	2023 年 10 月 16 日
Support 对比特币测试网的支持	Amazon Managed Blockchain (AMB) 查询现在支持在比特币测试网上查询。	2023 年 10 月 16 日
初始版本	AMB 查询服务的初始版本。	2023 年 7 月 27 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。