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 (NFT) 的历史代币余额，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 (NFT) 的标准。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 API 时，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 Amazon Managed Blockchain (AMB) 查询之前，请按照本节中的步骤创建 AWS account。以下部分讨论如何开始使用 AMB Query。

先决条件和注意事项

在首次使用 Amazon Web Services 之前，您必须拥有 AWS account。

报名参加 AWS

当您注册亚马逊 Web Services 时 (AWS)，你的 AWS 账户已自动注册为所有人 AWS 服务，包括 Amazon Managed Blockchain (AMB) 查询。您只需为使用的服务付费。

如果你有 AWS 账户 现在，请转到下一步。如果你没有 AWS 账户，请按以下步骤创建一个。

要创建 AWS account

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

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当你报名参加 AWS 账户，一个 AWS 账户根用户已创建。root 用户可以访问所有内容 AWS 服务以及账户中的资源。作为安全最佳实践，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，必须创建一个 AWS Identity and Access Management (IAM) 拥有允许必要托管区块链操作权限的委托人（用户或群组）。

只有IAM委托人才能提出AMB查询API请求。在调用 AMB Query 时APIs，您可以通过使用签名版本 4 签名过程进行身份验证的HTTPS连接进行调用。这意味着只有授权的IAM委托人在 AWS 账户可以拨API打AMB查询电话。为此，请执行以下操作： AWS 呼叫时必须提供证书（访问密钥 ID 和私有访问密钥）。

有关如何创建IAM用户的信息，请参阅在中创建IAM用户 AWS 账户。有关如何向用户附加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Amazon Managed Blockchain () AMB 查询的基于身份的策略示例。

安装和配置 AWS Command Line Interface

如果您还没有这样做，请安装最新的 AWS 要使用的命令行接口 (CLI) AWS 来自终端的资源。有关更多信息，请参阅安装或更新最新版本的 AWS CLI.

> **Note**
>
> 要进行CLI访问，您需要一个访问密钥 ID 和一个私有访问密钥。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，请参阅将临时证书与配合使用 AWS 《IAM用户指南》中的资源。

使用 AWS Management Console 使用 Amazon Managed Blockchain (AMB) 查询查询区块链

你可以使用 Amazon Managed Blockchain (AMB) 在支持的区块链网络上进行查询和查询 AWS Management Console。 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。

AMBQuery 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 Amazon Managed Blockchain (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 (KEY) 查询具有相应IAM权限的用户证书（ AWS_ACCESSAWS_SECRETACCESS_ KEY _ID 和 _AMB ）。在航站楼里有 AWS CLI 安装后，运行以下命令来创建访问AMB查询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 AWS Management Console，导航到该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 建议您授予访问特定API操作的权限，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 Amazon Managed Blockchain (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。AMBQuery 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-20代币的余额。ERC

以下示例是用 Go 语言创建的，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查询的更多信息API，请参阅 Amazon Managed Blockchain (AMB) 查询API参考文档。

以下示例使用ListTransactions和GetTransactionAPI操作首先获取以太坊主网上给定外部拥有的地址 (EOA) 的所有交易的列表，然后下一个示例从列表中检索单个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
  }
}
```

为您GetTokenBalanceAPI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 — 在 Go 中使用**GetTokenBalance**API操作获取原生代币的余额

在以下示例中，您使用GetTokenBalanceAPI来获取以太坊主网上的 Ether (ETH) 地址余额。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，然后使用`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用户帐户中的相应值。

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

> **Note**
>
> - 完成所有先决条件后，您可以提交已签名的请求HTTPS以访问 Amazon Managed Blockchain (AMB) 查询API操作并使用 Node.js 中的原生 https 模块提出请求，也可以使用第三方库（例如查询）AXIOS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 SDK向 AMB Query 提出请求。

- 以下示例向您展示了如何使用 Axios 发出AMB查询API请求以及 AWS SDK适用于 Sigv4 的模块。

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查询从特定外部拥有的地址 (EOA) 检索历史代币余额 **GetTokenBalance** API

您可以使用GetTokenBalanceAPI来获取各种代币（例如 ERC2 0 ERC721、和ERC1155）和原生硬币（例如和）的余额，ETH您可以使用这些余额根据历史（Unix 时间戳-秒EOA）获取外部拥有的账户 timestamp () 的当前余额。BTC在此示例中，您使用在GetTokenBalanceAPI以太坊主网上获取ERC2 0 代币的地址余额。USDC

要测试 GetTokenBalanceAPI，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EOA列出的USDC余额为 ERC2 0。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AWSSDK适用于 Python (Boto3) 的。
3. 安装 AWS 命令行界面并运行命令`aws configure`为您的`Access Key ID`、`Secret Access Key`、和设置变量`Region`。

完成所有先决条件后，可以使用 AWS SDK让 Python 过HTTPS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 中的模块向查询操作发送附有所需 SigV4 标头的请求。AMB `ListTransactionEvents` API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 Amazon Managed Blockchain (AMB) 查询 AWS Management Console 来运行 GetTokenBalance 操作

以下示例展示了如何使用以太坊主网上的 Amazon Managed Blockchain (AMB) 查询来获取代币的余额 AWS Management Console

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 ETHEREUM_ MAINNET 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。

AMBQuery 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 (EOA) 的通用时间戳（Unix 时间戳，以秒为单位）获取支持的代币（ERC20、ERC721、ERC1155）和原生硬币（ETH、BTC）的余额，以获得当前或历史余额。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的代币（ERC20、ERC721、ERC1155）和原生硬币

（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，您可以使用检索在ListTransactionEvents以太坊区块链上调用 ERC20 代币合约函数的交易的结果事件，例如 ERC20 合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 操作在以太坊区块链上检索与 ERC721 合约标准铸造的不可替代代币 (NFT) 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。这包括用于查询代币、交易和合约的 API。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，您可以从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中受益。

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 Cloud。 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 客户端传输到服务端点的所有数据。 AWS

Amazon Managed Blockchain (AMB) 查询的身份和访问管理

AWS Identity and Access Management (IAM) AWS 服务 可以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查询资源。 IAM无需支付额外费用即可使用。 AWS 服务

主题

受众

使用 AWS Identity and Access Management (IAM) 的方式会有所不同，具体取决于你在 AMB Query 中所做的工作。

服务用户-如果您使用AMB查询服务完成工作，则您的管理员会为您提供所需的凭证和权限。当你使用更多的AMB查询功能来完成工作时，你可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 AMB Query 中的功能，请参阅Amazon Managed Blockchain 疑难解答 (AMB) 查询身份和访问权限。

服务管理员-如果您负责公司的 AMB Query 资源，则可能拥有对 AMB Query 的完全访问权限。您的工作是确定您的服务用户应访问哪些AMB查询功能和资源。然后，您必须向 IAM 管理员提交请求，这样才能更改您的服务用户的权限。查看此页面的信息，了解 IAM 的基本概念。要详细了解贵公司如何使用 AMB QueryIAM，请参阅Amazon Managed Blockchain (AMB) 查询的工作原理 IAM。

IAM管理员-如果您是IAM管理员，则可能需要详细了解如何编写策略来管理 AMB Query 的访问权限。要查看可在中使用的基于身份的AMB查询策略示例IAM，请参阅。Amazon Managed Blockchain () 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 AWS 账户根用户、IAM用户身份或通过担任IAM角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （ IAM身份中心）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。在您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM用户指南》中的API请求AWS 签名版本 4。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅用户指南中的多因素身份验证和AWS IAM Identity Center 用户指南IAM中的AWS 多因素身份验证。IAM

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以根用户身份登录的任务的完整列表，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（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C enter 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。 AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 Ident IAM ity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户和应用程序中使用。有关IAM身份中心的信息，请参阅什么是IAM身份中心？ 在《AWS IAM Identity Center 用户指南》中。

IAM 用户和组

IAM用户是您内部 AWS 账户 对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时证书，而不是创建拥有密码和访问密钥等长期凭证的IAM用户。但是，如果您有需要IAM用户长期凭证的特定用例，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组 是指定一个 IAM 用户集合的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可以拥有一个名为的群组，IAMAdmins并授予该群组管理IAM资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅用户指南中的IAMIAM用户用例。

IAM 角色

IAM角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但未与特定人员关联。要在中临时扮IAM演角色 AWS Management Console，可以从用户切换到IAM角色（控制台）。您可以通过调用 AWS CLI 或 AWS API操作或使用自定义操作来代入角色URL。有关使用角色的方法的更多信息，请参阅《IAM用户指南》中的代入角色的方法。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建一个角色，并为该角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关用于联合身份验证的角色的信息，请参阅《IAM用户指南》中的为第三方身份提供商（联合）创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为了控制您的身份在进行身份验证后可以访问的内容，Ident IAM ity Center 会将权限集关联到中的IAM角色。有关权限集的信息，请参阅 AWS IAM Identity Center 用户指南中的 权限集。
- 临时IAM用户权限-IAM 用户或角色可以代入一个IAM角色，为特定任务临时获得不同的权限。
- 跨账户存取 - 您可以使用 IAM 角色允许其他账户中的某个人（可信任主体）访问您账户中的资源。角色是授予跨账户存取权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解角色和基于资源的跨账户访问策略之间的区别，请参阅IAM用户指南IAM中的跨账户资源访问权限。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在服务中拨打电话时，该服务通常会在 Amazon 中运行应用程序EC2或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。

- 转发访问会话 (FAS)-当您使用IAM用户或角色在中执行操作时 AWS，您被视为委托人。当你使用某些服务时，你可能会执行一个操作，然后在不同的服务中启动另一个操作。 FAS使用调用委托人的权限 AWS 服务以及 AWS 服务 向下游服务发出请求的请求。 FAS只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出请求。在这种情况下，您必须具有执行这两个操作的权限。有关提出FAS请求时的政策详情，请参阅转发访问会话。
- 服务角色-服务IAM角色是服务代替您执行操作的角色。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》 AWS 服务中的创建角色以向委派权限。
- 服务相关角色-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相关角色的权限。

- 在 Amazon 上运行的应用程序 EC2 — 您可以使用IAM角色管理在EC2实例上运行并发出 AWS CLI 或 AWS API请求的应用程序的临时证书。这优先于在 EC2 实例中存储访问密钥。要为EC2实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建一个附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅IAM用户指南中的使用IAM角色向在 Amazon EC2 实例上运行的应用程序授予权限。

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。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以JSON文档的 AWS 形式存储在中。有关JSON策略文档结构和内容的更多信息，请参阅《IAM用户指南》中的JSON策略概述。

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。

默认情况下，用户和角色没有权限。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。管理员随后可以向角色添加 IAM 策略，用户可以代入这些角色。

IAM 策略定义操作的权限，无论您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或获取角色信息 AWS API。

基于身份的策略

基于身份的策略是可以附加到身份（例如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托管策略定义自定义IAM权限。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管策略或内联策略之间进行选择，请参阅《IAM用户指南》中的在托管策略和内联策略之间进行选择。

基于资源的策略

基于资源的JSON策略是您附加到资源的策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 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。

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IAM中使用 AWS 托管策略。

访问控制列表 (ACLs)

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。

Amazon S3 AWS WAF、和亚马逊VPC就是支持的服务示例ACLs。要了解更多信息ACLs，请参阅《亚马逊简单存储服务开发者指南》中的访问控制列表 (ACL) 概述。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界-权限边界是一项高级功能，您可以在其中设置基于身份的策略可以向IAM实体（IAM用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，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)-SCPs 是为中的组织或组织单位 (OU) 指定最大权限的JSON策略 AWS Organizations。 AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或

所有帐户。对成员账户中的实体（包括每个实体）的权限进行了SCP限制 AWS 账户根用户。有关 Organization SCPs s 和的更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。

- 资源控制策略 (RCPs) — RCPs 这些JSON策略可用于设置账户中资源的最大可用权限，而无需更新附加到您拥有的每项资源的IAM策略。这会RCP限制成员账户中资源的权限，并可能影响身份（包括身份）的有效权限 AWS 账户根用户，无论这些身份是否属于您的组织。有关 Organizations 的更多信息RCPs，包括 AWS 服务 该支持的列表RCPs，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略 – 会话策略是当您以编程方式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。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。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。

Amazon Managed Blockchain (AMB) 查询的工作原理 IAM

在使用IAM管理 AMB Query 访问权限之前，请先了解 Q AMB uery 中可以使用哪些IAM功能。

IAM你可以在 Amazon Managed Blockchain (AMB) 查询中使用的功能

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

IAM 功能	AMB查询支持
临时凭证	是
主体权限	是
服务角色	否
服务相关角色	否

要全面了解 AMB Query 和其他 AWS 服务如何使用大多数IAM功能，请参阅《IAM用户指南》IAM中与之配合使用的AWS 服务。

基于身份的查询策略 AMB

支持基于身份的策略：是

基于身份的策略是可以附加到身份（例如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托管策略定义自定义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。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可以在JSON策略中使用的所有元素，请参阅IAM用户指南中的IAMJSON策略元素参考。

基于身份的查询策略示例 AMB

要查看基于身份的AMB查询策略的示例，请参阅。Amazon Managed Blockchain () AMB 查询的基于身份的策略示例

Query 中AMB基于资源的策略

支持基于资源的策略：否

基于资源的JSON策略是您附加到资源的策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 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。

要启用跨账户存取，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。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同位置时AWS 账户，可信账户中的IAM管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM用户指南》IAM中的跨账户资源访问权限。

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Action元素描述了可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API操作同名。也有一些例外，例如没有匹配API操作的仅限权限的操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

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。

AMBQuery 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"
         ]
```

要查看基于身份的AMB查询策略的示例，请参阅。Amazon Managed Blockchain () AMB 查询的基于身份的策略示例

用于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。

ResourceJSON策略元素指定要应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。最佳做法是，使用资源的 Amazon 资源名称 (ARN) 来指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），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，请参阅 Amazon Managed Blockchain (AMB) 查询定义的操作。ARN

要查看基于身份的AMB查询策略的示例，请参阅。Amazon Managed Blockchain () 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。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。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。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 AWS 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，仅当 IAM 用户使用其 IAM 用户名进行标记时，您才可为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 策略元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。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查询条件密钥列表，请参阅《服务授权参考》中的 Amazon Managed Blockchain (AMB) 查询的条件密钥。要了解您可以使用哪些操作和资源使用条件密钥，请参阅 Amazon Managed Blockchain (AMB) 查询定义的操作。

要查看基于身份的AMB查询策略的示例，请参阅。Amazon Managed Blockchain () 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。

ABAC使用AMB查询

支持ABAC（策略中的标签）：否

基于属性的访问控制 (ABAC) 是一种基于属性定义权限的授权策略。在中 AWS，这些属性称为标签。您可以将标签附加到IAM实体（用户或角色）和许多 AWS 资源。为实体和资源添加标签是的第一步。ABAC然后，您可以设计ABAC策略，允许在委托人的标签与他们尝试访问的资源上的标签匹配时进行操作。

ABAC在快速增长的环境中很有用，也有助于解决策略管理变得繁琐的情况。

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。

有关更多信息ABAC，请参阅《IAM用户指南》中的使用ABAC授权定义权限。要查看包含设置步骤的教程ABAC，请参阅IAM用户指南中的使用基于属性的访问控制 (ABAC)。

在 AMB Query 中使用临时证书

支持临时凭证：是

当你使用临时证书登录时，有些 AWS 服务 不起作用。有关其他信息，包括哪些 AWS 服务 适用于临时证书 AWS 服务 ，请参阅《IAM用户指南》IAM中的 “适用于临时证书”。

如果您使用除用户名和密码之外的任何方法登录，则 AWS Management Console 使用的是临时证书。例如，当您 AWS 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以

用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《用户指南》中的从IAM用户切换到IAM角色（控制台）。

您可以使用 AWS CLI 或手动创建临时证书 AWS API。然后，您可以使用这些临时证书进行访问 AWS。 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，请参阅中的临时安全证书IAM。

查询的跨服务主体权限 AMB

支持转发访问会话 (FAS)：是

当您使用IAM用户或角色在中执行操作时 AWS，您被视为委托人。当你使用某些服务时，你可能会执行一个操作，然后在不同的服务中启动另一个操作。 FAS使用调用委托人的权限 AWS 服务以及 AWS 服务 向下游服务发出请求的请求。 FAS只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出请求。在这种情况下，您必须具有执行这两个操作的权限。有关提出FAS请求时的政策详情，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代入、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》 AWS 服务中的创建角色以向委派权限。

> **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。

查询的服务相关角色 AMB

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相关角色的权限。

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

Amazon Managed Blockchain () 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。他们也无法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或来执行任务 AWS API。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。管理员随后可以向角色添加 IAM 策略，用户可以代入这些角色。

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

有关AMB查询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
- 允许用户查看他们自己的权限
- 访问特定的亚马逊托管区块链 (AMB) 查询API操作

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查询资源。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 托管式策略 或 工作职能的AWS 托管式策略。
- 应用最低权限许可 – 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，您可以编写一个策略条件来指定所有请求都必须使用发送SSL。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 AWS CloudFormation。有关更多信息，请参阅《IAM用户指南》中的IAMJSON策略元素：条件。
- 使用 A IAM ccess Analyzer 验证您的IAM策略以确保权限的安全性和功能性 — A IAM ccess Analyzer 会验证新的和现有的策略，以便策略符合IAM策略语言 (JSON) 和IAM最佳实践。

IAMAccess Analyzer 提供了 100 多项策略检查和可行的建议，可帮助您制定安全和实用的策略。有关更多信息，请参阅《IAM用户指南》中的使用 A IAM ccess Analyzer 验证策略。

- 需要多重身份验证 (MFA)-如果您的场景需要IAM用户或 root 用户 AWS 账户，请打开MFA以提高安全性。要要求MFA何时调用API操作，请在策略中添加MFA条件。有关更多信息，请参阅《IAM用户指南》MFA中的使用进行安全API访问。

有关 IAM 中最佳实践的更多信息，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

此示例显示您可以如何创建策略，以便允许 IAM 用户查看附加到其用户身份的内联和托管策略。此策略包括在控制台上或使用或以编程方式完成此操作的 AWS CLI 权限。 AWS API

```
{
    "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查询进行API呼叫，您需要具有相应AMB查询IAM权限的用户证书（AWS_ACCESS_KEY_ID和AWS_SECRET_ACCESS_KEY）。

Example IAM访问所有 Amazon Managed Blockchain (AMB) 查询的政策 APIs

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

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

Example IAM访问亚马逊托管区块链的政策 (AMB) 查询**`ListTransactions`**和 **`GetTransaction`** APIs

此示例授予您中的IAM用户 AWS 账户 访问AMB查询的权限`ListTransaction`和 `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查询列表APIs，请参阅 Amazon Managed Blockchain (AMB) 查询API参考指南。

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

Amazon Managed Blockchain 疑难解答 (AMB) 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 和时可能遇到的常见问题IAM。

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，表明您无权执行某个操作，则您必须更新策略以允许执行该操作。

当mateojacksonIAM用户尝试使用控制台查看虚构*my-example-widget*资源的详细信息但没有虚构权限时，就会出现以下示例错误。managedblockchain-query::*GetWidget*

```
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	描述
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` 操作的响应中移除对该`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 Testnet 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。