aws

デベロッパーガイド

Amazon Managed Blockchain Query

Amazon Managed Blockchain Query: デベロッパーガイド

Table of Contents

Amazon Managed Blockchain (AMB) クエリとは何ですか?

Amazon Managed Blockchain (AMB) は、パブリックブロックチェーンとプライベートブロックチェーンの両方で耐障害性の高い Web3 アプリケーションを構築できるように設計された完全マネージド型サービスです。AMB Access を使用すると、複数のブロックチェーンに瞬時にサーバーレスでアクセスできます。専用のブロックチェーン・インフラストラクチャーをデプロイしたり、ブロックチェーン・ネットワークに接続したままにしたりしなくても、Web3 対応アプリケーションを構築できます。AMB Query では、開発者が使いやすい API 操作を使用して、複数のブロックチェーンのリアルタイムデータや履歴データにアクセスできます。標準化されたブロックチェーンデータは、特別なブロックチェーンインフラストラクチャや ETL (抽出、変換、読み込み) を必要とせずに AWS サービスと統合できます。AMB のすべての機能は、機関レベルのアプリケーションや主流の消費者向けアプリケーションのビルドに合わせて安全に拡張できます。

Amazon Managed Blockchain (AMB) クエリは、開発者が使いやすい API 操作により、標準化されたマルチブロックチェーンデータセットへのサーバーレスアクセスを提供します。AMB Query を使用すると、ブロックチェーンデータの解析、契約のトレース、特殊なインデックスインフラストラクチャの維持などのオーバーヘッドを必要とせずに、1 つ以上のパブリックブロックチェーンからのデータを必要とするアプリケーションを迅速に出荷できます。ファンジブルトークンまたはノンファンジブルトークン（NFT）の過去のトークン残高を分析する場合でも、特定のウォレットアドレスの取引履歴を表示する場合も、Etherなどのネイティブ暗号通貨の分布に関するデータ分析を実行する場合も、AMB Queryを使用するとブロックチェーンデータにアクセスできます。

AMB Query を初めて使用する方ですか？

AMB Query を初めて使用する方には、まず以下のセクションを読むことをおすすめします。

- 主な概念: Amazon Managed Blockchain (AMB) クエリ
- Amazon Managed Blockchain (AMB) クエリの設定
- Amazon Managed Blockchain (AMB) クエリの開始方法
- Amazon Managed Blockchain (AMB) クエリのユースケース

主な概念: Amazon Managed Blockchain (AMB) クエリ

> **Note**
>
> このガイドでは、ブロックチェーンの基本的な概念に精通していることを前提としています。これらの概念には、分散化、トークン、契約、トランザクション proof-of-work、ウォレット、パブリックキーとプライベートキー、ステーキング、マイニング、半分などが含まれます。

Amazon Managed Blockchain (AMB) Query では、マルチブロックチェーンネットワークデータに簡単にアクセスできるため、ブロックチェーンアクティビティに関連するコンテキストデータを抽出しやすくなります。AMB Query を使用して、Bitcoin Mainnet や Ethereum Mainnet などのパブリックブロックチェーンネットワークからデータを読み取ることができます。また、現在および過去の住所残高などの情報を取得したり、特定の期間のブロックチェーントランザクションのリストを取得したりできます。さらに、トランザクションイベントなど、特定のトランザクションの詳細を取得できます。トランザクションイベントは、アプリケーションのビジネスロジックでさらに分析または使用できます。

Amazon Managed Blockchain (AMB) クエリを使用する際の考慮事項と制限事項

AMB クエリを使用する場合は、次の点を考慮してください。

- 利用可能なリージョン

  AMB クエリは、米国東部 (バージニア北部) `us-east-1`リージョンでサポートされています。

- サービスエンドポイント

  AMB クエリには、次のエンドポイントを使用してアクセスできます。

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

- サポートされているブロックチェーンネットワーク

  AMB Query は、以下のパブリックブロックチェーンネットワークをサポートしています。

- Bitcoin Mainnet — proof-of-work コンセンサスによって保護され、Bitcoin (BTC) 暗号通貨が発行されて取引されるパブリック Bitcoin ブロックチェーンネットワーク。Mainnet のトランザクションは実際の値 (つまり、実際のコストが発生します) を持ち、パブリックブロックチェーンに記録されます。
- Bitcoin Testnet — Bitcoin Mainnet のテストネット。このネットワーク上のビットコイン (BTC) は、メインネット BTC とは別個に区別され、通常は値がありません。
- Ethereum Mainnet — パブリック Ethereum ブロックチェーンの proof-of-stake メインネットワーク。Mainnet のトランザクションは実際の値 (つまり、実際のコストが発生します) を持ち、分散台帳に記録されます。
- Sepolia Testnet — Ethereum Mainnet のテストネット。このネットワーク上の Ether (ETH) は Mainnet ETH とは別個に区別され、通常は値がありません。

- サポートされているブロックチェーントークンと契約

  AMB Query は、以下のネイティブおよび標準の Ethereum 契約トークンをサポートしています。

  - パブリックブロックチェーンネイティブトークン
    - Bitcoin (BTC) — これは Bitcoin 関連のブロックチェーンのネイティブトークンです。
    - Ether (ETH) — これは Ethereum 関連のブロックチェーンのネイティブトークンです。

  - Ethereum 契約基準
    - ERC-20 トークンスタンダード — ERC-20 は、ファンジブルトークンの標準です。これには、各 ERC-20 トークンを別の ERC-20 トークンと完全に同じ (タイプと値) にするプロパティがあります。つまり、1 つのトークンは であり、常に他のすべてのトークンと等しくなります。詳細については、Ethereum.org のERC-20 トークンスタンダード」を参照してください。
    - ERC-721 非ファンジブルトークン標準 — ERC-721 は非ファンジブルトークン (NFTs。このタイプのトークンは一意であり、同じ契約内の別のトークンとは異なる値を持つことができます。これは、その経過時間、希少性、またはその他のプロパティが原因である可能性があります。詳細については、Ethereum.org のERC-721 トークンスタンダード」を参照してください。

      ERC-1155 マルチトークン標準 — ERC-1155 は、任意の数のファンジブルトークンタイプと非ファンジブルトークンタイプを表して制御できる契約インターフェイスを作成する標準です。このようにして、ERC-1155 トークンは ERC-20 トークンと ERC-721 トークンと同じように機能し、両方を同時に機能させることができます。ERC-1155 トークンは、ERC-20 標準と ERC-721 標準の両方の機能を改善し、実装の明らかなエラーを修正しながら、より効率的

にします。詳細については、Ethereum.org の ERC-1155 トークンスタンダードを参照してください。

- 確定性

  ブロックチェーンでは、最終度は有効なトランザクションが逆転する可能性が低いことを意味します。Bitcoin Mainnet の場合、AMB クエリは 6 ブロック後にトランザクションを確定と見なします。Bitcoin Testnet の場合、トランザクションは 6 ブロックまたは 60 分のいずれか早い方後に確定したと見なされます。サポートされている Ethereum ネットワークの場合、AMB クエリは 64 ブロック後にトランザクションを確定と見なします。

  AMB Query のトークン残高と契約 API オペレーションは、確定度に達したデータのみを返します。ただし、AMB Query のトランザクションおよびトランザクションイベント API オペレーションは、ブロックチェーンネットワークで確認されたトランザクションのデータを、まだ確定していない場合でも返すことができます。

- NULL アドレスはサポートされていません

  AMB クエリは NULL (`0x0000000000000000000000000000000000000000`) アドレスをサポートしていません。

- API コールの署名バージョン 4 の署名

  AMB クエリ APIs、署名バージョン 4 の署名プロセス を使用して認証された HTTPS 接続を介して呼び出すことができます。つまり、アカウント内の AWS 認可された IAM プリンシパルのみが AMB クエリ API コールを実行できます。これを行うには、 コールで AWS 認証情報 (アクセスキー ID とシークレットアクセスキー) を指定する必要があります。

  > **Important**
  > ユーザー向けアプリケーションにクライアント認証情報を埋め込まないでください。

- AMB クエリが Bitcoin トランザクション識別子とトランザクションハッシュをサポート

  Bitcoin ネットワークの場合、AMB クエリ API オペレーションはトランザクション識別子 (`transactionId`) とトランザクションハッシュ () の両方をサポートしますtransactionHash。`transactionId` は、監視データを含まないトランザクションのダブル

SHA ハッシュです。transactionHash は、監視データ (監視トランザクション ID とも呼ばれます) を含むトランザクションのダブル SHA ハッシュです。

Bitcoin ネットワークの GetTransactionまたは ListTransactionEvents API オペレーションを呼び出すときは、 transactionIdまたは を指定できますtransactionHash。また、transactionIdまたは を返す Bitcoin ネットワーク上のすべての AMB クエリオペレーションtransactionHashには、レスポンスの一部として両方の値が含まれます。

Amazon Managed Blockchain (AMB) クエリの設定

Amazon Managed Blockchain (AMB) クエリを初めて使用する前に、このセクションの手順に従ってを作成します。 AWS アカウント。次のセクションでは、AMBクエリの使用を開始する方法について説明します。

前提条件と考慮事項

Amazon Web Services を初めて使用するには、 が必要です。 AWS アカウント。

サインアップ: AWS

Amazon Web Services (AWS) 、 AWS アカウントはすべての に自動的にサインアップされます AWS のサービス Amazon Managed Blockchain (AMB) クエリを含む 。サービスを実際に使用した分の料金のみが請求されます。

をお持ちの場合 AWS アカウント 既に、次のステップに進みます。をお持ちでない場合 AWS アカウント、次の手順を使用して作成します。

を作成するには AWS アカウント

1. https://portal.aws.amazon.com/billing/サインアップ を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、通話呼び出しを受け取り、電話キーパッドで検証コードを入力するように求められます。

   にサインアップするとき AWS アカウント、 AWS アカウントのルートユーザー が作成されます。ルートユーザーはすべての にアクセスできます AWS のサービス アカウントの および リソース。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用してルートユーザーアクセスが必要なタスクを実行してください。

適切なアクセス許可を持つ IAM ユーザーを作成する

クエリを作成して使用するには、 AMB を作成する必要があります。 AWS Identity and Access Management （ IAM) 必要な Managed Blockchain アクションを許可するアクセス許可を持つプリンシパル (ユーザーまたはグループ) 。

プリンIAMシパルのみがAMBクエリAPIリクエストを行うことができます。AMB クエリ を呼び出すときはAPIs、署名バージョン 4 の署名プロセス を使用して認証されたHTTPS接続を介して呼び出すことができます。つまり、 の認可されたIAMプリンシパルのみが AWS アカウントはAMBクエリAPI呼び出しを行うことができます。以下の手順に従ってください。 AWS 認証情報 (アクセスキー ID とシークレットアクセスキー) は、 呼び出しとともに提供する必要があります。

IAM ユーザーの作成方法については、「 での IAM ユーザーの作成」を参照してください。 AWS アカウント 。アクセス許可ポリシーをユーザーにアタッチする方法の詳細については、「ユーザーのアクセス許可の変更IAM」を参照してください。クエリを操作するアクセス許可をユーザーに付与するために使用できるアクセス許可ポリシーの例については、AMB「」を参照してくださいAmazon Managed Blockchain (AMB) クエリのアイデンティティベースのポリシーの例。

のインストールと設定 AWS Command Line Interface

まだインストールしていない場合は、最新の AWS で使用するコマンドラインインターフェイス (CLI) AWS ターミナルからの リソース。詳細については、「 の最新バージョンのインストールまたは更新」を参照してください。 AWS CLI.

> **Note**
>
> CLI アクセスには、アクセスキー ID とシークレットアクセスキーが必要です。長期のアクセスキーの代わりに一時的な認証情報をできるだけ使用します。一時的な認証情報には、アクセスキー ID、シークレットアクセスキー、および認証情報の失効を示すセキュリティトークンが含まれています。詳細については、「 での一時的な認証情報の使用」を参照してください。 AWSIAM ユーザーガイドの リソース。

を使用する AWS Management Console Amazon Managed Blockchain (AMB) クエリを使用してブロックチェーンをクエリするには

Amazon Managed Blockchain (AMB) Query にアクセスし、 を使用してサポートされているブロックチェーンネットワークに対してクエリを実行できます。 AWS Management Console。 次の手順は、これを行う方法を示しています。

1. で Amazon Managed Blockchain コンソールを開きますhttps://console.aws.amazon.com/managedblockchain/。

2. クエリセクションからクエリエディタを選択します。
3. サポートされているブロックチェーンネットワーク のいずれかを選択します。
4. 実行するクエリタイプを選択します。
5. 選択したクエリタイプの関連パラメータを入力し、クエリ を実行します。

AMB クエリはクエリを実行し、クエリ結果ウィンドウに結果が表示されます。

Amazon Managed Blockchain (AMB) クエリの開始方法

Amazon step-by-step Managed Blockchain (AMB) クエリを使用してタスクを実行する方法については、このセクションのチュートリアルを参照してください。これらの手順にはいくつかの前提条件が必要です。AMB クエリを初めて使用する場合は、このガイドの「セットアップ」セクションを参照してください。詳細については、「Amazon Managed Blockchain (AMB) クエリの設定」を参照してください。

> **Note**
>
> これらの例の一部の変数は意図的に難読化されています。これらの例を実行する前に、それらを独自の有効なものに置き換えてください。

トピック

- クエリAPIオペレーションにアクセスするための AMB IAMポリシーを作成する
- Go を使用して Amazon Managed Blockchain (AMB) クエリAPIリクエストを行う
- Node.js を使用して Amazon Managed Blockchain (AMB) クエリAPIリクエストを行う
- Python を使用して Amazon Managed Blockchain (AMB) クエリAPIリクエストを行う
- で Amazon Managed Blockchain (AMB) クエリ AWS Management Console を使用して GetTokenBalance オペレーションを実行する

クエリAPIオペレーションにアクセスするための AMB IAMポリシーを作成する

AMB クエリAPIリクエストを行うには、Amazon Managed Blockchain (KEY) クエリの適切なIAMアクセス許可を持つユーザー認証情報 (AWS_ACCESS_KEY_ID および AWS_SECRET_ACCESS_AMB) を使用する必要があります。 AWS CLI がインストールされているターミナルで、次のコマンドを実行して、クエリAPIオペレーションにアクセスするための AMB IAMポリシーを作成します。

```
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
```

ポリシーを作成したら、そのポリシーを IAM ユーザーのロールにアタッチして有効にします。で AWS Management Console、IAMサービスに移動し、サービスを使用するIAMユーザーに割り当てられたAmazonManagedBlockchainQueryAccessロールにポリシーをアタッチします。詳細については、「ロールの作成」およびIAM「 ユーザーへの の割り当て」を参照してください。

> **Note**
>
> AWS では、ワイルドカード を使用するのではなく、特定の APIオペレーションへのアクセスを許可することをお勧めします*。詳細については、「特定の Amazon Managed Blockchain (AMB) クエリAPIアクションへのアクセス」を参照してください。

Go を使用して Amazon Managed Blockchain (AMB) クエリAPIリクエストを行う

Amazon Managed Blockchain (AMB) Query を使用すると、ブロックチェーンデータへの即時アクセスに依存するアプリケーションを構築できます。これは、ブロックチェーンで確認されても、まだ確定度に達していない場合でも可能です。 AMBクエリを使用すると、ウォレットのトランザクション履歴の入力、トランザクションハッシュに基づくトランザクションに関するコンテキスト情報の提供、ネイティブトークンと ERC-721、ERC-1155、ERC-20 トークンのバランスの取得など、いくつかのユースケースが可能になります。

次の例は、Go 言語で作成され、AMBクエリAPIオペレーションを使用します。Go の詳細については、Go ドキュメントを参照してください。AMB クエリ の詳細についてはAPI、「Amazon Managed Blockchain (AMB) クエリAPIリファレンスドキュメント」を参照してください。

次の例では、 ListTransactionsおよび GetTransactionAPIアクションを使用して、Ethereum Mainnet 上の特定の外部所有アドレス (EOA) のすべてのトランザクションのリストを取得し、次の例では、リストから 1 つのトランザクションのトランザクション詳細を取得します。

Example — Go を使用して**ListTransactions**APIアクションを実行する

次のコードを ListTransactions ディレクトリlistTransactions.goの という名前のファイルにコピーします。

```
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)
    }
}
```

ファイルを保存したら、 ListTransactions ディレクトリ内の次のコマンドを使用してコードを実行します。 `go run listTransactions.go`

次の出力は次のようになります。

```
{
  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 — Go を使用して**GetTransaction**APIアクションを実行する

この例では、前の出力のトランザクションハッシュを使用します。次のコードを GetTransaction ディレクトリGetTransaction.goの という名前のファイルにコピーします。

```
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)
    }
}
```

ファイルを保存したら、 GetTransaction ディレクトリ内の次のコマンドを使用してコードを実行します。 go run GetTransaction.go

次の出力は次のようになります。

```
{
  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
  }
```

```
}
```

GetTokenBalance API は、ある時点で外部所有アカウント (ETHBTC) の現在の残高を取得するために使用できるネイティブトークン (および EOA) の残高を取得する方法を提供します。

Example — **GetTokenBalance**APIアクションを使用して Go のネイティブトークンの残高を取得します。

次の例では、 を使用して Ethereum Mainnet のアドレス Ether (ETH) 残高GetTokenBalanceAPIを取得します。次のコードを GetTokenBalance ディレクトリGetTokenBalanceEth.goの という名前のファイルにコピーします。

```
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)
    }
}
```

ファイルを保存したら、 GetTokenBalance ディレクトリ内の次のコマンドを使用してコードを実行します。 `go run GetTokenBalanceEth.go`

次の出力は次のようになります。

```
{
  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"
  }
}
```

Node.js を使用して Amazon Managed Blockchain (AMB) クエリ APIリクエストを行う

これらのノード例を実行するには、次の前提条件が適用されます。

1. マシンにノードバージョンマネージャー (nvm) と Node.js がインストールされている必要があります。OS のインストール手順については、こちらを参照してください。
2. node --version コマンドを使用して、Node バージョン 14 以降を使用していることを確認します。必要に応じて、 nvm install 14 コマンドを使用し、続いて nvm use 14 コマンドを使用してバージョン 14 をインストールできます。
3. 環境変数 AWS_ACCESS_KEY_IDと には、アカウントに関連付けられている認証情報が含まれているAWS_SECRET_ACCESS_KEY必要があります。

   次のコマンドを使用して、これらの変数をクライアントで文字列としてエクスポートします。以下の強調表示された値をIAM、ユーザーアカウントの適切な値に置き換えます。

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

> **Note**
>
> - すべての前提条件を完了したら、 を介して署名付きリクエストを送信HTTPSして Amazon Managed Blockchain (AMB) クエリAPIオペレーションにアクセスし、Node.js のネイティブ https モジュールを使用してリクエストを行うことができます。または、 などのサードパーティーライブラリを使用してAMBクエリからデータAXIOSを取得できます。
> - これらの例では Node.js のサードパーティーHTTPクライアントを使用していますが、AMB を使用してクエリを AWS JavaScript SDKリクエストすることもできます。
> - 次の例は、Axios AMB と AWS SigV4 のSDKモジュールを使用してクエリAPIリクエストを行う方法を示しています。

次のpackage.jsonファイルをローカル環境の作業ディレクトリにコピーします。

```
{
  "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 — クエリを使用して、特定の外部所有アドレス (EOA) AMB からトークン残高の履歴を取得する **`GetTokenBalance`** API

を使用してGetTokenBalanceAPI、さまざまなトークン (ERC20、、 ERC721などERC1155) とネイティブコイン (ETHや などBTC) の残高を取得できます。これを使用して、履歴 (Unix タイムスタンプ - 秒) に基づいて外部所有アカウント timestamp (EOA) の現在の残高を取得できます。この例では、 を使用して、Ethereum Mainnet で ERC20 トークンUSDCのアドレスバランスGetTokenBalanceAPIを取得します。

をテストするにはGetTokenBalanceAPI、次のコードを という名前のファイルにコピーしtoken-balance.js、そのファイルを同じ作業ディレクトリに保存します。

```
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);
```

コードを実行するには、 ファイルと同じディレクトリでターミナルを開き、次のコマンドを実行します。

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

このコマンドは、スクリプトを実行し、コードで定義された引数を渡して、Ethereum Mainnet に EOAリストされている の ERC20 USDC残高をリクエストします。応答は次の例のようになります。

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

Python を使用して Amazon Managed Blockchain (AMB) クエリAPI リクエストを行う

これらの Python の例を実行するには、次の前提条件が適用されます。

1. マシンに Python がインストールされている必要があります。OS のインストール手順については、こちらを参照してください。
2. AWS SDK for Python (Boto3) をインストールします。
3. AWS コマンドラインインターフェイスをインストールし、 コマンドを実行してAccess Key ID、、Secret Access Key、および の変数aws configureを設定しますRegion。

すべての前提条件を完了したら、 SDK for Python over を使用して AWS Amazon Managed Blockchain (AMB) クエリAPIリクエストHTTPSを行うことができます。

次の Python の例では、boto3 のモジュールを使用して、必要な SigV4 ヘッダーがアタッチされたリクエストをAMBクエリListTransactionEventsAPIオペレーションに送信します。この例では、Ethereum Mainnet 上の特定のトランザクションによって出力されるイベントのリストを取得します。

次のlist-transaction-events.pyファイルをローカル環境の作業ディレクトリにコピーします。

```
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')))
```

サンプルコードを に実行するにはListTransactionEvents、ファイルを作業ディレクトリに保存し、コマンド を実行しますpython3 list-transaction-events.py。このコマンドはスクリプトを実行し、コードで定義された引数を渡して、Ethereum Mainnet 上の指定されたトランザクションハッシュに関連付けられたイベントをリクエストします。応答は次の例のようになります。

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

で Amazon Managed Blockchain (AMB) クエリ AWS Management Console を使用して GetTokenBalance オペレーションを実行する

次の例は、 で Amazon Managed Blockchain (AMB) クエリを使用して Ethereum Mainnet でトークンの残高を取得する方法を示しています。 AWS Management Console

Example

1. で Amazon Managed Blockchain コンソールを開きますhttps://console.aws.amazon.com/managedblockchain/。
2. クエリセクションからクエリエディタを選択します。
3. ブロックチェーンネットワークとして ETHEREUM_MAINNET を選択します。
4. クエリタイプGetTokenBalanceとして を選択します。
5. トークンのブロックチェーンアドレスを入力します。

6. トークンの契約アドレスを入力します。
7. トークンのオプションのトークン ID を入力します。
8. トークン残高の日付を選択します。
9. トークン残高のオプションの At time を入力します。
10. [Run query] (クエリの実行) を選択します。

AMB クエリはクエリを実行し、クエリ結果ウィンドウに結果が表示されます。

Amazon Managed Blockchain (AMB) クエリのユースケース

このトピックでは AMB Query のユースケースのリストを提供します。

トピック

現在および過去のトークン残高を照会します。

GetTokenBalanceAPI は、外部所有アカウント (EOA) のユニバーサルタイムスタンプ (Unix タイムスタンプ、秒単位) を使用して、サポートされているトークン (ERC20、ERC721、ERC1155) とネイティブコイン (ETH、BTC) の残高を取得し、現在または過去の残高を取得します。たとえば、GetTokenBalance APIオペレーションを使用して、イーサリアムメインネット上のERC20トークン (USDC) のアドレス残高を取得できます。API オペレーションを使用して、トークンとネイティブコインの残高を一括取得することもできます。BatchGetTokenBalance

詳細については、Amazon Managed Blockchain (AMB) クエリリファレンスガイドを参照してください。

過去のトランザクションデータを取得する

Amazon Managed Blockchain (AMB) クエリを使用すると、イーサリアムやビットコインなどのパブリックブロックチェーンから履歴データを取得できます。この機能により、ブロックチェーンウォレットでトランザクション履歴を取得したり、トランザクションハッシュに基づいてトランザクションに関するコンテキスト情報を提供したりするなど、さまざまなユースケースが可能になります。ListTransactionsAPI オペレーションを使用して Ethereum メインネット上の特定の外部所有アドレス (EOA) のトランザクションのリストを取得し、次に GetTransactionAPI オペレーションを使用してリストから 1 つのトランザクションのトランザクション詳細を取得できます。

詳細については、Amazon Managed Blockchain (AMB) クエリリファレンスガイドを参照してください。

特定のアドレスのトークン残高をすべて取得します。

ListTokenBalancesAPI オペレーションを使用して、ウォレット、ユーザーインターフェイス、web3 ユーティリティなどの残高を取得できます。このAPIオペレーションは、1つのAPIオペレーションを使用して、特定のパブリックブロックチェーン上のトークン (ERC20、ERC721、ERC1155) とネイティブコイン (ETH、BTC) にわたるアドレスの全残高のリストを返します。たとえば、外部所有アドレス (EOA) とネットワーク (Ethereum Mainnet) を指定すると、応答でトークンとネイティブコイン残高のリストを受け取ることができます。

詳細については、Amazon Managed Blockchain (AMB) クエリリファレンスガイドを参照してください。

トランザクションで発生したイベントを一覧表示します。

ListTransactionEventsAPI オペレーションを使用して、特定のトランザクションの結果として発生するコントラクトイベントのリストを、そのハッシュ (トランザクション識別子) で識別して取得できます。たとえば、ListTransactionEventsを使用して、Ethereum Blockchain上のERC20トークンコントラクトの関数を呼び出すトランザクションの結果となるイベント (ERC20コントラクトからのTransferイベントや出金イベントなど) を取得できます。

詳細については、Amazon Managed Blockchain (AMB) クエリリファレンスガイドを参照してください。

契約によって発行されたトークンをすべて取得する

ListTokenBalancesAPI オペレーションを使用すると、コントラクトアドレスを入力として渡したときにコントラクトによって発行された、サポートされているすべてのトークン (ERC20、ERC721、ERC1155) のリストを返すことができます。たとえば、Ethereumブロックチェーン上でERC721契約標準によって発行されたノンファンジブルトークン (NFT) に関する情報をAPIオペレーションを使用して取得できます。ListTokenBalances

詳細については、Amazon Managed Blockchain (AMB) クエリリファレンスガイドを参照してください。

契約の一覧と契約情報の取得

ListAssetContractsAPI オペレーションを使用して、特定のアドレスでデプロイされた ERC-721、ERC-1155、または ERC-20 コントラクトを一覧表示できます。さらに、コントラクトア

ドレスがわかっている場合は、GetAssetContractAPI オペレーションを使用して、コントラクトタイプ、デプロイヤアドレス、関連するトークンメタデータなどのコントラクトのプロパティを取得できます。

詳細については、Amazon Managed Blockchain (AMB) クエリリファレンスガイドを参照してください。

Amazon Managed Blockchain (AMB) クエリ API リファレンス

Amazon Managed Blockchain (AMB) クエリは、サポートされているブロックチェーンをクエリするための API オペレーションを提供します。これには、トークン、トランザクション、コントラクトをクエリするための API が含まれます。詳細については、AMB Query API リファレンスをご覧ください。

Amazon Managed Blockchain (AMB) クエリのセキュリティ

AWS クラウドセキュリティは最優先事項です。 AWS お客様は、最もセキュリティに敏感な組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャの恩恵を受けることができます。

セキュリティは、 AWS お客様とお客様との間で共有される責任です。責任分担モデルでは、これをクラウドのセキュリティとクラウドのセキュリティの両方と表現しています。

- クラウドのセキュリティ — AWS AWS AWS クラウドクラウド内でサービスを実行するインフラストラクチャを保護する責任があります。 AWS また、安全に使用できるサービスも提供します。サードパーティーの監査人は、AWS コンプライアンスプログラム の一環として、セキュリティの有効性を定期的にテストおよび検証します。Amazon Managed Blockchain (AMB) クエリに適用されるコンプライアンスプログラムについては、「AWS コンプライアンスプログラムによる対象サービス」を参照してください。
- クラウドのセキュリティ — お客様の責任は、 AWS 使用するサービスによって決まります。また、お客様は、 お客様のデータの機密性、企業の要件、および適用可能な法律や規制といった他の要因 についても責任を担います。

データ保護、認証、アクセス制御を提供するために、Amazon Managed Blockchain は、 AWS マネージドブロックチェーンで実行されているオープンソースフレームワークの機能と機能を使用しています。

このドキュメントは、AMB Query を使用する際に責任分担モデルを適用する方法を理解するのに役立ちます。以下のトピックでは、セキュリティとコンプライアンスの目標を満たすように AMB Query を設定する方法を説明します。AMB Query AWS リソースの監視と保護に役立つ他のサービスの使い方についても学ぶことができます。

トピック

- データ暗号化
- Amazon Managed Blockchain (AMB) クエリの Identity and Access Management

データ暗号化

データ暗号化は、権限のないユーザーがブロックチェーンネットワークおよび関連するデータストレージシステムからデータを読み取るのを防ぐのに役立ちます。これには、ネットワーク上を移動する際に傍受される可能性のあるデータ (「転送中のデータ」) が含まれます。

転送中の暗号化

デフォルトでは、Managed Blockchain は HTTPS/TLS 接続を使用して、クライアントからサービスエンドポイントに送信されるすべてのデータを暗号化します。 AWS CLI AWS

Amazon Managed Blockchain (AMB) クエリの Identity and Access Management

AWS Identity and Access Management （ IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。 IAM管理者は、誰を認証 (サインイン) し、誰に Query リソースの使用を承認する (アクセス許可を付与 AWS のサービス する) AMB かを制御します。 IAM は、追加料金なしで使用できる です。

トピック

- 対象者
- アイデンティティを使用した認証
- ポリシーを使用したアクセスの管理
- Amazon Managed Blockchain (AMB) クエリと の連携 IAM
- Amazon Managed Blockchain (AMB) クエリのアイデンティティベースのポリシーの例
- Amazon Managed Blockchain (AMB) クエリのアイデンティティとアクセスのトラブルシューティング

対象者

AWS Identity and Access Management （ IAM) の使用方法は、AMBクエリで行う作業によって異なります。

サービスユーザー – ジョブを実行するために AMB クエリサービスを使用する場合、管理者から必要な認証情報とアクセス許可が提供されます。さらに多くのAMBクエリ機能を使用して作業を行

う場合は、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておくと、管理者に適切な許可をリクエストするうえで役立ちます。Query の機能にアクセスできない場合は、AMB「」を参照してくださいAmazon Managed Blockchain (AMB) クエリのアイデンティティとアクセスのトラブルシューティング。

サービス管理者 – 社内のAMBクエリリソースを担当している場合は、通常、AMBクエリへのフルアクセスがあります。サービスユーザーがどのAMBクエリ機能とリソースにアクセスするかを決めるのは管理者の仕事です。その後で、サービスユーザーのアクセス許可を変更するために、IAM管理者にリクエストを送信する必要があります。IAM の基本概念については、このページの情報を確認します。会社が クエリIAMで AMB を使用する方法の詳細については、「」を参照してくださいAmazon Managed Blockchain (AMB) クエリと の連携 IAM。

IAM 管理者 – IAM管理者は、AMBクエリへのアクセスを管理するポリシーの作成方法の詳細について確認する場合があります。で使用できる AMB Query アイデンティティベースのポリシーの例を表示するにはIAM、「」を参照してくださいAmazon Managed Blockchain (AMB) クエリのアイデンティティベースのポリシーの例。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、 IAM ユーザーとして AWS アカウントのルートユーザー、または IAMロールを引き受けることによって、認証 (にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center （ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーティッド ID の例です。フェデレーテッドアイデンティティ としてサインインする場合、管理者は以前に IAM ロールを使用して ID フェデレーションを設定しています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 AWS サインイン ユーザーガイド」の「 にサインインする方法 AWS アカウント」を参照してください。

AWS プログラムで にアクセスする場合、 はソフトウェア開発キット (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストに暗号で署名します。 AWS ツールを使用しない場合は、リクエストに自分で署名する必要があります。推奨される方法を使用してリクエストに署名する方法の詳細については、「 IAMユーザーガイド」のAWS API「リクエストの署名バージョン 4」を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。例えば、 では、アカウントのセキュリティを強化するために多要素認証 (MFA) を使用する AWS ことをお勧めします。詳細については、「 AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「 ユーザーガイド」のAWS 「 での多要素認証IAMIAM」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、IAMユーザー ガイドの「ルートユーザー資格情報が必要なタスク」を参照してください。

フェデレーテッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、ID プロバイダーとのフェデレーションを使用して一時的な認証情報 AWS のサービス を使用して にアクセスすることを要求します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリのユーザー、または ID ソースを通じて提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID がアクセスすると AWS アカウント、ロールを引き受け、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成することも、独自の ID ソースのユーザーとグループのセットに接続して同期し、すべての AWS アカウント とアプリケーションで使用できるようにすることもできます。IAM Identity Center の詳細については、「 AWS IAM Identity Center ユーザーガイド」のIAM「Identity Center とは」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を持つIAMユーザーを作成するのではなく、一時的な認証情報を使用することをお勧めします。ただし、IAMユーザーとの長期的な認証情報を必要とする特定のユースケースがある場合は、ア

クセスキーをローテーションすることをお勧めします。詳細については、IAMユーザーガイドの「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーのコレクションを指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、 という名前のグループを作成しIAMAdmins、そのグループに IAM リソースを管理するアクセス許可を付与できます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時認証情報が提供されます。詳細については、「 IAMユーザーガイド」のIAM「 ユーザーのユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のアイデンティティです。これは IAM ユーザーに似ていますが、特定のユーザーに関連付けられていません。で IAMロールを一時的に引き受けるには AWS Management Console、ユーザーから IAMロール (コンソール) に切り替えることができます。ロールを引き受けるには、 または AWS API オペレーションを AWS CLI 呼び出すか、カスタム を使用しますURL。ロールを使用する方法の詳細については、「 IAMユーザーガイド」の「ロールを引き受ける方法」を参照してください。

IAM ロールと一時認証情報は、次の状況で役立ちます。

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールの詳細については、「 ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) のロールを作成する」を参照してください。 IAM IAM Identity Center を使用する場合は、アクセス許可セットを設定します。ID が認証後にアクセスできる内容を制御するために、IAMIdentity Center はアクセス許可セットを のロールに関連付けますIAM。アクセス許可セットの詳細については、「AWS IAM Identity Center User Guide」の「Permission sets」を参照してください。
- 一時的なIAMユーザーアクセス許可 – IAM ユーザーまたはロールは、 IAMロールを引き受けて、特定のタスクに対して異なるアクセス許可を一時的に引き受けることができます。
- クロスアカウントアクセス IAMロールを使用して、自分のアカウントのリソースにアクセスすることを別のアカウントの信頼済みプリンシパルに許可できます。クロスアカウントアクセスを許可

する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、(ロールをプロキシとして使用する代わりに) リソースに直接ポリシーをアタッチできます。クロスアカウントアクセスのロールとリソースベースのポリシーの違いについては、「 IAMユーザーガイド」の「 でのクロスアカウントリソースアクセスIAM」を参照してください。

- クロスサービスアクセス — 一部の は他の の機能 AWS のサービス を使用します AWS のサービス。例えば、サービスで呼び出しを行うと、そのサービスが Amazon でアプリケーションを実行EC2したり、Amazon S3 にオブジェクトを保存したりするのが一般的です。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用してアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用すると、別のサービスで別のアクションを開始するアクションを実行できます。 FASは、 を呼び出すプリンシパルのアクセス許可をリクエストと組み合わせて使用し AWS のサービス、ダウンストリームサービス AWS のサービス にリクエストを送信します。 FAS リクエストは、他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストをサービスが受信した場合にのみ行われます。この場合、両方のアクションを実行するための権限が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。
  - サービスロール – サービスロールは、ユーザーに代わってアクションを実行するためにサービスが引き受ける IAMロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「 IAMユーザーガイド」の「 にアクセス許可を委任するロールを作成する AWS のサービス」を参照してください。
  - サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、 サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。
- Amazon で実行されているアプリケーション EC2 – IAMロールを使用して、EC2インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。 AWS ロールをEC2インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時認証情報を取得することができます。詳細については、「 ユーザーガイド」の「 IAMロールを使用して Amazon EC2インスタンスで実行されているアプリケーションにアクセス許可を付与する」を参照してください。IAM

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは のオブジェクト AWS であり、アイデンティティまたはリソースに関連付けられると、そのアクセス許可を定義します。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーはJSON、ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「 IAMユーザーガイド」のJSON「ポリシーの概要」を参照してください。

管理者はポリシーを使用して AWS JSON、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

デフォルトでは、ユーザーやロールに権限はありません。IAM管理者は、リソースで必要なアクションを実行するためのアクセス許可をユーザーに付与する IAMポリシーを作成できます。その後、管理者はロールに IAMポリシーを追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションのアクセス許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS からロール情報を取得できます API。

アイデンティティベースのポリシー

アイデンティティベースのポリシーは、 IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできるJSONアクセス許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。ID ベースのポリシーを作成する方法については、「 IAMユーザーガイド」の「カスタマー管理ポリシーを使用してカスタムIAMアクセス許可を定義する」を参照してください。

アイデンティティベースポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。管理ポリシーまたはインラインポリシーを選択する方法については、「 IAMユーザーガイド」の「管理ポリシーとインラインポリシーの選択」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースにアタッチするJSONポリシードキュメントです。リソースベースポリシーの例としては、IAMロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーIAMでは、 から AWS 管理ポリシーを使用することはできません。

アクセスコントロールリスト (ACLs)

アクセスコントロールリスト (ACLs) は、リソースへのアクセス許可を持つプリンシパル (アカウントメンバー、ユーザー、またはロール) を制御します。 ACLsは、ポリシードキュメント形式を使用しませんが、リソースベースのJSONポリシーと似ています。

Amazon S3、 AWS WAF、および Amazon VPCは、 をサポートするサービスの例ですACLs。の詳細についてはACLs、Amazon Simple Storage Service デベロッパーガイドの「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的ではない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 – アクセス許可の境界は、アイデンティティベースのポリシーがIAMエンティティ (IAM ユーザーまたはロール) に付与できるアクセス許可の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、IAM ユーザー ガイドの「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) の最大アクセス許可を指定するJSONポリシーです AWS Organizations。 AWS Organizations は、ビジネスが所

有する複数の AWS アカウント をグループ化して一元管理するためのサービスです。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCPs) をアカウントの一部またはすべてに適用できます。は、各 を含むメンバーアカウントのエンティティのアクセス許可SCPを制限します AWS アカウントのルートユーザー。Organizations と の詳細についてはSCPs、「 AWS Organizations ユーザーガイド」の「サービスコントロールポリシー」を参照してください。

- リソースコントロールポリシー (RCPs) – 所有する各リソースにアタッチされたJSONポリシーを更新することなく、アカウント内のリソースに使用可能なアクセス許可の上限を設定するために使用できるIAMポリシーRCPsです。は、メンバーアカウントのリソースのアクセス許可RCPを制限し、組織に属しているかどうかにかかわらず AWS アカウントのルートユーザー、 を含む ID の有効なアクセス許可に影響を与える可能性があります。をサポートする のリストを含む Organizations と の詳細についてはRCPs、AWS Organizations 「 ユーザーガイドRCPs」の AWS のサービス「リソースコントロールポリシー (RCPs) 」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、IAM ユーザーガイドの「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関係する場合にリクエストを許可するかどうかを AWS が判断する方法については、「 IAMユーザーガイド」の「ポリシーの評価ロジック」を参照してください。

Amazon Managed Blockchain (AMB) クエリと の連携 IAM

IAM を使用してAMBクエリへのアクセスを管理する前に、AMBクエリで使用できるIAM機能を確認してください。

IAM Amazon Managed Blockchain (AMB) クエリで使用できる の機能

IAM 機能	AMB クエリのサポート
アイデンティティベースポリシー	はい

IAM 機能	AMB クエリのサポート
リソースベースのポリシー	いいえ
ポリシーアクション	はい
ポリシーリソース	いいえ
ポリシー条件キー	いいえ
ACLs	いいえ
ABAC (ポリシー内のタグ)	いいえ
一時的な認証情報	はい
プリンシパル権限	はい
サービスロール	いいえ
サービスリンクロール	いいえ

Query およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要を把握するには、AMB「 IAMユーザーガイド」のAWS 「 と連携する のサービスIAM」を参照してください。

AMB クエリのアイデンティティベースのポリシー

アイデンティティベースのポリシーのサポート: あり

アイデンティティベースのポリシーは、 IAM ユーザー、ユーザーのグループ、ロールなど、アイデンティティにアタッチできるJSONアクセス許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、「 IAMユーザーガイド」の「カスタマー管理ポリシーによるカスタムIAMアクセス許可の定義」を参照してください。

IAM のアイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、またアクションが許可または拒否される条件を指定できます。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素については、「 IAMユーザーガイド」の「 IAMJSONポリシー要素リファレンス」を参照してください。

AMB クエリのアイデンティティベースのポリシーの例

AMB Query アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいいAmazon Managed Blockchain (AMB) クエリのアイデンティティベースのポリシーの例。

AMB クエリ内のリソースベースのポリシー

リソースベースのポリシーのサポート: なし

リソースベースのポリシーは、リソースにアタッチするJSONポリシードキュメントです。リソースベースポリシーの例としては、IAMロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる場合 AWS アカウント、信頼されたアカウントのIAM管理者は、リソースへのアクセス許可をプリンシパルエンティティ (ユーザーまたはロール) に付与する必要もあります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーをさらに付与する必要はありません。詳細については、「 IAMユーザーガイド」の「 でのクロスアカウントリソースアクセスIAM」を参照してください。

AMB クエリのポリシーアクション

ポリシーアクションのサポート:あり

管理者はポリシーを使用して AWS JSON、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action`要素は、ポリシーでアクセスを許可または拒否するために使用できるアクションを記述します。ポリシーアクションの名前は通常、関連する AWS APIオペレーションと同じ

です。一致するAPIオペレーションがないアクセス許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数のアクションが必要なオペレーションもあります。これらの追加アクションは、依存アクションと呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

AMB クエリアクションのリストを確認するには、「サービス認可リファレンス」の「Amazon Managed Blockchain (AMB) クエリで定義されるアクション」を参照してください。

Query AMB のポリシーアクションは、アクションの前に次のプレフィックスを使用します。

```
managedblockchain-query:
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

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

AMB Query アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAmazon Managed Blockchain (AMB) クエリのアイデンティティベースのポリシーの例。

AMB クエリのポリシーリソース

ポリシーリソースに対するサポート: いいえ

管理者はポリシーを使用して AWS JSON、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

Policy ResourceJSON要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource 要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの許可と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

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

AMB クエリリソースタイプとその のリストを確認するにはARNs、「サービス認可リファレンス」の「Amazon Managed Blockchain (AMB) クエリで定義されるリソース」を参照してください。各リソースARNの を指定できるアクションについては、「Amazon Managed Blockchain (AMB) クエリで定義されるアクション」を参照してください。

AMB Query アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAmazon Managed Blockchain (AMB) クエリのアイデンティティベースのポリシーの例。

AMB クエリのポリシー条件キー

サービス固有のポリシー条件キーのサポート: なし

管理者はポリシーを使用して AWS JSON、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルが、どのリソースに対してどのような条件下でアクションを実行できるかということです。

Condition 要素 (または Condition ブロック) を使用すると、ステートメントが有効な条件を指定できます。Condition 要素はオプションです。イコールや未満などの条件演算子を使用して条件式を作成することで、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の Condition 要素を指定する場合、または 1 つの Condition 要素に複数のキーを指定する場合、 AWS では AND 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理ORオペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。たとえば、IAM ユーザー名でタグ付けされている場合のみ、リソースにアクセスする IAM ユーザーアクセス許可を付与できます。詳細については、IAMユーザーガイドの「 IAMポリシーエレメント: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートしています。すべての AWS グローバル条件キーを確認するには、「 IAMユーザーガイド」のAWS 「 グローバル条件コンテキストキー」を参照してください。

AMB クエリ条件キーのリストを確認するには、「サービス認可リファレンス」の「Amazon Managed Blockchain (AMB) クエリの条件キー」を参照してください。条件キーを使用できるアクションとリソースについては、「Amazon Managed Blockchain (AMB) で定義されるアクション」を参照してください。

AMB Query アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいAmazon Managed Blockchain (AMB) クエリのアイデンティティベースのポリシーの例。

ACLs AMBクエリの

をサポートACLs： いいえ

アクセスコントロールリスト (ACLs) は、リソースへのアクセス許可を持つプリンシパル (アカウントメンバー、ユーザー、またはロール) を制御します。 ACLsは、ポリシードキュメント形式を使用しませんが、リソースベースのJSONポリシーと似ています。

ABAC AMBクエリを使用する

サポート ABAC (ポリシー内のタグ): いいえ

属性ベースのアクセスコントロール (ABAC) は、属性に基づいてアクセス許可を定義する認可戦略です。では AWS、これらの属性はタグと呼ばれます。タグは、IAMエンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、 の最初のステップですABAC。次に、プリンシパルのタグがアクセスしようとしているリソースのタグと一致する場合に、オペレーションを許可するABACポリシーを設計します。

ABAC は、急速に成長している環境や、ポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

の詳細についてはABAC、「 IAMユーザーガイド」のABAC「認可によるアクセス許可の定義」を参照してください。をセットアップする手順を含むチュートリアルを表示するにはABAC、「 ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。 IAM

AMB クエリでの一時的な認証情報の使用

一時的な認証情報のサポート: あり

一部の AWS のサービス は、一時的な認証情報を使用してサインインすると機能しません。一時的な認証情報と AWS のサービス 連携する などの詳細については、「 IAMユーザーガイド」のAWS のサービス「 と連携する IAM 」を参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合、一時的な認証情報を使用します。例えば、会社のシングルサインオン (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えの詳細については、「 IAMユーザーガイド」の「ユーザーから IAM ロールへの切り替え (コンソール) 」を参照してください。

一時的な認証情報は、 AWS CLI または を使用して手動で作成できます AWS API。その後、これらの一時的な認証情報を使用してアクセスすることができます AWS。 AWS では、長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成することをお勧めします。詳細については、「」の「一時的なセキュリティ認証情報IAM」を参照してください。

AMB クエリのクロスサービスプリンシパル許可

転送アクセスセッションをサポート (FAS): はい

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用すると、別のサービスで別のアクションを開始するアクションを実行できます。 FASは、 を呼び出すプリンシパルのアクセス許可をリクエストと組み合わせて使用し AWS のサービス、ダウンストリームサービス AWS のサービス にリクエストを送信します。 FAS リクエストは、他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストをサービスが受信した場合にのみ行われます。この場合、両方のアクションを実行するための権限が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

AMB クエリのサービスロール

サービスロールのサポート: なし

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロール です。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「 IAMユーザーガイド」の「 にアクセス許可を委任するロールを作成する AWS のサービス」を参照してください。

> **Warning**
>
> サービスロールのアクセス許可を変更すると、AMBクエリ機能が破損する可能性があります。Query AMB が指示する場合以外は、サービスロールを編集しないでください。

AMB クエリのサービスにリンクされたロール

サービスにリンクされたロールのサポート: なし

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、 サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

サービスにリンクされたロールの作成または管理の詳細については、「IAMと連携するAWS サービス」を参照してください。表の中から、[サービスリンクロール] 列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

Amazon Managed Blockchain (AMB) クエリのアイデンティティベースのポリシーの例

デフォルトでは、ユーザーとロールには Query AMB リソースを作成または変更するアクセス許可はありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または を使用してタスクを実行することはできません AWS API。IAM管理者は、リソースで必要なアクションを実行するためのアクセス許可をユーザーに付与する IAMポリシーを作成できます。その後、管理者はロールに IAMポリシーを追加し、ユーザーはロールを引き受けることができます。

これらのポリシードキュメント例を使用してIAMアイデンティティベースのJSONポリシーを作成する方法については、「 IAMユーザーガイド」のIAM「ポリシーの作成 (コンソール) 」を参照してください。

ARNs 各リソースタイプの の形式など、Query AMB で定義されるアクションとリソースタイプの詳細については、「サービス認可リファレンス」の「Amazon Managed Blockchain (AMB) クエリのアクション、リソース、および条件キー」を参照してください。

トピック

- ポリシーのベストプラクティス
- 自分の権限の表示をユーザーに許可する
- 特定の Amazon Managed Blockchain (AMB) クエリAPIアクションへのアクセス

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウントで誰かが Query AMB リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可を付与するには、多くの一般的なユースケースにアクセス許可を付与する AWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義して、アクセス許可をさらに減らすことをお勧めします。詳細については、「IAMユーザーガイド」の「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要なアクセス許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用してアクセス許可を適用する方法の詳細については、IAM ユーザー ガイドの「IAM のポリシーとアクセス許可」を参照してください。
- IAMポリシーで条件を使用してアクセスをさらに制限する – ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを を使用して送信するように指定できますSSL。条件を使用して、サービスアクションが などの特定の を通じて使用される場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます AWS CloudFormation。詳細については、「 IAMユーザーガイド」のIAMJSON「ポリシー要素: 条件」を参照してください。
- IAM Access Analyzer を使用してIAMポリシーを検証し、安全で機能的なアクセス許可を確保する – IAM Access Analyzer は、ポリシーがポリシー言語 (JSON) とIAMベストプラクティスに準拠するように、新規および既存のIAMポリシーを検証します。 IAMAccess Analyzer には、安全で機能的なポリシーの作成に役立つ 100 を超えるポリシーチェックと実用的な推奨事項が用意されています。詳細については、「 IAMユーザーガイド」のIAM「Access Analyzer によるポリシーの検証」を参照してください。
- 多要素認証を要求する (MFA) – でIAMユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、セキュリティを強化MFAするために をオンにします。API オペレー

ションが呼び出されるMFAタイミングを要求するには、ポリシーにMFA条件を追加します。詳細については、「 IAMユーザーガイド」の「 を使用した安全なAPIアクセスMFA」を参照してください。

IAM でのベストプラクティスの詳細については、「IAMユーザーガイド」の「IAM でのセキュリティのベストプラクティス」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザー ID にアタッチされたインラインおよび管理ポリシーの表示を IAM ユーザーに許可するポリシーを作成する方法を示します。このポリシーには、コンソールで、または AWS CLI または を使用してプログラムでこのアクションを実行するアクセス許可が含まれています 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": "*"
        }
    ]
}
```

特定の Amazon Managed Blockchain (AMB) クエリAPIアクションへのアクセス

> **Note**
>
> AMB クエリにアクセスしてAPI呼び出しを行うには、クエリに対する適切なIAMアクセス許可を持つユーザー認証情報 (`AWS_ACCESS_KEY_ID` および `AWS_SECRET_ACCESS_KEY`) AMB が必要です。

Example IAM すべての Amazon Managed Blockchain (AMB) クエリにアクセスするポリシー APIs

この例では、 の IAMユーザーにすべてのAMBクエリ AWS アカウント へのアクセスを許可します APIs。

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

Example IAM Amazon Managed Blockchain (AMB) クエリ**`ListTransactions`**と にアクセスするためのポリシー **`GetTransaction`** APIs

この例では、 クエリおよび AMB AWS アカウント へのアクセスを のIAMユーザーに許可`ListTransaction`します。 `GetTransaction` APIs

> **Note**
>
> この例APIsの を他の に置き換えたり追加したりAPIsして、他の またはそれ以上の へのアクセスを許可できますAPIs。AMB クエリ のリストについてはAPIs、「Amazon Managed Blockchain (AMB) クエリAPIリファレンスガイド」を参照してください。

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

Amazon Managed Blockchain (AMB) クエリのアイデンティティとアクセスのトラブルシューティング

次の情報は、AMBクエリと の使用時に発生する可能性がある一般的な問題の診断と修正に役立ちますIAM。

トピック

- AMB クエリでアクションを実行する権限がない

AMB クエリでアクションを実行する権限がない

「I am not authorized to perform an action in Amazon Bedrock」というエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

次の例のエラーは、mateojacksonIAMユーザーがコンソールを使用して架空の*my-example-widget*リソースの詳細を表示しようとしているが、架空のmanagedblockchain-query::*GetWidget*アクセス許可がない場合に発生します。

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

この場合、managedblockchain-query::*GetWidget* アクションを使用して *my-example-widget*リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

Amazon での Amazon Managed Blockchain (AMB) クエリ API 使用メトリクス CloudWatch

Amazon での API 使用状況メトリクス CloudWatch

Amazon Managed Blockchain (AMB) CloudWatch クエリサービスのクォータに対応するように公開されている API 使用メトリクス。使用量がサービスクォータに近づいたときに警告するようにアラームを設定できます。 CloudWatchサービスクォータとの統合の詳細については、Amazon CloudWatch ユーザーガイドの「AWS 使用状況メトリクス」を参照してください。

AMB Query は、次の API `AWS/Usage` メトリクスをサービス名とともに名前空間に公開します。`Amazon Managed Blockchain Query`

メトリクス	説明
`CallCount`	AMB Query 内の API に対して行われた呼び出しの総数。SUM は、指定した期間に API に対して行われた呼び出しの総数を表します。

Amazon Managed Blockchain (AMB) クエリは、`AWS/Usage`以下のディメンションの使用状況メトリクスを名前空間に公開します。

ディメンション	説明
Service	AWS リソースを含むサービスの名前。`Amazon Managed Blockchain Query` は常にこのディメンションの値になります。
タイプ	報告されるエンティティのタイプ。 APIは常にこのディメンションの値になります。
リソース	報告されるリソースのタイプ。使用する AMB Query API オペレーションの名前がこのディメンションの値になります。

ディメンション	説明
Class	レポート対象リソースのクラス。 Noneは常にこのディメンションの値になります。

『AMB Query ユーザーガイド』のドキュメント履歴

次の表は、AMB Query のドキュメントリリースをまとめたものです。

変更	説明	日付
AMB Query はビットコインのトランザクション ID とトランザクションハッシュをサポートします。	ビットコインネットワークでは、AMB Query API オペレーションはトランザクション識別子 (`transactionId`) とトランザクションハッシュ () の両方をサポートします。`transactionHash`	2024 年 3 月 21 日
Amazon での API 使用状況メトリクスのSupport CloudWatch	AMB Query は API 使用量メトリクスのサポートをに追加しました。 CloudWatchこれらの使用状況メトリックは AMB Query サービスのクォータに対応しています。	2024 年 2 月 8 日
ファイナリティに達していないトランザクションのSupport	AMB Query は、ファイナリティに達していないトランザクションのサポートを追加しました。また、`status`操作の応答からプロパティのサポートが削除されました。`GetTransaction` 代わりに、`confirmationStatus` `executionStatus` およびプロパティを使用してトランザクションのステータスを判断します。	2024 年 2 月 1 日
`status`トランザクションデータ型のプロパティは廃止されました。	Amazon Managed Blockchain (AMB) クエリは、`status`トランザクションデータタイプ	2023 年 12 月 20 日

	のプロパティを廃止しました。`confirmationStatus` `executionStatus` およびフィールドを使用して、トランザクションがまたはかどうかを判断する必要があります。`status` `FINAL` `FAILED`	
セポリアテストネットのSupport	Amazon Managed Blockchain (AMB) クエリが Ethereum Sepolia テストネットでのクエリをサポートするようになりました。	2023 年 10 月 19 日
資産契約のSupport	`ListAssetContracts` API オペレーションを使用して、指定したアドレスでデプロイされたものを一覧表示できます。さらに、コントラクトアドレスがわかっている場合は、`GetAssetContract` API オペレーションを使用してコントラクトの詳細を取得できます。	2023 年 10 月 16 日
ビットコインテストネットのSupport	Amazon Managed Blockchain (AMB) クエリがビットコインテストネットでのクエリをサポートするようになりました。	2023 年 10 月 16 日
初回リリース	AMB クエリサービスの初回リリース。	2023 年 7 月 27 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。