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

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回復力のある Web3 アプリケーションを構築するのに役立つように設計されたフルマネージドサービスです。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にアクセスできます。特殊なブロックチェーンインフラストラクチャをデプロイし、ブロックチェーンネットワークに接続したままにすることなく、Web3-readyアプリケーションを構築します。AMB Query を使用すると、デベロッパー向けの API オペレーションを使用して、複数のブロックチェーンからのリアルタイムデータと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、特殊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、ロード) を必要とせずに、AWS のサービスに統合できます。すべての AMB 機能は、組織のグレードや主要なコンシューマーアプリケーションの構築に合わせて安全に拡張できます。

Amazon Managed Blockchain (AMB) Query は、開発者向けの API オペレーションを使用して、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 クエリを使用すると、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作成インフラストラクチャの維持に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のデータを必要とするアプリケーションをすばやく出荷できます。AMB Query では、機能可能なトークンまたは機能しないトークン (NFTs) のトークン残高の履歴を分析する場合でも、特定のウォレットアドレスのトランザクション履歴を表示する場合でも、Ether などのネイティブ暗号通貨のディストリビューションでデータ分析を実行する場合でも、ブロックチェーンデータにアクセスできます。

AMB クエリを初めてお使いになる方向けの情報

AMB クエリを初めて使用する場合は、まず以下のセクションを読むことをお勧めします。

- 主要な概念: 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

主要な概念: 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、ステーク、マイニング、半分などが含まれます。

Amazon Managed Blockchain (AMB) クエリは、マルチブロックチェーンネットワークデータへの便利なアクセスを提供するため、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簡単に抽出できます。AMB クエリを使用して、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。アドレスの現在および過去の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することもできます。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 クエリは、次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処理されるパブリック 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には実際の値 (つまり、実際のコストが発生します) があり、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、 Mainnet BTC とは別個かつ個別であり、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には実際の値 (つまり、実際のコストが発生します) があり、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 クエリは、次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 契約標準
    - ERC-20 トークン標準 — ERC-20 は、実用的な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、おそらくその経過時間、希少性、またはその他のプロパティが原因で、同じ契約とは異なる値を持つことができます。詳細については、Ethereum.org の ERC-721 トークン標準を参照してください。

      ERC-1155 マルチトークン標準 — ERC-1155 は、任意の数の有効なトークンタイプと無効な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 および 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、明確な実装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 トークン標準を参照してください。

- 確定性

  ブロックチェーンでは、確定性とは、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 では、6 ブロックまたは 60 分のいずれか早い方後にトランザクションが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最終と見なします。

  AMB クエリ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 クエリ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、まだ確定に達していなくても、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 を呼び出す場合、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 承認された 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、 呼び出しで 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 AWS アカウントを作成します。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、 AWS アカウントが必要です。

にサインアップする AWS

Amazon Web Services (AWS) にサインアップすると AWS のサービス、Amazon Managed Blockchain (AMB) クエリを含むすべての に AWS アカウントが自動的にサインアップされます。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。

AWS アカウント をすでにお持ちの場合は、次のステップに進みます。 AWS アカウントをお持ちでない場合は、次に説明する手順に従ってアカウントを作成してください。

AWS アカウントを作成するには

1. https://portal.aws.amazon.com/billing/signup 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、通話呼び出しまたはテキストメッセージを受け取り、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 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。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、ルートユーザーアクセスが必要なタスクの実行にはルートユーザーのみを使用するようにしてください。

適切なアクセス許可を持つ IAM ユーザーを作成する

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

IAM プリンシパルのみが AMB クエリ API リクエストを行うことができます。AMB クエリ APIs を呼び出すときは、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 承認された 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、 呼び出しで 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。

IAM ユーザーを作成する方法については、「アカウントでの IAM ユーザーの作成 AWS」を参照してください。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、「IAM ユーザーのアクセス許可の変更」を参照してください。AMB 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、「」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、最新のコマンドラインインターフェイス (CLI) AWS をインストールして、ターミナルの AWS リソースを操作します。詳細については、「Installing or updating the latest version of the AWS CLI」を参照してください。

> **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、IAM ユーザーガイドの「 AWS リソースでの一時的な認証情報の使用」を参照してください。

AWS Management Console を使用して 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

Amazon Managed Blockchain (AMB) クエリにアクセスし、 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. https://console.aws.amazon.com/managedblockchain/ で Amazon Managed Blockchain コンソールを開きます。

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に関連する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 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step-by-step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、独自の有効なものに置き換えてください。

トピック

- AMB クエリ API オペレーションにアクセスするための 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

AMB クエリ API オペレーションにアクセスするための 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 (AMB) 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_KEY) 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、AMB クエリ API オペレーションにアクセスするための 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、「ロールの作成と IAM ユーザーへの割り当て」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) クエリを使用すると、ブロックチェーンデータがまだ確定していない場合でも、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。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-1155ERC-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 API の詳細については、「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 GetTransaction API 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 (EOA) 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 (ETH と BTC) 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バランスを取得します。

次の例では、 GetTokenBalance API を使用して Ethereum Mainnet のアドレス Ether (ETH) 残高を取得します。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、HTTPS 経由で署名付きリクエストを送信して 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、AXIOS などのサードパーティーライブラリを使用して AMB クエリからデータを取得できます。
> - これらの例では、Node.js にサードパーティーの HTTP クライアントを使用していますが、 AWS JavaScript SDK を使用して AMB クエリにリクエストを行うこともできます。
> - 次の例は、Axios と AWS SigV4 用の SDK モジュールを使用して AMB 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 — AMB クエリ **GetTokenBalance** API を使用して、特定の外部所有アドレス (EOA) からトークン残高の履歴を取得する

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

GetTokenBalance API をテストするには、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 にリストされている EOA 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、HTTPS 経由で SDK for Python を使用して AWS Amazon Managed Blockchain (AMB) クエリ API リクエスト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 `ListTransactionEvents` API 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 で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. https://console.aws.amazon.com/managedblockchain/ で Amazon Managed Blockchain コンソールを開きます。
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. クエリタイプとして GetTokenBalance 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 クエリのユースケースを一覧表示します。

トピック

現在および過去のトークン残高をクエリする

`GetTokenBalance` API は、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) のバランスを取得し、外部所有アカウント (EOAs) 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、現在または過去のバランスを取得します。例えば、 `GetTokenBalance` API オペレーションを使用して、Ethereum Mainnet の ERC20 トークン USDC のアドレスバランスを取得できます。`BatchGetTokenBalance` 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バランスをバッチ取得することもできます。

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。

履歴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、Ethereum や Bitcoin 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、いくつかのユースケースが可能になります。`ListTransactions` API オペレーションを使用して、Ethereum Mainnet 上の特定の外部所有アドレス (EOA) のトランザクションのリストを取得し、次に `GetTransaction` API 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。

特定のアドレスのすべてのトークン残高を取得する

ListTokenBalances API 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バランスを取得できます。この API オペレーションは、単一の 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すべての残高のリストを返します。例えば、外部所有のアドレス (EOA) とネットワーク (Ethereum Mainnet) を指定し、レスポンスでトークンとネイティブのコインバランス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。

トランザクションに対して出力されたイベントを一覧表示する

ListTransactionEvents API オペレーションを使用して、ハッシュ (トランザクション識別子) によって識別される特定のトランザクションの結果として出力される契約イベントのリストを取得できます。例えば、 ListTransactionEventsを使用して、ERC20 契約からの転送イベントや引き出しイベントなど、Ethereum ブロックチェーンで ERC20 トークン契約の関数を呼び出すトランザクションの結果イベントを取得できます。 ERC20

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。

契約で作成されたすべてのトークンを取得する

ListTokenBalances API オペレーションを使用して、契約アドレスを入力として渡したときに、契約によって作成されたすべてのサポートされているトークン (ERC20、ERC721, ERC1155 のリストを返すことができます。例えば、 ListTokenBalances API オペレーションを使用して、Ethereum ブロックチェーンの ERC721 契約標準によって作成された非ファウンブルトークン (NFTs) に関連する情報を取得できます。

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。

契約を一覧表示し、契約情報を取得する

ListAssetContracts API オペレーションを使用して、特定のアドレスによってデプロイされた ERC-721, ERC-1155、または ERC-20 契約を一覧表示できます。さらに、契約アドレスがある場合

は、 GetAssetContract API オペレーションを使用して、契約タイプのデプロイアドレスや関連するトークンメタデータなどの契約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、および契約をクエリするための APIs が含まれます。詳細については、「AMB クエリ API リファレンス」を参照してください。

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

のクラウドセキュリティ AWS は最優先事項です。 AWS のお客様は、セキュリティを最も重視する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からメリットを得られます。

セキュリティは、 AWS とお客様の間で共有される責任です。責任共有モデルでは、これをクラウドのセキュリティとクラウド内のセキュリティの両方と定義しています。

- クラウドのセキュリティ – AWS は、 で AWS 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 クラウド。 AWS また、 は、お客様が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の詳細については、AWS 「コンプライアンスプログラムによる対象範囲内のサービス」を参照してください。
- クラウド内のセキュリティ – お客様の責任は、使用する AWS 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 Managed Blockchain 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を使用します。

このドキュメントは、AMB クエリを使用する際の責任共有モデルの適用方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンスの目的を達成するように AMB クエリを設定する方法について説明します。また、AMB クエリリソースのモニタリングや保護に役立つ他の AWS のサービスの使用方法についても説明します。

トピック

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、ネットワークを移動するときに傍受される可能性のあるデータが含まれます。これは転送中のデータと呼ばれます。

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、 AWS CLI クライアントから AWS 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。

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 AMB クエリリソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで AWS のサービス 使用できる です。

トピック

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与えられます。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すると、管理者に適切なアクセス許可をリクエストするのに役に立ちます。AMB クエリの機能にアクセスできない場合は、「」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、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。会社で AMB クエリで IAM 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 IAM の連携方法。

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。IAM で使用できる AMB クエリ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。

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 ロールを引き受けて認証 (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 認証情報は、フェデレーティッド ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーションを使用して にアクセスすると、間接的 AWS にロールを引き受けます。

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 Software Development Kit (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。リクエストに自分で署名する推奨方法の使用については、「IAM ユーザーガイド」の「API リクエストに対するAWS Signature Version 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「IAM ユーザーガイド」の「IAM のAWS 多要素認証」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「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティ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が、一時的な認証情報 AWS のサービス を使用して にアクセスするために ID プロバイダーとのフェデレーションを使用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 にアクセスするすべてのユーザーです。フェデレーティッド ID が にアクセスすると 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 ソース内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「What is IAM Identity Center?」(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保有する IAM ユーザーを作成する代わりに、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 ユーザーでの長期的な認証情報が必要な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 ユーザーの集団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、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

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 ユーザーに関するユースケース」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、特定のユーザーには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 ロールに切り替えることができます (コンソール) 。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 URL を使用します。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ための各種方法」を参照してください。

IAM ロールと一時的な認証情報は、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については、「IAM 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) 用のロールを作成する」を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。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 ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで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、別のアカウントの人物 (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権を付与する主な方法は、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における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「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

- クロスサービスアクセス — 一部の では、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、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。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際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストをリクエストする を組み合わせて使用します。FAS リクエストは、サービスが他の 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受け取っ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アクセス許可が必要です。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、削除することができます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに許可を委任するロールを作成する」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、サービスリンク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 EC2 で実行されているアプリケーション – 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 リクエストを実行している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的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」を参照してください。

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には AWS 、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 ポリシー概要を参照してください。

管理者は JSON AWS ポリシーを使用して、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権限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 API からロール情報を取得できます。

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ポリシーは、IAM ユーザーグループ、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 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ポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム 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のポリシーは、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。マネージドポリシーまたはインラインポリシーのいずれか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いずれかを選択する」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、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。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制御できます。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では、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、 AWS WAF、および Amazon VPC は、ACLs。ACL の詳細については、「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ています。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ポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限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 (SCP) を一部またはすべてのアカウントに適用できます。SCP 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を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー (SCP)」を参照してください。

- リソースコントロールポリシー (RCP) – RCP は、所有する各リソースにアタッチされた IAM 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利用可能な最大数のアクセス許可を設定するために使用できる JSON ポリシーです。RCP は、メンバーアカウントのリソースのアクセス許可を制限し、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。RCP をサポートする のリストを含む Organizations と RCP の詳細については、AWS Organizations 「 ユーザーガイド」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。 RCPs
- 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 がリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

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

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 機能について学びます。

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

AMB クエリおよびその他の 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、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」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

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ポリシーは、IAM ユーザーグループ、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 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 ベースのポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM 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、およびアクションを許可または拒否す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学ぶには、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」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 クエリの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、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。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制御できます。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、アカウント全体、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、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も付与する必要があります。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「IAM でのクロスアカウントリソースアクセス」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 JSON AWS ポリシーを使用して、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 要素にはポリシー内の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が記述されます。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 オペレーションのない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アクセス許可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。

AMB クエリ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 クエリの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

ポリシーリソースのサポート: なし

管理者は JSON AWS ポリシーを使用して、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。

Resource JSON ポリシー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 ARNs「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。 各リソースの ARN 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 クエリの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へのサポート: なし

管理者は JSON AWS ポリシーを使用して、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して、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。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、「IAM ユーザーガイド」の「IAM ポリシーの要素: 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) クエリで定義されるアクション」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 ACLs

ACL のサポート: なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

AMB クエリでの ABAC

ABAC (ポリシー内のタグ) のサポート: なし

属性ベースのアクセス制御 (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、ABAC の最初の手順です。その後、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した場合にオペレーションを許可するように ABAC ポリシーをします。

ABAC は、急成長す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。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM 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。

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 AWS のサービス を使用する機能などの詳細については、AWS のサービス 「IAM ユーザーガイド」の「IAM と連携する 」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は、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。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ユーザーから IAM ロールに切り替える (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 AWS API を使用して手動で作成できます。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。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 AWS を動的に生成することをお勧めします。詳細については、「IAM の一時的セキュリティ認証情報」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 (FAS) のサポート: あり

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際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストのリクエストリクエストを使用します。FAS リクエストは、サービスが他の 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受け取っ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アクセス許可が必要です。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「AWS のサービスに許可を委任するロールを作成する」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。AMB クエリ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 と提携するAWS のサービス」を参照してください。表の「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 AMB クエリ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 AWS API を使用してタスクを実行することはできません。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権限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継ぐことができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM 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「IAM ポリシーを作成する (コンソール)」を参照してください。

各リソースタイプの ARNs「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

- ポリシーに関するベストプラクティス
- ユーザーが自分の権限を表示できるようにする
- 特定の Amazon Managed Blockchain (AMB) クエリ API アクションへのアクセス

ポリシーに関する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 AMB クエリリソースを作成、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「ジョブ機能のAWS マネージドポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な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 で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。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。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「IAM JSON 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、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、「IAM ユーザーガイド」の「IAM Access Analyzer でポリシーを検証する」を参照してください。
- 多要素認証 (MFA) を要求する – 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するために MFA を有効にします。API オペレー

ションが呼び出されるときに MFA を必須にするには、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「MFA を使用した安全な API アクセス」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 でのセキュリティベストプラクティス」を参照してください。

ユーザーが自分の権限を表示できるようにする

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API 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。

```
{
    "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 コールを行うには、AMB クエリに適切な 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`) が必要です。

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

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

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

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

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

> **Note**
>
> この例APIs を他の API に置き換えたり、追加したりして、他の APIs または複数の APIs へのアクセスを許可したりできます。AMB クエリ 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 IAM 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。

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

アクションを実行する権限がないというエラーが表示された場合は、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。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要なmanagedblockchain-query::*GetWidget* アクセス許可を持っていない場合に発生するものです。

```
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 CloudWatch での Amazon Managed Blockchain (AMB) クエリ API 使用状況メトリクス

Amazon CloudWatch の API 使用状況メトリクス

CloudWatch に発行される API 使用状況メトリクスは、Amazon Managed Blockchain (AMB) クエリサービスクォータに対応しています。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 とサービスクォータ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、 `AWS/Usage`名前空間に次の API メトリクスを`Amazon Managed Blockchain Query`サービス名とともに発行します。

メトリクス	説明
`CallCount`	AMB クエリで API に対して行われた呼び出しの合計数。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、以下のディメンションを使用して、使用状況メトリクスを `AWS/Usage`名前空間に発行します。

ディメンション	説明
Service	リソースを含む AWS 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です。
タイプ	レポート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です。
リソース	レポートされるリソースのタイプ。使用される AMB クエリ API オペレーションの名前は、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	レポートされるリソースのクラス。 Noneは常にこのディメンションの値です。

AMB クエリユーザーガイドのドキュメント履歴

次の表に、AMB クエリのドキュメントリリースを示します。

変更	説明	日付
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` 。	2024 年 3 月 21 日
Amazon CloudWatch での API 使用状況メトリクスのサポート	AMB Query に、CloudWatch での API 使用状況メトリクスのサポートが追加されました。これらの使用状況メトリクスは、AMB クエリサービスクォータに対応しています。	2024 年 2 月 8 日
確定性に達していないトランザクションのサポート	AMB Query は、確定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、 `GetTransaction` オペレーションのレスポンスから`status`プロパティのサポートも削除されます。代わりに、 プロパティ`confirmationStatus` と `executionStatus` 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
トランザクションデータ型での `status`プロパティの非推奨化	Amazon Managed Blockchain (AMB) Query は、トランザクションデータ型の	2023 年 12 月 20 日

	statusプロパティを廃止しました。confirmationStatus および executionStatus フィールドを使用して、トランザクションstatusの が FINALかかを判断する必要がありますFAILED。	
Sepolia Testnet のサポート	Amazon Managed Blockchain (AMB) Query で、Ethereum Sepolia Testnet でのクエリがサポートされるようになりました。	2023 年 10 月 19 日
アセット契約のサポート	ListAssetContracts API オペレーションを使用して、特定のアドレスによってデプロイされた を一覧表示できます。さらに、契約住所がある場合は、GetAssetContract API オペレーションを使用して契約の詳細を取得できます。	2023 年 10 月 16 日
Bitcoin Testnet のサポート	Amazon Managed Blockchain (AMB) クエリが Bitcoin Testnet 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。