aws

開發人員指南

Amazon Managed Blockchain Query

Amazon Managed Blockchain Query: 開發人員指南

Table of Contents

什麼是 Amazon Managed Blockchain（AMB）查詢？ 1
您是第一次使用 AMB 查詢的使用者嗎？ 1
重要概念 2
使用 Amazon Managed Blockchain (AMB) 查詢的考量和限制 2
設定 5
先決條件和考量事項 5
註冊成為 AWS 5
建立具有適當權限的 IAM 使用者 5
安裝和配置 AWS Command Line Interface 6
使用 AWS Management Console 來使用 AMB 查詢查詢區塊鏈 6
開始使用 7
建立 IAM 政策 7
使用 Go 的示例 8
使用 Node.js 的範例 14
使用 Python 的例子 18
使用範例 AWS Management Console 20
AMB 查詢使用案例 21
查詢當前和歷史令牌餘額 21
檢索歷史交易數據 21
獲取給定地址的所有令牌餘額 21
列出針對交易發出的事件 22
獲取合同鑄造的所有代幣 22
列出合約並取得合約資訊 22
AMB 查詢 API 參考資料 23
安全 24
資料加密 24
傳輸中加密 24
身分與存取管理 25
物件 25
使用身分驗證 25
使用政策管理存取權 28
亞馬遜託管區塊鏈（AMB）查詢如何與 IAM 30
身分型政策範例 36
故障診斷 39

API 使用量度 40
Amazon 上的 API 使用量指標 CloudWatch 40
文件歷史紀錄 41
..... xliii

什麼是 Amazon Managed Blockchain（AMB）查詢？

Amazon Managed Blockchain (AMB) 是一項全受管服務，旨在協助您在公有和私有區塊鏈上建置彈性 Web3 應用程式。使用 AMB Access 即時和無服務器訪問多個區塊鏈。無需部署專門的區塊鏈基礎架構並使其與區塊鏈網路保持連線，即可建置適用於 Web3 的應用程式。借助 AMB Query，您可以使用對開發人員友好的 API 操作來訪問來自多個區塊鏈的實時和歷史數據。標準化區塊鏈資料可與 AWS 服務整合，無需專門的區塊鏈基礎設施或 ETL (擷取、轉換和載入)。所有 AMB 功能都可針對機構級和主流消費者應用程式組建安全地擴充。

Amazon Managed Blockchain (AMB) 查詢提供無伺服器存取標準化的多區塊鏈資料集，並提供適合開發人員使用的 API 操作。您可以使用 AMB Query 快速交付需要來自一個或多個公共區塊鏈數據的應用程序，而無需開銷來解析區塊鏈數據，跟踪合同並維護專門的索引基礎設施。無論您是分析可替代令牌還是不可替代令牌（NFT）的歷史令牌餘額，查看給定錢包地址的交易歷史記錄，還是對以太幣等本地加密貨幣的分佈進行數據分析，AMB Query 都可以讓您訪問區塊鏈數據。

您是第一次使用 AMB 查詢的使用者嗎？

如果您是 AMB 查詢的初次使用者，建議您先閱讀以下章節：

- 關鍵概念：Amazon Managed Blockchain（AMB）查詢
- 設置 Amazon Managed Blockchain（AMB）查詢
- 開始使用 Amazon Managed Blockchain (AMB) 查詢
- Amazon Managed Blockchain (AMB) 查詢的使用案例

關鍵概念：Amazon Managed Blockchain（AMB）查詢

> **Note**
>
> 本指南假設您熟悉基本的區塊鏈概念。這些概念包括去中心化，令牌，合同，交易 proof-of-work，錢包，公鑰和私鑰，抵押，採礦，減半等。

Amazon Managed Blockchain (AMB) 查詢可讓您方便地存取多區塊鏈網路資料，讓您輕鬆擷取與區塊鏈活動相關的情境資料。您可以使用 AMB 查詢從公共區塊鏈網絡（例如比特幣主網和以太坊主網）讀取數據。您還可以獲取信息，例如地址的當前和歷史餘額，或者您可以獲取給定時間段內的區塊鏈交易列表。此外，您可以取得指定交易的詳細資訊，例如交易事件，您可以進一步分析或在應用程式的商務邏輯中使用這些事件。

使用 Amazon Managed Blockchain (AMB) 查詢的考量和限制

使用 AMB 查詢時，請考慮下列事項：

- 可用地區

  美國東部 (維吉尼亞北部) `us-east-1` 區域支援 AMB 查詢。

- 服務端點

  AMB 查詢可透過下列端點存取：

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

- 支援區塊鏈網路

  AMB 查詢支持以下公共區塊鏈網絡：

  - 比特幣主網 — 通過 proof-of-work 共識保護的公共比特幣區塊鏈網絡，並在其上發行和交易比特幣（BTC）加密貨幣。主網上的交易具有實際價值（即它們產生實際成本），並記錄在公共區塊鏈上。
  - 比特幣測試網-比特幣主網的測試網。該網絡上的比特幣（BTC）與主網 BTC 是分開的，並且通常沒有任何價值。

- 以太坊 proof-of-stake 主網 — 公共以太坊區塊鏈的主要網絡。主網上的交易具有實際價值（即它們產生實際成本），並記錄在分佈式分類帳中。
- Sepolia 測試網 — 以太坊主網的測試網。該網絡上的以太幣（ETH）與主網 ETH 分開且不同，通常沒有任何價值。

- 支持的區塊鏈代幣和合約

  AMB 查詢支持以下本地和標準以太坊合約令牌。

  - 公共區塊鏈本機令牌
    - 比特幣（BTC）— 這是與比特幣相關的區塊鏈的本機令牌。
    - 以太幣（ETH）— 這是以太坊相關區塊鏈的本機令牌。

  - 以太坊合約標準
    - ERC-20 令牌標準 — ERC-20 是可替代令牌的標準。它有一個屬性，使每個 ERC-20 令牌與另一個 ERC-20 令牌完全相同（類型和值），這意味著一個令牌是並且將始終等於所有其他令牌。有關更多信息，請參閱以太坊上的 ERC-20 令牌標準。
    - ERC-721 不可替代令牌標準 — ERC-721 是不可替代令牌（NFT）的標準。這種類型的令牌是唯一的，可能與同一合同中的另一個令牌具有不同的值，可能是由於其年齡，稀有性或其他屬性。有關更多信息，請參閱以太坊上的 ERC-721 令牌標準。

      ERC-1155 多令牌標準 — ERC-1155 是創建合同接口的標準，可以表示和控制任意數量的可替代和不可替代令牌類型。通過這種方式，ERC-1155 令牌可以與 ERC-20 和 ERC-721 令牌相同的功能，甚至可以同時作為兩者兼容。ERC-1155 令牌改進了 ERC-20 和 ERC-721 標準的功能，使其更有效率，同時糾正明顯的實施錯誤。有關更多信息，請參閱以太坊上的 ERC-1155 令牌標準。
- 終局性

  在區塊鏈中，最終性意味著有效交易不太可能被撤銷。對於比特幣主網，AMB 查詢在 6 個區塊之後考慮最終交易。對於比特幣測試網，它將在 6 個區塊或 60 分鐘之後（以先到者為準）認為最終交易。對於支持的以太坊網絡，AMB 查詢將交易視為 64 個塊之後的最終交易。

  AMB 查詢的令牌餘額和合同 API 操作僅返回已達到特定性的數據。但是，AMB Query 的交易和交易事件 API 操作可以返回在區塊鏈網絡上確認的交易的數據，即使它們尚未達到最終性。

- 不支援空位址

  AMB 查詢不支援 NULL (0x0000000000000000000000000000000000000000) 位址。

- 簽名版本 4 簽署 API 調用

  對 AMB 查詢 API 進行呼叫時，您可以透過使用簽章第 4 版簽署程序驗證的 HTTPS 連線進行呼叫。這表示只有 AWS 帳戶中獲得授權的 IAM 主體才能進行 AMB 查詢 API 呼叫。若要這麼做，呼叫時必須提供 AWS 認證 (存取金鑰 ID 和秘密存取金鑰)。

  > **Important**
  >
  > 請勿在使用者對應的應用程式中內嵌用戶端認證

- AMB 查詢支持比特幣交易標識符和交易哈希

  對於比特幣網絡，AMB 查詢 API 操作同時支持交易標識符（`transactionId`）和交易哈希（`transactionHash`）。`transactionId`這是不包括證人數據的交易的雙 SHA 哈希。`transactionHash`這是交易的雙 SHA 哈希，包括見證數據（也稱為見證交易 ID）。

  呼叫比特幣網路的`GetTransaction`或 `ListTransactionEvents`API 作業時，您可以指定`transactionId`或. 此外，在比特幣網絡上返回 a `transactionHash` 或 a 的`transactionId`所有 AMB 查詢操作都`transactionHash`將包含這兩個值作為響應的一部分。

設置 Amazon Managed Blockchain（AMB）查詢

第一次使用 Amazon Managed Blockchain (AMB) 查詢之前，請按照本節中的步驟建立 AWS 帳戶。下一節討論如何開始使用 AMB 查詢。

先決條件和考量事項

在您第一次使用 Amazon Web Services 之前，您必須擁有一個 AWS 帳戶。

註冊成為 AWS

當您註冊 Amazon Web Services (AWS) 時，您的 AWS 帳戶會自動註冊所有帳戶 AWS 服務，包括 Amazon Managed Blockchain (AMB) 查詢。您只需針對所使用的服務付費。

如果您已 AWS 帳戶 經擁有，請轉到下一步。如果您還沒有 AWS 帳戶，請使用下列程序建立新帳戶。

建立 AWS 帳號

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

建立具有適當權限的 IAM 使用者

若要建立並使用 AMB Query，您必須建立具有允許必要管理區塊鏈動作的權限的 AWS Identity and Access Management (IAM) 主體 (使用者或群組)。

只有 IAM 主體可以發出 AMB 查詢 API 請求。對 AMB 查詢 API 進行呼叫時，您可以透過使用簽章第 4 版簽署程序驗證的 HTTPS 連線進行呼叫。這表示只有 AWS 帳戶中獲得授權的 IAM 主體才能進行 AMB 查詢 API 呼叫。若要這麼做，呼叫時必須提供 AWS 認證 (存取金鑰 ID 和秘密存取金鑰)。

如需如何建立 IAM 使用者的詳細資訊，請參閱在您的 AWS 帳戶中建立 IAM 使用者。如需如何將許可政策附加至使用者的詳細資訊，請參閱變更 IAM 使用者的許可。如需可用來授與使用者使用 AMB Query 之權限原則的範例，請參閱Amazon Managed Blockchain () AMB 查詢的身分型政策範例。

安裝和配置 AWS Command Line Interface

如果您尚未這樣做，請安裝最新的 AWS 命令列介面 (CLI) 以使用終端機中的 AWS 資源。如需詳細資訊，請參閱安裝或更新最新版本的 AWS CLI。

> **Note**
>
> 對於 CLI 存取，您需要存取金鑰 ID 和私密存取金鑰。盡可能使用臨時憑證，而不是長期存取金鑰。臨時憑證包含存取金鑰 ID、私密存取金鑰，以及指出憑證何時到期的安全符記。如需詳細資訊，請參閱 IAM 使用者指南中的將臨時登入資料與 AWS 資源搭配使用。

使用 Amazon Managed Blockchain 塊鏈（AMB）查詢查詢 AWS Management Console 來查詢區塊鏈

您可以使用存取 Amazon Managed Blockchain (AMB) 查詢，並在支援的區塊鏈網路上進行查詢。AWS Management Console以下步驟顯示如何執行此操作：

1. 開啟 Amazon Managed Blockchain 主控台，網址為 https://console.aws.amazon.com/managedblockchain/。
2. 從查詢部分選擇查詢編輯器。
3. 從支援的區塊鏈網路中選擇一個。
4. 選擇您要執行的「查詢」類型。
5. 輸入您選取的「查詢」類型和「執行查詢」的相關參數。

AMB 查詢將運行您的查詢，您將在查詢結果窗口中看到結果。

開始使用 Amazon Managed Blockchain (AMB) 查詢

使用本節中的 step-by-step 教學課程，了解如何使用 Amazon Managed Blockchain (AMB) 查詢執行任務。這些程序需要一些先決條件。如果您是 AMB 查詢的新手，則可以查看本指南的「設置」部分。如需詳細資訊，請參閱 設置 Amazon Managed Blockchain（AMB）查詢。

> **Note**
>
> 這些範例中的某些變數已被故意混淆。在運行這些示例之前，請將它們替換為您自己的有效內容。

主題

- 建立身分與存取權管理政策以存取 AMB 查詢 API 作業
- 使用 Go 提出 Amazon Managed Blockchain (AMB) 查詢 API 請求
- 使用 Node.js 提出 Amazon Managed Blockchain (AMB) 查詢 API 請求
- 使用 Python 提出 Amazon Managed Blockchain (AMB) 查詢 API 請求
- 使用 Amazon Managed Blockchain（AMB）查詢 AWS Management Console 來運行操作 GetTokenBalance

建立身分與存取權管理政策以存取 AMB 查詢 API 作業

若要提出 AMB 查詢 API 請求，您必須使用具有適當 IAM 許可的使用者登入資料 (AWS_ACCESS_KEY_ID 和 AWS_SECRET_KEY_KEY)，這些登入資料具有適當的 Amazon Managed Blockchain (AMB) 查詢。在 AWS CLI 已安裝的終端機中，執行下列命令以建立 IAM 政策以存取 AMB 查詢 API 作業：

```
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
```

建立政策後，請將該政策附加到 IAM 使用者的角色以使其生效。在中 AWS Management Console，導覽至 IAM 服務，並將政策附加AmazonManagedBlockchainQueryAccess到指派給將使用該服務的 IAM 使用者的角色。如需詳細資訊，請參閱建立角色和指派給 IAM 使用者。

> **Note**
>
> AWS 建議您授予特定 API 作業的存取權，而不是使用萬用字元*。如需詳細資訊，請參閱 存取特定的 Amazon Managed Blockchain (AMB) 查詢API動作。

使用 Go 提出 Amazon Managed Blockchain (AMB) 查詢 API 請求

使用 Amazon Managed Blockchain (AMB) 查詢，您可以建立依賴於在區塊鏈上確認區塊鏈資料之後立即存取的應用程式，即使該資料尚未達到最終性。AMB Query 可啟用多種使用案例，例如填入錢包的交易歷史記錄、根據交易雜湊提供有關交易的內容資訊，或取得原生權杖以及 ERC-721、ERC-1155 和 ERC-20 記號的餘額。

下列範例會以 Go 語言建立，並使用 AMB 查詢 API 作業。如需 Go 的詳細資訊，請參閱 Go 文件。如需 AMB 查詢 API 的詳細資訊，請參閱 Amazon Managed Blockchain (AMB) 查詢 API 參考文件。

以下示例使用ListTransactions和 GetTransaction API 操作首先獲取 Ethereum Mainnet 上給定外部擁有地址（EOA）的所有交易列表，然後下一個示例從列表中檢索單個交易的詳細信息。

Example — 使用圍棋進行 **`ListTransactions`** API 操作

將下列程式碼複製到ListTransactions目錄listTransactions.go中名為的檔案。

```
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)
    }
}
```

儲存檔案之後，請使用ListTransactions目錄內的下列命令來執行程式碼：`go run listTransactions.go`。

接下來的輸出如下所示：

```
{
  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 — 通過使用轉到進行 **`GetTransaction`** API 操作

此範例會使用先前輸出的交易雜湊。將下列程式碼複製到GetTransaction目錄`GetTransaction.go`中名為的檔案。

```
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)
    }
}
```

儲存檔案之後，請使用GetTransaction目錄內的下列命令來執行程式碼：go run GetTransaction.go。

接下來的輸出如下所示：

```
{
  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
  }
}
```

該 GetTokenBalance API 為您提供了一種獲取本地令牌（ETH 和 BTC）餘額的方法，該平衡可用於在某個時間點獲取外部擁有帳戶（EOA）的當前餘額。

Example — 使用 **GetTokenBalance** API 動作在 Go 中獲取本機令牌的餘額

在以下示例中，您使用 GetTokenBalance API 獲取以太坊主網上的地址以太幣（ETH）餘額。將下列程式碼複製到GetTokenBalance目錄GetTokenBalanceEth.go中名為的檔案。

```
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)
    }
}
```

儲存檔案之後，請使用GetTokenBalance目錄內的下列命令來執行程式碼：`go run GetTokenBalanceEth.go`。

接下來的輸出如下所示：

```
{
  AtBlockchainInstant: {
    Time: 2020-12-05 11:51:01 +0000 UTC
  },
  Balance: "4343260710",
  LastTransactionHash:
 "0x00000ce94398e56641888f94a7d586d51664eb9271bf2b3c48297a50a0711111",
  LastTransactionTime: 2023-03-14 18:33:59 +0000 UTC,
  OwnerIdentifier: {
    Address: "0x12345d31750D727E6A3a7B534255BADd********"
  },
  TokenIdentifier: {
    Network: "ETHEREUM_MAINNET",
    TokenId: "eth"
  }
}
```

使用 Node.js 提出 Amazon Managed Blockchain (AMB) 查詢 API 請求

若要執行這些節點範例，請套用下列先決條件：

1. 您的電腦上必須安裝節點版本管理員 (nvm) 和 Node.js。您可以在這裡找到您的操作系統的安裝說明。
2. 使用指`node --version`令並確認您使用的是節點版本 14 或更高版本。如果需要，您可以使用`nvm install 14`命令，然後使用`nvm use 14`命令來安裝版本 14。
3. 環境變數`AWS_ACCESS_KEY_ID`和`AWS_SECRET_ACCESS_KEY`必須包含與帳戶相關聯的認證。

   使用下列命令將這些變數匯出為用戶端上的字串。使用 IAM 使用者帳戶中的適當值取代下列中反白顯示的值。

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

> **Note**
>
> - 完成所有先決條件後，您可以透過 HTTPS 提交已簽署的請求，以存取 Amazon Managed Blockchain (AMB) 查詢 API 操作並使用 Node.js 中的原生 https 模組提出請求，或者您也可以使用第三方程式庫 (例如 AXIOS) 並從 AMB 查詢擷取資料。
> - 這些範例會針對 Node.js 使用協力廠商 HTTP 用戶端，但您也可以使用 AWS JavaScript SDK 向 AMB 查詢發出要求。
> - 下列範例說明如何使用 Axios 和適用於 Sigv4 的 AWS SDK 模組來提出 AMB 查詢 API 要求。

將下列`package.json`檔案複製到本機環境的工作目錄中：

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

Example — 通過使用 AMB 查詢 API 從特定的外部擁有地址（EOA）檢索歷史令牌餘額 **`GetTokenBalance`**

您可以使用 `GetTokenBalance` API 獲取各種令牌（例如 ERC20，ERC721 和 ERC1155）和本機硬幣（例如 ETH 和 BTC）的餘額，您可以使用它們根據歷史記錄（Unix 時間戳-秒）獲取外部擁有帳戶`timestamp`（EOA）的當前餘額。在此示例中，您使用 `GetTokenBalance`API 獲取以太坊主網上 ERC20 令牌 USDC 的地址餘額。

若要測試 GetTokenBalance API，請將下列程式碼複製到名為的檔案中token-balance.js，然後將檔案儲存到相同的工作目錄中：

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

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

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

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

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


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

  try {
```

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

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


}


let methodArg = 'get-token-balance';

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

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

若要執行程式碼，請在與檔案相同的目錄中開啟終端機，然後執行下列命令：

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

此命令運行腳本，傳遞代碼中定義的參數，以請求以太坊主網上列出的 EOA 的 ERC20 USDC 餘額。回應類似如下：

```
 {
  atBlockchainInstant: { time: 1688076218 },
```

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

使用 Python 提出 Amazon Managed Blockchain (AMB) 查詢 API 請求

若要執行這些 Python 範例，請套用下列先決條件：

1. 您必須在計算機上安裝 Python。您可以在這裡找到您的操作系統的安裝說明。
2. 安裝適用於 Python 的 AWS 開發套件。
3. 安裝命AWS 令列介面並執行指令`aws configure`來設定`Access Key ID`、`Secret Access Key`、和的變數`Region`。

完成所有先決條件之後，您可以使用透過 HTTPS 進行 Python 專用的 AWS 開發套件來進行 Amazon Managed Blockchain (AMB) 查詢 API 請求。

下列 Python 範例會使用 boto3 中的模組，將貼有必要 Sigv4 標頭的要求傳送至 AMB 查詢 API 作業。`ListTransactionEvents`此示例檢索以太坊主網上給定交易發出的事件列表。

將下列`list-transaction-events.py`檔案複製到本機環境的工作目錄中：

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

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

    session = Session()
    sigv4 = SigV4Auth(session.get_credentials(), service, region)
    data = json.dumps(params)
    request = AWSRequest(method, url, data=data)
    sigv4.add_auth(request)
```

```
    http_session = URLLib3Session()
    response = http_session.send(request.prepare())

    return(response)

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

# Call the listTransactionEvents operation. This operation will return transaction
 details for
# all transactions that are con#rmed on the blockchain, even if they have not reached
# #nality.
listTransactionEvents = signed_request(url, method, params, service, region)

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

若要執行範例程式碼ListTransactionEvents，請將檔案儲存在工作目錄中，然後執行命令python3 list-transaction-events.py。此命令運行腳本，傳遞代碼中定義的參數，以請求與以太坊主網上給定交易哈希相關聯的事件。回應類似如下：

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

使用 Amazon Managed Blockchain（AMB）查詢 AWS Management Console 來運行操作 GetTokenBalance

以下示例演示瞭如何使用 Amazon Managed Blockchain（AMB）查詢在以太坊主網上獲得令牌的餘額 AWS Management Console

Example

1. 開啟 Amazon Managed Blockchain 主控台，網址為 https://console.aws.amazon.com/managedblockchain/。
2. 從查詢部分選擇查詢編輯器。
3. 選擇以太坊主網作為區塊鏈網絡。
4. 選擇GetTokenBalance作為「查詢」類型。
5. 輸入代幣的區塊鏈地址。
6. 輸入權杖的「合約地址」。
7. 輸入權杖的選擇性變數記號 ID。
8. 選擇權杖餘額的「開始日期」。
9. 輸入權杖餘額的選擇性「在時間」。
10. 選擇 Run query (執行查詢)。

AMB 查詢將運行您的查詢，您將在查詢結果窗口中看到結果。

Amazon Managed Blockchain (AMB) 查詢的使用案例

本主題提供 AMB 查詢使用案例的清單。

主題

- 查詢當前和歷史令牌餘額
- 檢索歷史交易數據
- 獲取給定地址的所有令牌餘額
- 列出針對交易發出的事件
- 獲取合同鑄造的所有代幣
- 列出合約並取得合約資訊

查詢當前和歷史令牌餘額

該 `GetTokenBalance`API 獲取支持的令牌（ERC20，ERC721，ERC1155）和本機硬幣（ETH，BTC）的餘額，以通過使用外部擁有帳戶（EOA）的通用時間戳（Unix 時間戳，以秒為單位）來獲取當前或歷史餘額。例如，您可以使用 `GetTokenBalance` API 操作在以太坊主網上獲取 ERC20 令牌 USDC 的地址餘額。您還可以使用 `BatchGetTokenBalance` API 操作批量檢索令牌和本地硬幣的餘額。

如需詳細資訊，請參閱 Amazon Managed Blockchain (AMB) 查詢參考指南。

檢索歷史交易數據

使用 Amazon Managed Blockchain（AMB）查詢，您可以從以太坊和比特幣等公共區塊鏈中檢索歷史數據。此功能可啟用多種使用案例，例如擷取區塊鏈錢包上的交易歷史記錄，或根據交易雜湊提供有關交易的內容資訊。您可以使用 `ListTransactions`API 操作獲取以太坊主網上給定外部擁有地址（EOA）的交易列表，然後您可以使用 `GetTransaction`API 操作從列表中檢索單個交易的交易詳細信息。

如需詳細資訊，請參閱 Amazon Managed Blockchain (AMB) 查詢參考指南。

獲取給定地址的所有令牌餘額

您可以使用 `ListTokenBalances`API 操作來獲取錢包，用戶界面，web3 實用程序等的餘額。此 API 操作通過使用單個 API 操作返回給定公共區塊鏈上令牌（ERC20，ERC721，ERC1155）和本地硬幣

（ETH，BTC）的地址的所有餘額列表。例如，您可以提供外部擁有的地址（EOA）和網絡（以太坊主網），並且可以在響應中收到令牌和本地硬幣餘額的列表。

如需詳細資訊，請參閱 Amazon Managed Blockchain (AMB) 查詢參考指南。

列出針對交易發出的事件

您可以使用 `ListTransactionEvents`API 作業擷取由指定交易結果發出的合約事件清單，這些合約事件由其雜湊 (交易識別碼) 識別。例如，您可以使用`ListTransactionEvents`來檢索在以太坊區塊鏈上調用 ERC20 令牌合約函數的交易的結果事件，例如轉移事件或 ERC20 合約中的提款事件。

如需詳細資訊，請參閱 Amazon Managed Blockchain (AMB) 查詢參考指南。

獲取合同鑄造的所有代幣

您可以使用 `ListTokenBalances`API 操作返回合同地址作為輸入時，由合同鑄造的所有支持令牌（ERC20，ERC721，ERC1155）的列表。例如，您可以使用 API 操作檢索與以太坊區塊鏈上 ERC721 合約標準鑄造的不可替代令牌（NFT）相關的信息。`ListTokenBalances`

如需詳細資訊，請參閱 Amazon Managed Blockchain (AMB) 查詢參考指南。

列出合約並取得合約資訊

您可以使用 `ListAssetContracts`API 作業列出由指定位址部署的 ERC-721、ERC-1155 或 ERC-20 合約。此外，如果您有合約位址，則可以使用 `GetAssetContract`API 作業擷取合約的屬性，例如合約類型部署程式位址和相關的 Token 中繼資料。

如需詳細資訊，請參閱 Amazon Managed Blockchain (AMB) 查詢參考指南。

Amazon Managed Blockchain (AMB) 查詢 API 參考

Amazon Managed Blockchain (AMB) 查詢提供 API 操作，以查詢支援的區塊鏈。這包括用於查詢令牌，交易和合同的 API。如需詳細資訊，請參閱 AMB 查詢 API 參考資料。

Amazon Managed Blockchain (AMB) 查詢中的安全性

雲安全性 AWS 是最高優先級的。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，這些架構是為了滿足對安全性最敏感的組織的需求而建置的。

安全是 AWS 與您之間共同承擔的責任。共同的責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端安全性 — AWS 負責保護中執行 AWS 服務的基礎架構 AWS 雲端。 AWS 還為您提供可以安全使用的服務。在 AWS 合規計劃中，第三方稽核員會定期測試並驗證我們的安全功效。若要了解適用於 Amazon Managed Blockchain (AMB) 查詢的合規計劃，請參閱合規計劃適用範圍的AWS 服務。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

為了提供資料保護、身分驗證和存取控制，Amazon Managed Blockchain 使用 AWS 在受管區塊鏈中執行的開放原始碼架構的功能和功能。

本文件可協助您瞭解如何在使用 AMB 查詢時套用共同的責任模型。下列主題說明如何設定 AMB 查詢以符合安全性與合規性目標。您還可以了解如何使用其他 AWS 服務來幫助您監視和保護 AMB 查詢資源。

主題

- 資料加密
- Amazon 受管區塊鏈的身分和存取管理 (AMB) 查詢

資料加密

資料加密有助於防止未經授權的使用者從區塊鏈網路和相關資料儲存系統讀取資料。這包括在網路傳輸時可能遭到攔截的資料，稱為傳輸中的資料。

傳輸中加密

默認情況下，託管區塊鏈使用 HTTPS/TLS 連接來加密從客戶端傳輸到 AWS 服務 AWS CLI 端點的所有數據。

Amazon 受管區塊鏈的身分和存取管理 (AMB) 查詢

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM管理員控制誰可以驗證 (登入) 和授權 (有權限) 使用AMB查詢資源。IAM是您 AWS 服務 可以免費使用的。

主題

- 物件
- 使用身分驗證
- 使用政策管理存取權
- 亞馬遜託管區塊鏈（AMB）查詢如何與 IAM
- Amazon Managed Blockchain () AMB 查詢的身分型政策範例
- 疑難排解 Amazon Managed Blockchain (AMB) 查詢身分和存取

物件

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，具體取決於您在 AMB Query 中執行的工作。

服務使用者 — 如果您使用 AMB Query 服務執行工作，則管理員會為您提供所需的認證和權限。當您使用更多AMB查詢功能來完成工作時，您可能需要其他權限。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法在「AMB查詢」中存取特徵，請參閱疑難排解 Amazon Managed Blockchain (AMB) 查詢身分和存取。

服務管理員 — 如果您負責公司的 AMB Query 資源，您可能擁有「AMB查詢」的完整存取權。判斷服務使用者應存取哪些 AMB Query 功能和資源是您的工作。然後，您必須向IAM管理員提交請求，才能變更服務使用者的權限。檢閱此頁面上的資訊，以瞭解的基本概念IAM。若要深入瞭解貴公司如何IAM搭配 AMB Query 使用，請參閱亞馬遜託管區塊鏈（AMB）查詢如何與 IAM。

IAM系統管理員 — 如果您是IAM系統管理員，您可能想要瞭解如何撰寫原則以管理 AMB Query 存取權的詳細資訊。若要檢視可在中使用的以AMB查詢身分識別為基礎的策略範例IAM，請參閱。Amazon Managed Blockchain () AMB 查詢的身分型政策範例

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以IAM使用者身分或假設IAM角色來驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM身分識別中心) 使用者、貴公司的單一登入驗證，以及您的 Google 或 Facebook 認證都是聯合身分識別的範例。當您以同盟身分登入時，您的管理員先前會使用IAM角色設定聯合身分識別。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以密碼編譯方式簽署您的要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署要求的詳細資訊，請參閱使用IAM者指南中的簽署 AWS API要求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。若要深入瞭解，請參閱使用AWS IAM Identity Center 者指南中的多重要素驗證和使用多重要素驗證 (MFA) AWS的使用IAM者指南。

AWS 帳戶 根使用者

建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務 和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需需要您以 root 使用者身分登入的完整工作清單，請參閱《使用指南》中的〈需要 root 使用者認證的IAM工作〉。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時認證 AWS 服務 來存取。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步處理至您自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需IAM身分識別中心的相關資訊，請參閱IAM識別中心是什麼？ 在《AWS IAM Identity Center 使用者指南》中。

IAM 使用者和群組

IAM使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定權限。在可能的情況下，我們建議您仰賴臨時登入資料，而不要建立具有長期認證 (例如密碼和存取金鑰) 的IAM使用者。不過，如果您的特定使用案例需要使用IAM者的長期認證，建議您輪換存取金鑰。如需詳細資訊，請參閱《使用指南》中的「IAM定期輪換存取金鑰」以瞭解需要長期認證的使用案例。

IAM群組是指定IAM使用者集合的身分識別。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為的群組，IAMAdmins並授與該群組管理IAM資源的權限。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。要了解更多信息，請參閱《IAM用戶指南》中的創建用戶（而不是角色）的IAM時間。

IAM 角色

IAM角色是您 AWS 帳戶 中具有特定權限的身份。它類似於用IAM戶，但不與特定人員相關聯。您可以 AWS Management Console 透過切換角色來暫時擔任中的角色。IAM您可以呼叫 AWS CLI 或 AWS API作業或使用自訂來擔任角色URL。如需有關使用角色方法的詳細資訊，請參閱《使用指南》中的IAM〈使用IAM角色〉。

IAM具有臨時認證的角色在下列情況下很有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需聯合角色的相關資訊，請參閱《使用指南》中的〈建立第三方身分識別提供IAM者的角色〉。如果您使用IAM身分識別中心，則需要設定權限集。為了控制身分驗證後可以存取的內IAM容，IAMIdentity Center 會將權限集與中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時IAM使用者權限 — IAM 使用者或角色可以假定某個IAM角色，暫時取得特定工作的不同權限。
- 跨帳戶存取 — 您可以使用IAM角色允許不同帳戶中的某個人 (受信任的主體) 存取您帳戶中的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要瞭解跨帳戶存取角色與以資源為基礎的政策之間的差異，請參閱《IAM使用者指南》IAM中的〈跨帳號資源存取〉。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中撥打電話時，該服務通常會在 Amazon 中執行應用程式EC2或將物件存放在 Amazon S3 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。

- 轉寄存取工作階段 (FAS) — 當您使用使用IAM者或角色執行中的動作時 AWS，您會被視為主參與者。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS會使用主參與者呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。FAS只有當服務收到需要與其他 AWS 服務 資源互動才能完成的請求時，才會發出請求。在此情況下，您必須具有執行這兩個動作的許可。有關提出FAS請求時的策略詳細信息，請參閱轉發訪問會話。
- 服務角色 — 服務角IAM色是服務代表您執行動作的角色。IAM管理員可以從中建立、修改和刪除服務角色IAM。如需詳細資訊，請參閱《IAM使用指南》 AWS 服務中的建立角色以將權限委派給
- 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM管理員可以檢視 (但無法編輯服務連結角色) 的權限。

- 在 Amazon 上執行的應用程式 EC2 — 您可以使用IAM角色來管理在執行個體上EC2執行的應用程式以及發出 AWS CLI 或 AWS API請求的臨時登入資料。這比在EC2實例中存儲訪問密鑰更好。若要將 AWS 角色指派給EC2執行個體並讓其所有應用程式都能使用，請建立附加至執行個體的執行個體設定檔。執行個體設定檔包含角色，可讓執行個體上EC2執行的程式取得臨時登入資料。如需詳細資訊，請參閱使用者指南中的使用IAM角色將許可授與在 Amazon EC2 執行個體上執行的應IAM用程式。

要了解是否使用IAM角色還是用IAM戶，請參閱《用戶指南》中的「IAM創建IAM角色的時機（而不是用戶）」。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以JSON文件的形式儲存在中。如需有關JSON原則文件結構和內容的詳細資訊，請參閱《IAM使用指南》中的策略概觀。JSON

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授與使用者對所需資源執行動作的權限，IAM管理員可以建立IAM策略。然後，系統管理員可以將IAM原則新增至角色，使用者可以擔任這些角色。

IAM原則會定義動作的權限，不論您用來執行作業的方法為何。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或取得角色資訊 AWS API。

身分型政策

以身分識別為基礎的原則是您可以附加至身分識別 (例如使用者、使用IAM者群組或角色) 的JSON權限原則文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要瞭解如何建立以身分識別為基礎的策略，請參閱《IAM使用指南》中的〈建立IAM策略〉。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。若要了解如何在受管策略或內嵌策略之間進行選擇，請參閱《IAM使用手冊》中的「在受管策略和內嵌策略之間進行選擇」。

資源型政策

以資源為基礎的JSON策略是您附加至資源的政策文件。以資源為基礎的政策範例包括IAM角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的策略IAM中使用 AWS 受管政策。

存取控制清單 (ACLs)

存取控制清單 (ACLs) 控制哪些主參與者 (帳戶成員、使用者或角色) 具有存取資源的權限。ACLs類似於以資源為基礎的策略，雖然它們不使用JSON政策文件格式。

Amazon S3 和 Amazon VPC 是支持服務的示例ACLs。 AWS WAF若要進一步了解ACLs，請參閱 Amazon 簡單儲存服務開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 權限界限 — 權限界限是一項進階功能，您可以在其中設定以身分識別為基礎的原則可授與給IAM實體 (IAM使用者或角色) 的最大權限。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需有關權限界限的詳細資訊，請參閱《IAM使用指南》中的IAM實體的權限界限。
- 服務控制策略 (SCPs) — SCPs 是指定中組織或組織單位 (OU) 最大權限的JSON策略 AWS Organizations。 AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多

個服務。如果您啟用組織中的所有功能，則可以將服務控制策略 (SCPs) 套用至您的任何或所有帳戶。SCP限制成員帳戶中實體的權限，包括每個帳戶 AWS 帳戶根使用者。若要取得有關 Organizations 的更多資訊SCPs，請參閱《AWS Organizations 使用指南》中的〈SCPs運作方式〉

- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱《IAM使用指南》中的工作階段原則。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。若要瞭解如何在涉及多個原則類型時 AWS 決定是否允許要求，請參閱IAM使用指南中的原則評估邏輯。

亞馬遜託管區塊鏈（AMB）查詢如何與 IAM

在您使用IAM來管理AMB查詢的存取權之前，請先瞭解哪些IAM功能可搭配 AMB Query 使用。

IAM您可以搭配 Amazon Managed Blockchain 使用的功能 (AMB) 查詢

IAM 功能	AMB查詢支援
身分型政策	是
資源型政策	否
政策動作	是
政策資源	否
政策條件索引鍵	否
ACLs	否
ABAC(策略中的標籤)	否
臨時憑證	是
主體許可	是

IAM 功能	AMB查詢支援
服務角色	否
服務連結角色	否

若要取得AMB查詢和其他 AWS 服務如何與大部分IAM功能搭配運作的高階檢視，請參閱IAM使用者指南IAM中的使用AWS 服務。

以身分識別為基礎的查詢原則 AMB

支援以身分識別為基礎的原則：是

以身分識別為基礎的原則是您可以附加至身分識別 (例如使用者、使用IAM者群組或角色) 的JSON權限原則文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要瞭解如何建立以身分識別為基礎的策略，請參閱《IAM使用指南》中的〈建立IAM策略〉。

使用以IAM身分識別為基礎的策略，您可以指定允許或拒絕的動作和資源，以及允許或拒絕動作的條件。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。若要瞭解可在JSON策略中使用的所有元素，請參閱《使用IAM者指南》中的IAMJSON策略元素參考資料。

查詢的身分識別原則範例 AMB

若要檢視以身分識別為基礎的AMB策略範例，請參閱。Amazon Managed Blockchain () AMB 查詢的身分型政策範例

查詢內AMB的資源型政策

支援以資源為基礎的政策：否

以資源為基礎的JSON策略是您附加至資源的政策文件。以資源為基礎的政策範例包括IAM角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

若要啟用跨帳戶存取，您可以在以資源為基礎的策略中指定一個或多個帳戶中的一個或多個帳戶中的IAM實體作為主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主參與者和資源位於

不同時 AWS 帳戶，受信任帳戶中的IAM管理員也必須授與主參與者實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱《IAM使用指南》IAM中的〈跨帳號資源存取〉。

AMB查詢的策略動作

支援原則動作：是

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

JSON策略`Action`元素描述了您可以用來允許或拒絕策略中存取的動作。策略動作通常與關聯的 AWS API操作具有相同的名稱。有一些例外情況，例如沒有匹配API操作的僅限權限的操作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看AMB查詢動作清單，請參閱服務授權參考中的 Amazon 受管區塊鏈定義的動作 (AMB) 查詢。

AMB查詢中的策略動作會在動作之前使用下列前置詞：

```
managedblockchain-query:
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

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

若要檢視以身分識別為基礎的AMB策略範例，請參閱。Amazon Managed Blockchain () AMB 查詢的身分型政策範例

AMB查詢的策略資源

支援政策資源：否

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

ResourceJSON原則元素會指定要套用動作的一個或多個物件。陳述式必須包含 Resource 或 NotResource 元素。最佳做法是使用其 Amazon 資源名稱 (ARN) 指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

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

若要查看AMB查詢資源類型及其清單ARNs，請參閱服務授權參考中的 Amazon 受管區塊鏈定義的資源 (AMB) 查詢。若要了解可以針對每個資源指定哪些動作，請參閱 Amazon 受管區塊鏈定義ARN的動作 (AMB) 查詢。

若要檢視以身分識別為基礎的AMB策略範例，請參閱。Amazon Managed Blockchain () AMB 查詢的身分型政策範例

查詢的政策條件AMB索引鍵

支援服務特定政策條件金鑰：否

管理員可以使用 AWS JSON策略來指定誰可以存取什麼內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，只有在IAM使用者名稱標記資源時，您才可以授與IAM使用者存取資源的權限。如需詳細資訊，請參閱《IAM使用指南》中的IAM政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《使用指南》中的AWS 全域條件內IAM容索引鍵。

若要查看AMB查詢條件金鑰清單，請參閱服務授權參考中的 Amazon Managed Blockchain 的條件金鑰 (AMB) 查詢。若要了解可以使用條件金鑰的動作和資源，請參閱 Amazon 受管區塊鏈定義的動作 (AMB) 查詢。

若要檢視以身分識別為基礎的AMB策略範例，請參閱。Amazon Managed Blockchain () AMB 查詢的身分型政策範例

ACLs在AMB查詢中

支持ACLs：無

存取控制清單 (ACLs) 控制哪些主參與者 (帳戶成員、使用者或角色) 具有存取資源的權限。ACLs類似於以資源為基礎的策略，雖然它們不使用JSON政策文件格式。

ABAC使用AMB查詢

支援 ABAC (策略中的標籤): 否

以屬性為基礎的存取控制 (ABAC) 是一種授權策略，可根據屬性定義權限。在中 AWS，這些屬性稱為標籤。您可以將標籤附加至IAM實體 (使用者或角色) 和許多 AWS 資源。標記實體和資源是的第一步ABAC。然後，您可以設計ABAC策略，以便在主參與者的標籤與他們嘗試存取的資源上的標籤相符時允許作業。

ABAC在快速成長的環境中很有幫助，並且有助於原則管理變得繁瑣的情況。

如需根據標籤控制存取，請使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需有關的詳細資訊ABAC，請參閱什麼是ABAC？ 在《IAM使用者指南》中。若要檢視包含設定步驟的自學課程ABAC，請參閱《使用指南》中的〈使用以屬性為基礎的存取控制 (ABAC) IAM 〉。

搭配AMB查詢使用臨時認證

支持臨時憑據：是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配使用 AWS 服務 ，請參閱《IAM使用指南》IAM中的使用方式。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立臨時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需有關切換角色的詳細資訊，請參閱《IAM使用者指南》中的〈切換到角色 (主控台)〉。

您可以使用 AWS CLI 或手動建立臨時認證 AWS API。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱IAM。

查AMB詢的跨服務主體權限

支援轉寄存取工作階段 (FAS)：是

當您使用使用IAM者或角色在中執行動作時 AWS，您會被視為主參與者。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS會使用主參與者呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。FAS只有當服務收到需要與其他 AWS 服務 資源互動才能完成的請求時，才會發出請求。在此情況下，您必須具有執行這兩個動作的許可。有關提出FAS請求時的策略詳細信息，請參閱轉發訪問會話。

AMB查詢的服務角色

支援服務角色：否

服務角色是服務假定代表您執行動作的IAM角色。IAM管理員可以從中建立、修改和刪除服務角色 IAM。如需詳細資訊，請參閱《IAM使用指南》 AWS 服務中的建立角色以將權限委派給

> **Warning**
>
> 變更服務角色的權限可能會中斷AMB查詢功能。只有在 AMB Query 提供指引時才編輯服務角色。

查AMB詢的服務連結角色

支援服務連結角色：否

服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM管理員可以檢視 (但無法編輯服務連結角色) 的權限。

如需有關建立或管理服務連結角色的詳細資訊，請參閱使用IAM的AWS 服務。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

Amazon Managed Blockchain () AMB 查詢的身分型政策範例

根據預設，使用者和角色沒有建立或修改 AMB Query 資源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或執行工作 AWS API。若要授與使用者對所需資源執行動作的權限，IAM管理員可以建立IAM策略。然後，系統管理員可以將IAM原則新增至角色，使用者可以擔任這些角色。

若要瞭解如何使用這些範例原則文件來建立以IAM身分識別為基礎的JSON策略，請參閱使用指南中的IAM建立IAM策略。

有關AMB查詢定義的動作和資源類型的詳細資訊，包括每種資源類型的格式，請參閱服務授權參考中的 Amazon Managed Blockchain 的動作、資源和條件金鑰 (AMB) 查詢。ARNs

主題

- 政策最佳實務
- 允許使用者檢視他們自己的許可
- 存取特定的 Amazon Managed Blockchain (AMB) 查詢API動作

政策最佳實務

以身分識別為基礎的政策會決定某人是否可以建立、存取或刪除您帳戶中的 AMB Query 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需詳細資訊，請參閱AWS 《IAM使用指南》中針對工作職能的AWS 受管理策略或受管理的策略。
- 套用最低權限權限 — 當您使用原則設定權限時，IAM只授與執行工作所需的權限。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需有關使用套用權限IAM的詳細資訊，請參閱《使用指南》IAM中的IAM《策略與權限》。
- 使用IAM策略中的條件進一步限制存取 — 您可以在策略中新增條件，以限制對動作和資源的存取。例如，您可以撰寫政策條件，以指定必須使用傳送所有要求SSL。您也可以使用條件來授與對服務動作的存取權 (如透過特定 AWS 服務) 使用 AWS CloudFormation。如需詳細資訊，請參閱《IAM使用指南》中的IAMJSON策略元素：條件。
- 使用 IAM Access Analyzer 驗證您的原IAM則，以確保安全和功能性的權限 — IAM Access Analyzer會驗證新的和現有的原則，以便原則遵循IAM原則語言 (JSON) 和IAM最佳做法。IAMAccess

Analyzer 提供超過 100 項原則檢查和可行的建議，協助您撰寫安全且功能正常的原則。如需詳細資訊，請參閱IAM使IAM用指南中的存取分析器原則驗證。

- 需要多因素驗證 (MFA) — 如果您的案例需要使IAM用者或 root 使用者 AWS 帳戶，請開啟以取得額外MFA的安全性。若要在呼叫API作業MFA時需要，請在原則中新增MFA條件。如需詳細資訊，請參閱《IAM使用指南》中的 < 設定MFA受保護的API存取 > 。

如需有關中最佳作法的詳細資訊IAM，請參閱《IAM使用指南》IAM中的「安全性最佳作法」。

允許使用者檢視他們自己的許可

此範例顯示如何建立原則，讓使IAM用者檢視附加至其使用者身分識別的內嵌和受管理原則。此原則包含在主控台上或以程式設計方式使用或完成此動作的 AWS CLI 權限 AWS API。

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

```
        }
    ]
}
```

存取特定的 Amazon Managed Blockchain (AMB) 查詢API動作

> **Note**
>
> 若要存取「AMB查詢」以進行API呼叫，您將需要具有「查AMB詢」適當IAM權限的使用者認證 (AWS_ACCESS_KEY_ID和AWS_SECRET_ACCESS_KEY)。

Example IAM訪問所有 Amazon Managed Blockchain（AMB）查詢的政策 APIs

此範例會授與您 AWS 帳戶 存取所有 AMB Query 的IAM使用者APIs。

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

Example IAM訪問 Amazon Managed Blockchain（AMB）查詢**`ListTransactions`**和**`GetTransaction`** APIs

此示例授予IAM用戶 AWS 帳戶 訪問查AMB詢`ListTransaction`和 `GetTransaction` APIs

> **Note**
>
> 您可以使用 other 取代或新增範例APIs中的，以APIs便存取其他或更多項目APIs。如需AMB查詢清單APIs，請參閱 Amazon 受管區塊鏈 (AMB) 查詢API參考指南。

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

疑難排解 Amazon Managed Blockchain (AMB) 查詢身分和存取

使用下列資訊可協助您診斷及修正使用 AMB Query 和時可能會遇到的常見問題IAM。

主題

- 我沒有在AMB查詢中執行操作的權限

我沒有在AMB查詢中執行操作的權限

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

當使用mateojacksonIAM者嘗試使用主控台來檢視虛構*my-example-widget*資源的詳細資料，但沒有虛構的managedblockchain-query::*GetWidget*權限時，就會發生下列範例錯誤。

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

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 managedblockchain-query::*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

Amazon Managed Blockchain (AMB) 在 Amazon 上查詢 API 使用指標 CloudWatch

Amazon 上的 API 使用量指標 CloudWatch

針 CloudWatch 對 Amazon Managed Blockchain (AMB) 查詢服務配額而發佈的 API 使用量指標。您可以設定警示，以便在使用量接近服務配額時提醒您。如需與服務配額 CloudWatch整合的詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的 AWS 用量指標。

AMB 查詢會在`AWS/Usage`命名空間中以`Amazon Managed Blockchain Query`服務名稱發佈下列 API 指標。

指標	描述
`CallCount`	在 AMB 查詢中對 API 進行的呼叫總數。SUM 代表指定期間內對 API 的呼叫總數。

Amazon Managed Blockchain (AMB) 查詢會將使用量指標發佈至具有下列維度的`AWS/Usage`命名空間。

維度	描述
Service (服務)	包含資源的 AWS 服務名稱。 `Amazon Managed Blockchain Query` 將永遠是此維度的值。
Type	要報告的實體類型。 `API`將永遠是此維度的值。
資源	要報告的資源類型。使用的 AMB 查詢 API 作業的名稱將成為此維度的值。
類別	要報告的資源類別。 `None`將永遠是此維度的值。

AMB 查詢使用者指南的文件記錄

下表說明 AMB 查詢的文件版本。

變更	描述	日期
AMB 查詢支持比特幣交易標識符和交易哈希	對於比特幣網絡，AMB 查詢 API 操作同時支持交易標識符（`transactionId`）和交易哈希（`transactionHash`）。	2024年3月21日
Support Amazon 上的 API 使用指標 CloudWatch	AMB 查詢在 CloudWatch上添加了對 API 使用指標的支持。這些使用量度對應至 AMB 查詢服務配額。	2024年2月8日
Support 尚未達到終點的交易	AMB 查詢增加了對尚未達到最終事務的支持。它也會從作業的回應中移除對`status`屬性的支`GetTransaction`援。相反地，您將使用`confirmationStatus`和`executionStatus`屬性來判斷交易的狀態。	2024年2月1日
棄用交易資料類型中的`status`屬性	Amazon Managed Blockchain (AMB) 查詢已淘汰交易資料類型中的`status`屬性。您必須使用`confirmationStatus`和`executionStatus`欄位來判斷交易是否為`FINAL`或`FAILED`。`status`	2023 年 12 月 20 日
Support 塞波利亞測試網	Amazon Managed Blockchain（AMB）查詢現在支持以太坊 Sepolia 測試網上的查詢。	2023 年 10 月 19 日

Support 資產合同	您可以使用 ListAssetContracts API 作業來列出依指定位址部署的項目。此外，如果您有合約地址，則可以使用 GetAssetContract API 操作來擷取合約的詳細資訊。	2023 年 10 月 16 日
Support 比特幣測試網	Amazon Managed Blockchain（AMB）查詢現在支持比特幣測試網上的查詢。	2023 年 10 月 16 日
初始版本	AMB 查詢服務的初始版本。	2023 年 7 月 27 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。