aws

開発者ガイド

Amazon Location Service

Amazon Location Service: 開発者ガイド

Table of Contents

Amazon Location Service へようこそ

Amazon Location Service 開発者ガイドへようこそ。

次のトピックは、実行したい内容に基づいて、ドキュメントの活用方法を理解する上で役立ちます。

Amazon Location の概要

- Amazon Location の概要について学習できます。
- Amazon Location Service の使い方の章で機能をさらに深く掘り下げてください。
- Amazon Location デモサイトでは、デモアプリを試すことができます。
- AWS アカウント をすでにお持ちの場合は、Amazon Location Service コンソールを使用して機能を実際に確かめることができます。

Amazon Location を開発者として使用する

- クイックスタート を使用してアプリケーションを構築します。
- Amazon Location Service の使い方の章では、Amazon Location Service さまざまな機能の仕組みについて説明します。
- Amazon Location を使用した開発の章で使用できる SDK とツールを学習できます。
- 自身のアプリで使用できるコード例やチュートリアルを参照できます。Amazon Location デモサイトのサンプルページにアクセスして、機能、言語、またはプラットフォームでフィルタリングできるコード例を検索することもできます。
- API リファレンスガイドから、Amazon Location API に関する情報を参照できます。

Amazon Location Service とは？

Amazon Location Service では、地図、目標地点、ジオコーディング、ルーティング、ジオフェンス、追跡などの機能を含む位置データと機能をアプリケーションに追加できます。Amazon Location は、信頼できるグローバルプロバイダーである Esri、Grab、HERE からの高品質なデータを使用して、ロケーションベースのサービス（LBS）を提供しています。手頃な価格のデータ、追跡機能、ジオフェンシング機能、ヘルスモニタリング用の組み込みメトリクスにより、位置情報に対応した高度なアプリケーションを構築できます。

Amazon Location を利用すれば、組織のデータを常に管理できます。Amazon Location は、顧客メタデータとアカウント情報を削除することで、データプロバイダーに送信されるすべてのクエリを匿名

化します。さらに、施設、アセット、人員の所在地など、トラッキングやジオフェンシングに関する機密性の高い位置情報が、AWS アカウントから一切出ることはありません。これにより、機密情報を第三者から保護し、ユーザーのプライバシーを保護し、アプリケーションのセキュリティリスクを軽減できます。Amazon Location では、Amazon や第三者はお客様のデータを販売したり、広告に使用したりする権利を持ちません。

Amazon Location は、、Amazon AWS CloudTrail、Amazon CloudWatch、および AWS Identity and Access Management (IAM) などの サービスと完全に統合 EventBridgeされています。Amazon Location は、データ統合により開発ワークフローを簡素化し、ビルトインのモニタリング、セキュリティ、コンプライアンス機能によりアプリケーションを本番環境に迅速に移行させます。

詳細とハイライトについては、Amazon Location Service のサービスページを参照してください。

Amazon Location の主な機能

Amazon Location には次の機能があります:

マップ

Amazon Location Service Map は、位置情報を視覚化することができ、多くの位置情報サービス機能の基礎となっています。Amazon Location Service は、オープンデータマップのほか、グローバルな位置情報データプロバイダーである Esri、Grab、HERE から取得されたさまざまなスタイルのマップタイルを提供します。

場所

Amazon Location Service Places を使用すると、検索機能をアプリケーションに統合したり、住所を緯度と経度の地理座標に変換 (ジオコーディング) したり、座標を住所に変換 (リバースジオコーディング) したりできます。Amazon Location Service では、Places 機能をサポートするために Esri、Grab、HERE から高品質の地理空間データを収集しています。

ルーティング

Amazon Location Service Routes では、 up-to-date 道路やライブトラフィック情報に基づいてルートを検索し、移動時間を推定できます。任意の 2 つの場所間の移動時間、距離、ルートをアプリケーションに提供する機能を構築できます。ルートマトリックスの時間と距離を計算して、ルートプランニングに役立てることができます。

ジオフェンシング

Amazon Location Service Geofence を使用すると、ジオフェンスと呼ばれる定義された地理的境界にデバイスが出入りした際に、それを検出して動作する機能を、アプリケーションに提供できます。ジオフェンス違反が検出され EventBridge ると、エントリまたは終了イベントを Amazon に自動的に送信します。これにより、ターゲットに通知を送信するなどのダウンストリームアクションを開始できます。

トラッカー

Amazon Location Service ストラッカーを使用すると、追跡対応アプリケーションを実行しているデバイスの現在位置と過去の位置を取得できます。また、トラッカーを Amazon Location Service ジオフェンスとリンクさせて、デバイスからの位置情報のアップデートをジオフェンスと照合して自動的に評価することもできます。トラッカーは、位置の更新をジオフェンスに保存または評価する前にフィルタリングすることで、コスト削減に役立ちます。

トラッカーを使用すると、追跡対象デバイスの機密位置情報がAWS アカウントから出ることはありません。これにより、機密情報を第三者から保護し、ユーザーのプライバシーを保護し、セキュリティリスクを軽減できます。

Amazon Location で使用できるサービス

Amazon Location Service とあわせて以下のサービスを利用できます。

統合されたモニタリングと管理

Amazon Location Service は Amazon CloudWatch、AWS CloudTrail、および Amazon と統合されており、効率的なモニタリングとデータ管理 EventBridge が可能です。

- Amazon CloudWatch – リクエスト、レイテンシー、障害、ログなど、サービスの使用状況と状態に関するメトリクスを表示します。詳細については、「the section called “によるモニタリング CloudWatch”」を参照してください。
- AWS CloudTrail — ユーザー、ロール、または AWS サービスが実行したアクションを含む API 呼び出しをログしてモニタリングします。詳細については、「the section called “Amazon Location CloudTrail での の使用”」を参照してください。
- Amazon EventBridge – イベント駆動型アプリケーションアーキテクチャを有効にして、 AWS Lambda関数を使用してアプリケーションとワークフローの他の部分をアクティブ化できるよ

うにします。詳細については、「the section called “でのイベントへの対応 EventBridge”」を参照してください。

デベロッパーツール

Amazon Location Service には、開発者が位置情報対応アプリケーションを構築するためのさまざまなツールが用意されています。これには、標準 AWS SDKsモバイルおよびウェブ SDKs、および などのオープンソースライブラリと組み合わせるサンプルコードが含まれます MapLibre。Amazon Location Service コンソールから、視覚的でインタラクティブな学習ツールを使用してリソースについて学習しましょう。

Amazon Location Service のクイックスタート

Amazon Location Service の使用を開始する最も効率的な方法は、Amazon Location コンソールです。リソースを作成および管理し、Explore ページを使用して Amazon Location 機能を試すことができます。

> **Note**
>
> Amazon Location Service コンソールを使用するか、このチュートリアルの残りの部分に従うには、 ではAmazon Location Service を使用するための前提条件、 AWS アカウントの作成や Amazon Location へのアクセスの許可など、まず を完了する必要があります。

Amazon Location API について学習を始めるには、次のチュートリアルを使用して、インタラクティブマップを表示し、検索機能を使用する簡単なアプリケーションを作成してください。このチュートリアルには 3 つのバージョンがあります。1 つは を使用してシンプルなウェブページを作成する方法を示し JavaScript、2 つ目は Kotlin を使用する Android アプリケーションで同じ、3 つ目は Swift を使用する iOS アプリケーションで同じです。

トピック

- ウェブ ACL の作成
- Android アプリを作成する
- iOS アプリケーションの作成

ウェブ ACL の作成

このセクションでは、地図と特定の場所での検索機能を備えた静的ページを作成します。まず Amazon Location リソースを作成し、アプリケーションの API キーを作成します。

トピック

- アプリ用の Amazon Location リソースを作成します。
- アプリケーションの認証の設定
- アプリケーション用の HTML の作成
- アプリケーションにインタラクティブマップを追加

アプリ用の Amazon Location リソースを作成します。

まだ持っていない場合は、アプリケーションが利用される Amazon Location リソースaを作成する必要があります。ここでは、アプリケーションで地図を表示するためのマップリソースと、地図上の場所を検索するための場所インデックスを作成します。

位置情報リソースをアプリケーションに追加

1. 使用するマップスタイルを選択します。

   a. Amazon Location コンソールの マップページで、マップを作成を選択してマップスタイルをプレビューします。

   b. 新しいマップリソースの 名前と 説明を追加します。マップリソースに使用する名前をメモしておきます。チュートリアルの後半でスクリプトファイルを作成するときに必要になります。

   c. マップ をクリックします。

   > **Note**
   >
   > マップスタイルを選択すると、利用するマップデータプロバイダーも選択されます。配送車両や従業員など、ビジネスで使用する資産をアプリケーションで追跡またはルーティングする場合、使用できる位置情報プロバイダーはHEREのみとなります。詳細については、「AWS サービス規約」のセクション 82 を参照してください。

   d. Amazon Location の利用規約に同意し、Create map (マップを作成) を選択します。選択したマップを、拡大、縮小、または任意の方向への画面移動をすることができます。

   e. 新しいマップリソースに表示される Amazon リソースネーム (ARN) をメモします。チュートリアルの後半で、これを使って正しい認証を作成します。

2. 使用する場所インデックスを選択します。

   a. Amazon Location コンソールの [プレイスインデックス」ページで、「プレイスインデックスを作成」 を選択します。

b. 新しいプレイスインデックスリソースの 名前と 説明を追加します。場所インデックスリソースに入力した名前をメモしておきます。チュートリアルの後半でスクリプトファイルを作成するときに必要になります。

c. データプロバイダーを選択します。

> **Note**
>
> ほぼすべてのケースでは、すでに選択したマッププロバイダーと一致するデータプロバイダーを選択します。これにより、検索が地図と一致するようになります。配送車両や従業員など、ビジネスで使用する資産をアプリケーションで追跡またはルーティングする場合、使用できる位置情報プロバイダーはHEREのみとなります。詳細については、「AWS サービス規約」のセクション 82 を参照してください。

d. データストレージオプションを選択します。このチュートリアルでは結果は保存されないため、いいえ、1 回のみを選択することができます。

e. Amazon Location 利用規約に同意し、プレイスインデックスを作成を選択します。

f. 新しいプレースインデックスリソースに表示される ARN をメモします。このチュートリアルの次のセクションで、これを使って正しい認証を作成します。

アプリケーションの認証の設定

このチュートリアルで作成するアプリケーションは、匿名利用が可能です。つまり、ユーザはアプリケーションを利用するためにAWSにサインインする必要はありません。ただし、デフォルトでは、Amazon Location Service API を利用するには認証が必要です。Amazon Cognito や API キーを使用して、匿名ユーザーの認証と承認を行うことができます。このチュートリアルでは、サンプルアプリケーションで使用する API キーを作成します。

> **Note**
>
> Amazon Location Service で API キーまたは Amazon Cognito を使用する方法の詳細については、Amazon Location Service へのアクセスを許可するを参照してください。

アプリケーションの認証を設定

1. Amazon Location コンソールをクリックして、左側のメニューから API キーを選択します。
2. API の作成を選択します。

> **Important**
>
> 作成した API キーは、前のセクションで作成した Amazon Location Service AWS アカウント AWS リソースと同一のリージョンにある必要があります。

3. API キーの作成ページで、次の情報を入力します。

   - 名前— API キーの名前、例えば `MyWebAppKey`。
   - リソース — 前のセクションで作成した Amazon Location Map] リソースと Place index] リソースを選択します。リソースの追加を選択すると、複数のリソースを追加できます。これにより、API キーをそれらのリソースで使用できるようになります。
   - アクション — この API キーを利用して、承認したアクションを指定します。チュートリアルが期待どおりに機能するように、少なくとも geo:GetMap* と geo:SearchPlaceIndexfForPosition を選択する必要があります。
   - オプションで API キーに説明、有効期限、またはタグを追加できます。また、特定のドメインからのみ使用されるキーに制限するために、リファラ (例えば`*.example.com`) を追加することもできます。つまり、このチュートリアルはそのドメインでのみ動作することになります。

> **Note**
>
> APIキーの使用を保護するには、有効期限かリファラを設定することを推奨します。

4. API キーを作成を選択して API キーを作成します。
5. APIキーを表示するを選択し、チュートリアルの後半で使用できるようにキーの値をコピーします。これは `v1.public.`*`a1b2c3d4...`* 形式です。

> **Important**
>
> このキーは、このチュートリアルの後半でアプリケーションのコードを書くときに必要になります。

アプリケーション用の HTML の作成

このチュートリアルでは、マップを埋め込む静的 HTML ページを作成します。これにより、ユーザーはマップ上の特定の場所にある情報を検索することができます。アプリは、ウェブページ用の HTML ファイルと CSS ファイル、およびマップを作成し、ユーザーのインタラクションとマップイベントに応答するコード用の JavaScript (.js) ファイルの 3 つのファイルで構成されます。

まず、アプリケーションに使用される HTML と CSS のフレームワークを作成しましょう。これは、マップコンテナを保持する エレメントと、クエリに対する JSON レスポンスを表示する <pre> エレメントを持つシンプルなページになります。

クイックスタートアプリケーション用の HTML の作成

1. quickstart.html という名前の新しいファイルを作成します。
2. テキストエディターまたは任意の環境でファイルを編集します。以下を ファイルに追加します。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Quick start tutorial</title>

    <!-- Styles -->
    <link href="main.css" rel="stylesheet" />
  </head>

  <body>
    <header>
      <h1>Quick start tutorial</h1>
    </header>
    <main>
      <div id="map"></div>
      <aside>
        <h2>JSON Response</h2>
        <pre id="response"></pre>
      </aside>
    </main>
    <footer>This is a simple Amazon Location Service app. Pan and zoom. Click to
 see details about entities close to a point.</footer>

  </body>
```

```
</html>
```

この HTML には、次のステップで作成する CSS ファイルへのポインター、アプリケーション用のプレースホルダー要素、および説明テキストが含まれています。

このチュートリアルで後ほど実行する 2 つのプレースホルダー要素があります。1 つ目はマップコントロールを保持する <div id="map> エレメントです。2 つ目は <pre id="response"> エレメントで、マップ上の検索結果を表示します。

3. ファイルを保存します。

次に、Web ページに CSS を追加します。これにより、アプリケーションのテキストとプレースホルダー要素のスタイルが設定されます。

クイックスタートアプリケーション用の CSS の作成する

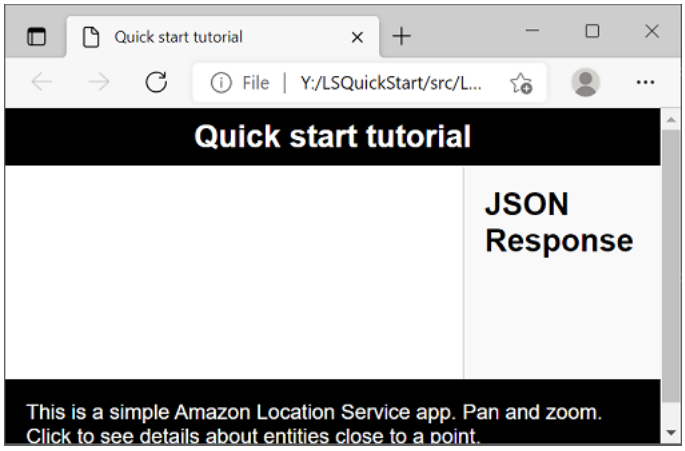
1. main.cssという新しいファイルを、前の手順で作成した quickstart.html ファイルと同じフォルダに作成します。
2. ご使用されたい工ディタでファイルを編集します。以下を ファイルに追加します。

```
* {
  box-sizing: border-box;
  font-family: Arial, Helvetica, sans-serif;
}

body {
  margin: 0;
}

header {
  background: #000000;
  padding: 0.5rem;
}

h1 {
  margin: 0;
  text-align: center;
  font-size: 1.5rem;
  color: #ffffff;
}
```

```
main {
  display: flex;
  min-height: calc(100vh - 94px);
}

#map {
  flex: 1;
}

aside {
  overflow-y: auto;
  flex: 0 0 30%;
  max-height: calc(100vh - 94px);
  box-shadow: 0 1px 1px 0 #001c244d, 1px 1px 1px 0 #001c2426, -1px 1px 1px 0
 #001c2426;
  background: #f9f9f9;
  padding: 1rem;
}

h2 {
  margin: 0;
}

pre {
  white-space: pre-wrap;
  font-family: monospace;
  color: #16191f;
}

footer {
  background: #000000;
  padding: 1rem;
  color: #ffffff;
}
```

これにより、他のユーザーが使用していないスペースを埋めるようにマップが設定され、回答用の領域がアプリの幅の 30% を占めるように設定され、タイトルと説明テキストの色とスタイルが設定されます。

3. ファイルを保存します。
4. これで、quickstart.htmlファイルをブラウザで表示して、アプリケーションのレイアウトを確認できるようになります。



次に、アプリケーションにマップコントロールを追加します。

アプリケーションにインタラクティブマップを追加

フレームワークと div プレースホルダーができたので、マップコントロールをアプリケーションに追加できます。このチュートリアルでは、マップコントロールとして MapLibre GL JS を使用し、Amazon Location Service からデータを取得します。また、APIキーでAmazon Location APIへのコールの署名を容易にするために JavaScript 認証ヘルパー を使用します。

インタラクティブマップをアプリケーションに追加

1. 前のセクションで作成したquickstart.htmlファイルを開きます。
2. 必要なライブラリへの参照と、これから作成するスクリプトファイルを追加します。必要な変更は**green**に示されています。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Quick start tutorial</title>

    <!-- Styles -->
    <link href="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.css"
 rel="stylesheet" />
    <link href="main.css" rel="stylesheet" />
  </head>

  <body>
    ...
```

```
    <footer>This is a simple Amazon Location Service app. Pan and zoom. Click to
 see details about entities close to a point.</footer>

    <!-- JavaScript dependencies -->
    <script src="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.js"></script>
    <script src="https://unpkg.com/@aws/amazon-location-client@1.x/dist/
amazonLocationClient.js"></script>
    <script src="https://unpkg.com/@aws/amazon-location-utilities-auth-helper@1.x/
dist/amazonLocationAuthHelper.js"></script>

    <!-- JavaScript for the app -->
    <script src="main.js"></script>
  </body>
</html>
```

アプリケーションに次の依存関係が追加されます。

- MapLibre GL JS 。このライブラリとスタイルシートには、マップタイルを表示し、パンやズームなどのインタラクティブ機能を含むマップコントロールが含まれています。このコントロールでは、地図上に独自の特徴を描くなどの機能拡張も可能です。
- Amazon Location クライアント。これにより、地図データを取得したり、地図上の場所を検索したりするために必要な Amazon Location 機能のインターフェースが提供されます。Amazon Location クライアントは AWS SDK for JavaScript v3 に基づいています。
- Amazon Location 認証ヘルパー。これにより、API キーまたは Amazon Cognito を利用して Amazon Location Service 認証するための便利な機能が提供されます。

このステップでは、次に作成するmain.jsへの参照先も追加されます。

3. quickstart.html ファイルを保存します。
4. HTML ファイルや CSS main.js ファイルと同じフォルダにという名前の新しいファイルを作成し、編集用に開きます。
5. 次のコードを ファイルに追加します。##のテキストは、以前に作成した API キー値、マップリソース名、場所リソース名、およびリージョンのリージョン識別子 (例えばus-east-1) に置き換えてください。

```
// Amazon Location Service resource names:
const mapName = "explore.map";
const placesName = "explore.place";
const region = "your_region";
```

```
const apiKey = "v1.public.a1b2c3d4...

// Initialize a map
async function initializeMap() {
  const mlglMap = new maplibregl.Map({
    container: "map", // HTML element ID of map element
    center: [-77.03674, 38.891602], // Initial map centerpoint
    zoom: 16, // Initial map zoom
    style: 'https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor?key=${apiKey}', // Defines the appearance of the map and authenticates
 using an API key
  });

  // Add navigation control to the top left of the map
  mlglMap.addControl(new maplibregl.NavigationControl(), "top-left");

  return mlglMap;
}

async function main() {
  // Initialize map and Amazon Location SDK client:
  const map = await initializeMap();
}

main();
```

このコードは Amazon Location リソースを設定し、 MapLibre GL JS マップコントロールを設定および初期化し、ID の 要素に配置しますmap。

initializeMap()関数を理解することは重要です。アプリケーションで MapLibre マップをレンダリングするために使用される新しいマップコントロール (mlglMapローカルでは呼び出されますが、残りのコードmapでは呼び出されます) が作成されます。

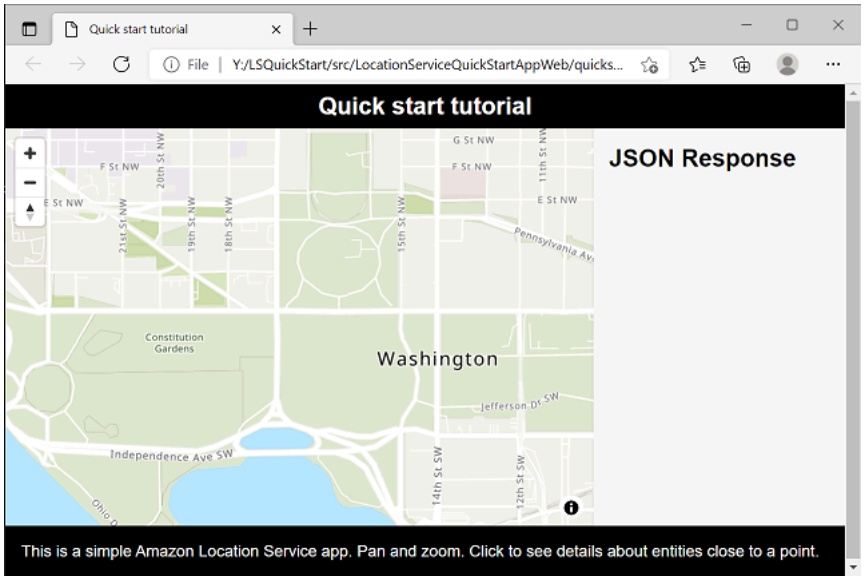
```
  // Initialize the map
  const mlglMap = new maplibregl.Map({
    container: "map", // HTML element ID of map element
    center: [-77.03674, 38.891602], // Initial map centerpoint
    zoom: 16, // Initial map zoom
    style: 'https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor?key=${apiKey}`, // Defines the appearance of the map and authenticates
 using an API key
  });
```

新しい MapLibre マップコントロールを作成すると、渡すパラメータはマップコントロールの初期状態を示します。以下のパラメータを設定します。

- HTML のマップ div 要素を使用する HTML コンテナ。
- ワシントン DC のある地点までのマップは初期中心。
- 16 へのズームレベル (近隣やブロックレベルに拡大)。
- マップに使用するスタイル MapLibre 。マップタイルを取得するために使用する URL と、マップをレンダリングするためのその他の情報を提供します。この URL には認証用の API キーが含まれていることに注意してください。

6. JavaScript ファイルを保存し、ブラウザで開きます。これで、ページ上にマップが作成され、画面移動やズーム操作を実行できるようになりました。

> **Note**
>
> このアプリを使用して、 MapLibre マップコントロールの動作を確認できます。Ctrl または Shift キーを押しながらドラッグ操作を試してみると、マップを操作する他の方法を確認できます。この機能はすべてカスタマイズ可能です。



アプリの設置はほぼ完成しました。次のセクションでは、マップ上の場所を選択し、選択した場所の住所を表示します。また、結果の JSON をページに表示して、結果全体を確認することもできます。

アプリケーションに検索を追加する

アプリケーションの最後のステップは、マップに検索を追加することです。この場合は、ある場所にあるアイテムを検索するリバースジオコーディング検索を追加します。

> **Note**
>
> Amazon Location Service では、名前または住所で検索して、地図上の場所の位置を検索することもできます。

アプリケーションに検索機能の追加

1. 前のセクションで作成したmain.jsファイルを開きます。
2. 図のようにmain関数を変更します。必要な変更は**green**に示されています。

```
async function main() {
  // Create an authentication helper instance using an API key
  const authHelper = await amazonLocationAuthHelper.withAPIKey(apiKey);


  // Initialize map and Amazon Location SDK client:
  const map = await initializeMap();

  const client = new amazonLocationClient.LocationClient({
    region,
    ...authHelper.getLocationClientConfig(), // Provides configuration required to
 make requests to Amazon Location
  });


  // On mouse click, display marker and get results:
  map.on("click", async function (e) {
    // Set up parameters for search call
    let params = {
      IndexName: placesName,
      Position: [e.lngLat.lng, e.lngLat.lat],
      Language: "en",
      MaxResults: "5",
    };

    // Set up command to search for results around clicked point
```

```
    const searchCommand = new
 amazonLocationClient.SearchPlaceIndexForPositionCommand(params);

    try {
      // Make request to search for results around clicked point
      const data = await client.send(searchCommand);

      // Write JSON response data to HTML
      document.querySelector("#response").textContent = JSON.stringify(data,
 undefined, 2);

      // Display place label in an alert box
      alert(data.Results[0].Place.Label);
    } catch (error) {
      // Write JSON response error to HTML
      document.querySelector("#response").textContent = JSON.stringify(error,
 undefined, 2);

      // Display error in an alert box
      alert("There was an error searching.");
    }
  });
}
```

このコードはまず、API キーを使用するように Amazon Location 認証ヘルパーを設定します。

```
const authHelper = await amazonLocationAuthHelper.withAPIKey(apiKey);
```

次に、その認証ヘルパーと、使用しているリージョンを使って新しい Amazon Location クライアントを作成します。

```
const client = new amazonLocationClient.LocationClient({
    region,
    ...authHelper.getLocationClientConfig(),
  });
```

次に、ユーザーがマップ・コントロール上のスポットを選択すると、コードはそれに応答する。これは、 に MapLibre 提供されたイベントをキャッチすることによって行われますclick。

```
map.on("click", async function(e) {
    ...
```

```
  });
```

MapLibre click イベントは、ユーザーが選択した緯度と経度 () を含むパラメータを提供しますe.lngLat。clickイベント内では、コードによってがsearchPlaceIndexForPositionCommand作成され、指定した緯度と経度にあるエンティティが検索されます。

```
    // Set up parameters for search call
    let params = {
      IndexName: placesName,
      Position: [e.lngLat.lng, e.lngLat.lat],
      Language: "en",
      MaxResults: "5"
    };

    // Set up command to search for results around clicked point
    const searchCommand = new
 amazonLocationClient.SearchPlaceIndexForPositionCommand(params);

    try {
      // Make request to search for results around clicked point
      const data = await client.send(searchCommand);
      ...
    });
```

ここで、IndexNameは先に作成したPlace Indexリソースの名前、PositionはLanguageを検索するための緯度と経度、Yは結果の優先言語、そしてMaxResultsは最大5つの結果のみを返すようにAmazon Locationに指示します。

残りのコードはエラーをチェックし、検索結果をresponseという<pre>要素に表示し、一番上の結果をアラートボックスに表示する。

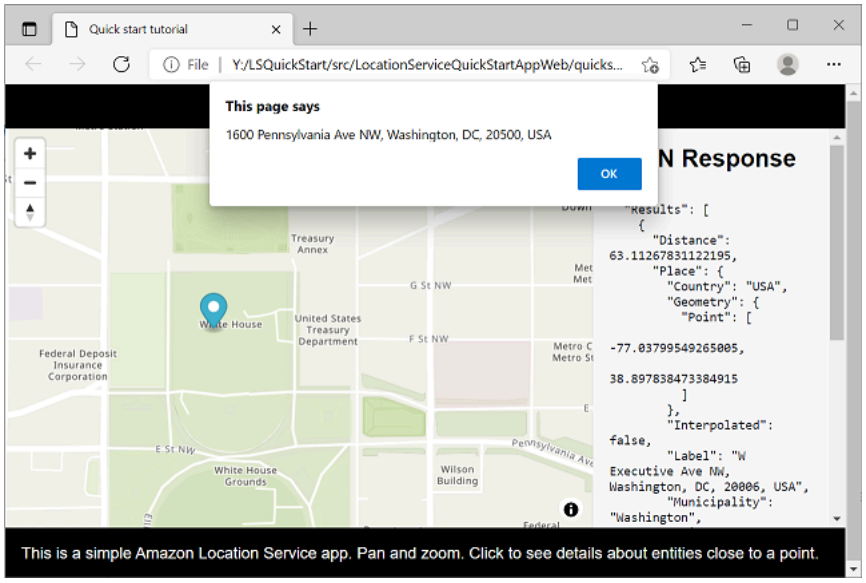
3. （オプショナル）ここでquickstart.htmlファイルを保存してブラウザで開くと、マップ上の場所を選択すると、選択した場所の名前または住所が表示されます。
4. アプリケーションの最後のステップは、 MapLibre 機能を使用して、ユーザーが選択した場所にマーカーを追加することです。main 関数を次のように呼び出します。必要な変更は**green**に示されています。

```
async function main() {
  // Create an authentication helper instance using an API key
  const authHelper = await amazonLocationAuthHelper.withAPIKey(apiKey);
```

```
 // Initialize map and Amazon Location SDK client
 const map = await initializeMap();
 const client = new amazonLocationClient.LocationClient({
   region,
   ...authHelper.getLocationClientConfig(), // Provides configuration required to
make requests to Amazon Location
 });

 // Variable to hold marker that will be rendered on click
 let marker;

 // On mouse click, display marker and get results:
 map.on("click", async function (e) {
   // Remove any existing marker
   if (marker) {
     marker.remove();
   }

   // Render a marker on clicked point
   marker = new maplibregl.Marker().setLngLat([e.lngLat.lng,
e.lngLat.lat]).addTo(map);

   // Set up parameters for search call
   let params = {
     IndexName: placesName,
     Position: [e.lngLat.lng, e.lngLat.lat],
     Language: "en",
     MaxResults: "5",
   };

   // Set up command to search for results around clicked point
   const searchCommand = new
amazonLocationClient.SearchPlaceIndexForPositionCommand(params);

...
```

このコードではmarker変数を宣言しており、ユーザーが場所を選択するたびに、選択した場所を示すZ変数が入力される。.addTo(map);でマップに追加されると、メーカーはマップコントロールによって自動的にレンダリングされる。また、このコードは以前のメーカーをチェックして削除し、画面には一度に 1 つのマーカーしか表示されないようにします。

5. `main.js`ファイルを保存し、ブラウザーで`quickstart.html`ファイルを開きます。以前と同様にマップ上で画面移動や拡大を行うことができますが、場所を選択すると、選択した場所の詳細が表示されるようになりました。



これでクイックスタートアプリケーションが完成しました。このチュートリアルでは、次のような静的 HTML アプリケーションを作成する方法を説明しました。

- ユーザーが操作できるマップを作成します。
- マップイベント `click` を処理します。
- Amazon Location Service API を呼び出し、特に`searchPlaceIndexForPosition`を使っている場所の地図を検索します。
- MapLibre マップコントロールを使用してマーカーを追加します。

最終のアプリケーションを表示する

このアプリケーションの最終的なソースコードはこのセクションに含まれています。最終プロジェクトは でも GitHub確認できます。

また、 で GitHub API キーの代わりに Amazon Cognito を使用するアプリケーションのバージョンを見つけることもできます。

Overview

各タブを選択すると、このクイックスタートチュートリアルのファイルの最終的なソースコードが表示されます。

が ファイルを読み取ります。

- quickstart.html — マップと検索結果の HTML 要素ホルダーを含む、アプリケーションのフレームワーク。
- main.css — アプリケーションのスタイルシート。
- main.js — ユーザーを認証し、マップを作成し、イベントを検索するアプリケーション用のスクリプト。`click`

quickstart.html

クイックスタートアプリケーション用の HTML フレームワーク。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Quick start tutorial</title>

    <!-- Styles -->
    <link href="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.css"
 rel="stylesheet" />
    <link href="main.css" rel="stylesheet" />
  </head>

  <body>
    ...
    <footer>This is a simple Amazon Location Service app. Pan and zoom. Click to see
 details about entities close to a point.</footer>

    <!-- JavaScript dependencies -->
    <script src="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.js"></script>
    <script src="https://unpkg.com/@aws/amazon-location-client@1.x/dist/
amazonLocationClient.js"></script>
    <script src="https://unpkg.com/@aws/amazon-location-utilities-auth-helper@1.x/
dist/amazonLocationAuthHelper.js"></script>

    <!-- JavaScript for the app -->
```

```
    <script src="main.js"></script>
  </body>
</html>
```

main.css

クイックスタートアプリケーションのスタイルシート。

```
* {
  box-sizing: border-box;
  font-family: Arial, Helvetica, sans-serif;
}

body {
  margin: 0;
}

header {
  background: #000000;
  padding: 0.5rem;
}

h1 {
  margin: 0;
  text-align: center;
  font-size: 1.5rem;
  color: #ffffff;
}

main {
  display: flex;
  min-height: calc(100vh - 94px);
}

#map {
  flex: 1;
}

aside {
  overflow-y: auto;
  flex: 0 0 30%;
  max-height: calc(100vh - 94px);
  box-shadow: 0 1px 1px 0 #001c244d, 1px 1px 1px 0 #001c2426, -1px 1px 1px 0
 #001c2426;
```

```
  background: #f9f9f9;
  padding: 1rem;
}

h2 {
  margin: 0;
}

pre {
  white-space: pre-wrap;
  font-family: monospace;
  color: #16191f;
}

footer {
  background: #000000;
  padding: 1rem;
  color: #ffffff;
}
```

main.js

クイックスタートアプリケーションのスタイルシート。##のテキストは、適切な Amazon Location オブジェクト名に置き換えてください。

```
// Amazon Location Service resource names:
const mapName = "explore.map";
const placesName = "explore.place";
const region = "your_region";
const apiKey = "v1.public.a1b2c3d4...

// Initialize a map
async function initializeMap() {
  // Initialize the map
  const mlglMap = new maplibregl.Map({
    container: "map", // HTML element ID of map element
    center: [-77.03674, 38.891602], // Initial map centerpoint
    zoom: 16, // Initial map zoom
    style: `https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor?key=${apiKey}`, // Defines the appearance of the map and authenticates
 using an API key
  });
```

```
  // Add navigation control to the top left of the map
  mlglMap.addControl(new maplibregl.NavigationControl(), "top-left");

  return mlglMap;
}

async function main() {
  // Create an authentication helper instance using an API key
  const authHelper = await amazonLocationAuthHelper.withAPIKey(apiKey);

  // Initialize map and Amazon Location SDK client
  const map = await initializeMap();
  const client = new amazonLocationClient.LocationClient({
    region,
    ...authHelper.getLocationClientConfig(), // Provides configuration required to
 make requests to Amazon Location
  });

  // Variable to hold marker that will be rendered on click
  let marker;

  // On mouse click, display marker and get results:
  map.on("click", async function (e) {
    // Remove any existing marker
    if (marker) {
      marker.remove();
    }

    // Render a marker on clicked point
    marker = new maplibregl.Marker().setLngLat([e.lngLat.lng,
 e.lngLat.lat]).addTo(map);

    // Set up parameters for search call
    let params = {
      IndexName: placesName,
      Position: [e.lngLat.lng, e.lngLat.lat],
      Language: "en",
      MaxResults: "5",
    };

    // Set up command to search for results around clicked point
    const searchCommand = new
 amazonLocationClient.SearchPlaceIndexForPositionCommand(params);
```

```
    try {
      // Make request to search for results around clicked point
      const data = await client.send(searchCommand);

      // Write JSON response data to HTML
      document.querySelector("#response").textContent = JSON.stringify(data,
 undefined, 2);

      // Display place label in an alert box
      alert(data.Results[0].Place.Label);
    } catch (error) {
      // Write JSON response error to HTML
      document.querySelector("#response").textContent = JSON.stringify(error,
 undefined, 2);

      // Display error in an alert box
      alert("There was an error searching.");
    }
  });
}

main();
```

次のステップ

クイックスタートチュートリアルを完了したので、Amazon Location Service を使用してアプリケーションを構築する方法を理解できるでしょう。Amazon Location をさらに活用するには、以下のリソースをご覧ください。

- Amazon Location Service コンセプトの詳細情報はこちら
- Amazon Locationの特徴や機能の使用方法に関する詳細情報はこちら
- Amazon Location を使用したコード例を見て、このサンプルを発展させて、より複雑なアプリケーションを構築する方法はこちら

Android アプリを作成する

このセクションでは、マップ、場所の検索機能、フォアグラウンドの追跡機能を備えた Android アプリケーションを作成します。まず、Amazon Location リソース、Amazon Cognito ID、アプリケーションの API キーを作成します。

トピック

アプリ用の Amazon Location リソースを作成する

まだ持っていない場合は、アプリケーションが利用される Amazon Location リソースを作成する必要があります。ここでは、アプリケーションにマップを表示するマップリソース、マップ上の位置を検索する場所インデックス、マップ全体のオブジェクトを追跡するトラッカーを作成します。

位置情報リソースをアプリケーションに追加

1. 使用するマップスタイルを選択します。

   a. Amazon Location コンソールの マップページで、[マップを作成] を選択してマップスタイルをプレビューします。

   b. 新しいマップリソースの [名前] と [説明] を追加します。マップリソースに使用する名前をメモしておきます。チュートリアルの後半でスクリプトファイルを作成するときに必要になります。

   c. マップには HERE マップstlye を選択することをお勧めします。

   > **Note**
   >
   > マップスタイルを選択すると、利用するマップデータプロバイダーも選択されます。配送車両や従業員など、ビジネスで使用する資産をアプリケーションで追跡またはルーティングする場合、使用できる位置情報プロバイダーは HERE のみとなります。詳細については、「AWS サービス規約」のセクション 82 を参照してください。

   d. Amazon Location の利用規約に同意し、[マップを作成] を選択します。選択したマップを、拡大、縮小、または任意の方向への画面移動をすることができます。

e. 新しいマップリソースに表示される Amazon リソースネーム (ARN) をメモします。チュートリアルの後半で、これを使って正しい認証を作成します。

2. 使用するプレースインデックスを選択します。

a. Amazon Location コンソールの [プレースインデックス」ページで、「プレイスーンデックスを作成」を選択します。

b. 新しいプレースインデックスリソースの 名前と 説明を追加します。プレースインデックスリソースに入力した名前をメモしておきます。チュートリアルの後半でスクリプトファイルを作成するときに必要になります。

c. データプロバイダーを選択します。

> **Note**
>
> ほぼすべてのケースでは、すでに選択したマッププロバイダーと一致するデータプロバイダーを選択します。これにより、検索が地図と一致するようになります。配送車両や従業員など、ビジネスで使用する資産をアプリケーションで追跡またはルーティングする場合、使用できる位置情報プロバイダーは HERE のみとなります。詳細については、「AWS サービス規約」のセクション 82 を参照してください。

d. [データストレージオプション] を選択します。このチュートリアルでは結果は保存されないため、いいえ、1 回のみを選択することができます。

e. Amazon Location の利用規約に同意し、[プレースインデックスを作成] を選択します。

f. 新しいプレースインデックスリソースに表示される ARN をメモします。このチュートリアルの次のセクションで、これを使って正しい認証を作成します。

3. Amazon Location コンソールを使用してトラッカーを作成するには。

a. Amazon Location Service コンソール を開きます。

b. 左のナビゲーションペインから、[トラッカー] を選択します。

c. トラッカーを作成を選択します。

d. すべての必須フィールドに入力します。

e. 位置フィルタリング では、デフォルト設定 を使用することをお勧めしますTimeBased。

f. 終了するトラッカーの作成を選択します。

アプリケーションの認証の設定

このチュートリアルで作成したアプリケーションは匿名で使用できます。つまり、ユーザーはアプリケーション AWS を使用するためにサインインする必要はありません。しかし、Amazon Location Service API を使用するには認証が必要です。API キーまたは Amazon Cognito のいずれかを使用して、匿名ユーザーの認証と承認を行うことができます。このチュートリアルでは、Amazon Cognito と API キーを使用してアプリケーションを認証します。

> **Note**
>
> Amazon Cognito または Amazon Location Service で API キーを使用する方法の詳細については、「Amazon Location Service へのアクセスを許可する」を参照してください。

以下のチュートリアルでは、 で作成したマップ、場所インデックス、トラッカーの認証を設定する方法と、Amazon Location のアクセス許可を設定する方法を示します。

認証を設定する。

1. Amazon Location コンソールに移動し、左側のメニューから API キーを選択します。
2. 「API キーの作成」をクリックします。API キーは、以前に作成した Amazon Location Service リソースと同じ AWS アカウントとリージョンに存在する必要があります。
3. 「API キーの作成」ページで必要な詳細を入力します。

   - 名前: API キーの名前を のように指定しますMyAppKey。
   - リソース: 前に作成した Amazon Location Service Map and Place インデックスリソースを選択します。「リソースの追加」を選択すると、複数のリソースを追加できます。これにより、指定されたリソースで API キーを使用できます。
   - アクション: この API キーに対して承認されたアクションを指定します。少なくとも、geo:GetMapと を選択してgeo:SearchPlaceIndexForPosition、チュートリアルが意図したとおりに機能することを確認します。
   - オプションで、説明、有効期限、タグ、またはリファラーを追加して、キーの使用を特定のドメインに制限https://www.example.comし、そのドメイン内でのみチュートリアルを機能させることができます。
4. API キーの作成をクリックして API キーを生成します。
5. API キーを表示を選択し、キー値をコピーして、チュートリアルで後で使用できるv1.public.a1b2c3d4ようにします。

追跡用の IAM ポリシーを作成する

1. 管理者権限を持つユーザーを使用して、https：//console.aws.amazon.com/iam/で IAM コンソールにサインインします。
2. ナビゲーションペインで、ポリシー を選択します。
3. コンテンツペインで、[ポリシーの作成] を選択します。
4. JSON オプションを選択し、この JSON ポリシーをコピーして JSON テキストボックスに貼り付けます。

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "geo:GetMapTile",
                    "geo:GetMapStyleDescriptor",
                    "geo:GetMapSprites",
                    "geo:GetMapGlyphs",
                    "geo:SearchPlaceIndexForPosition",
                    "geo:GetDevicePositionHistory",
                    "geo:BatchUpdateDevicePosition"
                ],
                "Resource": [
                    "arn:aws:geo:{Region}:{Account}:map/{MapName}",
                    "arn:aws:geo:{Region}:{Account}:place-index/{IndexName}",
                    "arn:aws:geo:{Region}:{Account}:tracker/{TrackerName}"
                ]
            }
        ]
    }
```

これは 追跡のポリシーの例です。独自のポリシーに例を使用するには、`Region`、`Account`、および TrackerNameプレースホルダーを置き換えます。

> **Note**
>
> 認証されていない ID プールはセキュリティで保護されていないインターネットサイトで公開されることを目的としていますが、標準で時間制限のある AWS 認証情報と交換されることに注意してください。
> 認証されていないアイデンティティプールに関連付けられている IAM ロールの範囲を適切に設定することが重要です。Amazon Location Service で Amazon Cognito でポリシーを使用し、適切にスコープ設定する方法の詳細については、「Amazon Location Service へのアクセス権の付与」を参照してください。

5. 確認と作成ページで、ポリシー名フィールドの名前を指定します。ポリシーによって付与されたアクセス許可を確認し、ポリシーの作成を選択して作業を保存します。

新しいポリシーが管理ポリシーの一覧に表示され、アタッチの準備ができます。

追跡用の認証を設定する

1. Amazon Cognito コンソール でマップアプリケーションの認証を設定します。
2. ID プールページを開きます。

> **Note**
>
> 作成するプールは、前のセクションで作成した Amazon Location Service リソースと同じ AWS アカウントと AWS リージョンに存在する必要があります。

3. ID プールの作成 を選択します。
4. ID プールの信頼を設定するステップから始めます。ユーザーアクセス認証では、ゲストアクセス を選択し、次を押します。
5. アクセス許可の設定ページで、既存の IAM ロールを使用する を選択し、前のステップで作成した IAM ロールの名前を入力します。準備ができたら、 の横にある を押して次のステップに進みます。
6. 「プロパティの設定」ページで、ID プールの名前を指定します。次に、次へ を押します。
7. 確認と作成ページで、存在するすべての情報を確認してから、ID プールの作成 を押します。
8. ID プールページを開き、先ほど作成した ID プールを選択します。次に、ブラウザスクリプトで後 IdentityPoolId で使用する をコピーまたは書き留めます。

ベースの Android アプリケーションの作成

このチュートリアルでは、マップを埋め込み、ユーザーがマップ上の場所にあるものを見つけられるようにする Android アプリケーションを作成します。

まず、Android Studio の新しいプロジェクトウィザードを使用して空の Kotlin アプリケーションを作成します。

空のアプリケーションを作成するには (AndroidStudio)

1. を起動します AndroidStudio。メニューを開き、ファイル 、新規 、新規プロジェクト を選択します。
2. [スマートフォンとタブレット] タブから [アクティビティを空にする] を選択し、[次へ] を選択します。
3. [アプリケーション名]、[パッケージ名]、[保存場所]を選択します。
4. [言語]のドロップダウンリストから「Kotlin」を選択します。
5. [Finish] を選択して空白のアプリケーションを作成します。

アプリケーションにインタラクティブマップを追加

基本的なアプリケーションを作成したら、アプリケーションにマップコントロールを追加できます。このチュートリアルでは、API キーを使用してマップビューを管理します。マップコントロール自体は、API キー と MapLibre を持つネイティブライブラリ の一部であり MapLibre、マップデータは Amazon Location から取得されます。

アプリケーションにマップを追加するには、次のアクションを実行する必要があります。

- MapLibre 依存関係をプロジェクトに追加します。
- 構成でマップビューコードを設定します。
- マップを表示するコードを記述します。

アプリにマップを追加するには、次の手順に従います。

1. MapLibre 依存関係をプロジェクトに追加する

a. で AndroidStudio、表示メニューを選択し、ツールウィンドウ 、プロジェクト を選択します。[プロジェクト] ウィンドウが開き、プロジェクト内のすべてのファイルにアクセスできます。

b. プロジェクトウィンドウで gradle を開き、ツリービューで `libs.versions.toml` ファイルを開きます。これで編集する `libs.versions.toml` ファイルが開きます。次に、以下のバージョンとライブラリデータを `libs.versions.toml` ファイルに追加します。

```
[versions]
 ...
 auth = "0.2.4"
 tracking = "0.2.4"

 [libraries]
 ...
 auth = { group = "software.amazon.location", name = "auth", version.ref =
 "auth" }
 tracking = { module = "software.amazon.location:tracking", version.ref =
 "tracking" }

 [plugins]
 ...
```

c. `libs.versions.toml` ファイルの編集が完了したら、プロジェクトを再同期 AndroidStudio する必要があります。`libs.versions.toml` 編集ウィンドウの上部で、同期するように AndroidStudio 求められます。「今すぐ同期」を選択して、続行する前にプロジェクトを同期します。

d. プロジェクトウィンドウで、ツリービューで Gradle スクリプトを開き、アプリケーションモジュールの `build.gradle` ファイルを選択します。これで編集する `build.gradle` ファイルが開きます。

e. ファイルの下部にある依存関係セクションに、次の依存関係を追加します。

```
dependencies {
    ...
    implementation(libs.org.maplibre.gl)
 }
```

f. Gradle 依存関係の追加が完了したら、Android Studio はプロジェクトを再同期する必要があります。build.gradle 編集ウィンドウの Android Studio の上部で、今すぐ同期を選択してプロジェクトを同期してから続行します。

2. 次に、構成でマップビューコードを設定します。以下のステップを使用します。

   a. プロジェクトウィンドウから、ツリービューで App、Java、######を開き、ui フォルダ内の ui フォルダに移動して、ビューディレクトリを作成します。

   b. 内部ビューディレクトリで MapLoadScreen.kt ファイルを作成します。

   c. 以下のコードを MapLoadScreen.kt ファイルに追加します。

```
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxHeight
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.viewinterop.AndroidView
import org.maplibre.android.maps.OnMapReadyCallback

 @Composable
 fun MapLoadScreen(
     mapReadyCallback: OnMapReadyCallback,
 ) {
     Box(
         modifier = Modifier
             .fillMaxWidth()
             .fillMaxHeight(),
     ) {
         MapView(mapReadyCallback)
     }
 }

 @Composable
 fun MapView(mapReadyCallback: OnMapReadyCallback) {
     AndroidView(
         factory = { context ->
             val mapView = org.maplibre.android.maps.MapView(context)
             mapView.onCreate(null)
             mapView.getMapAsync(mapReadyCallback)
             mapView
         },
     )
 }
```

3. マップを表示するコードを記述します。

a. 以下のコードを MainActivity.kt ファイルに追加します。

```
// ...other imports
import org.maplibre.android.MapLibre
import org.maplibre.android.camera.CameraPosition
import org.maplibre.android.geometry.LatLng
import org.maplibre.android.maps.MapLibreMap
import org.maplibre.android.maps.OnMapReadyCallback
import org.maplibre.android.maps.Style

 class MainActivity : ComponentActivity(), OnMapReadyCallback {
     private val region = "YOUR_AWS_REGION"
     private val mapName = "YOUR_AWS_MAP_NAME"
     private val apiKey = "YOUR_AWS_API_KEY"
     override fun onCreate(savedInstanceState: Bundle?) {
         MapLibre.getInstance(this)
         super.onCreate(savedInstanceState)
         setContent {
             TestMapAppTheme {
                 Surface(
                     modifier = Modifier.fillMaxSize(),
                     color = MaterialTheme.colorScheme.background
                 ) {
                     MapLoadScreen(this)
                 }
             }
         }
     }

     override fun onMapReady(map: MapLibreMap) {
         map.setStyle(
             Style.Builder()
                 .fromUri(
                     "https://maps.geo.$region.amazonaws.com/maps/v0/maps/
$mapName/style-descriptor?key=$apiKey"
                 ),
         ) {
             map.uiSettings.isAttributionEnabled = true
             map.uiSettings.isLogoEnabled = false
             map.uiSettings.attributionGravity = Gravity.BOTTOM or Gravity.END
             val initialPosition = LatLng(47.6160281982247,
 -122.32642111977668)
             map.cameraPosition = CameraPosition.Builder()
```

```
                .target(initialPosition)
                .zoom(14.0)
                .build()
        }
    }
}
```

b. MainActivity.kt ファイルを保存します。これで、アプリケーションを構築できます。これを実行するには、 でエミュレートするデバイスを設定するか AndroidStudio、デバイスでアプリを使用する必要がある場合があります。このアプリを使用して、マップコントロールの動作を確認します。パンするには、マップ上でドラッグし、ピンチしてズームします。

次のセクションでは、マップにマーカーを追加し、マップを移動するときにマーカーがある場所のアドレスを表示します。

アプリケーションへのリバースジオコーディング検索の追加

次に、アプリケーションにリバースジオコーディング検索を追加し、その場所に項目を見つけます。Android アプリの使用を簡素化するために、画面の中央を検索します。新しい場所を見つけるには、検索したい場所にマップを移動します。マップの先頭にマーカーを配置して、検索場所を示します。

リバースジオコーディング検索を追加すると、2 つの部分で構成されます。

- 画面の先頭にマーカーを追加して、検索する場所をユーザーに示します。
- 結果のテキストボックスを追加し、マーカーの場所を検索してテキストボックスに表示します。

アプリケーションにマーカーを追加するには

1. このイメージを app/res/drawableフォルダに として保存します red_marker.png (からイメージにアクセスすることもできますGitHub。または、イメージを作成することもできます。表示したくない部分には、透明度のある .png ファイルを使用することもできます。
2. 次のコードを MapLoadScreen." ファイルに追加します。

```
// ...other imports
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.size
import androidx.compose.ui.Alignment
```

```
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import com.amazon.testmapapp.R

@Composable
fun MapLoadScreen(
    mapReadyCallback: OnMapReadyCallback,
) {
    Box(
        modifier = Modifier
            .fillMaxWidth()
            .fillMaxHeight(),
    ) {
        MapView(mapReadyCallback)
        Box(
            modifier = Modifier
                .align(Alignment.Center),
        ) {
            Image(
                painter = painterResource(id = R.drawable.red_marker),
                contentDescription = "marker",
                modifier = Modifier
                    .size(40.dp)
                    .align(Alignment.Center),
            )
        }
    }
}

@Composable
fun MapView(mapReadyCallback: OnMapReadyCallback) {
    AndroidView(
        factory = { context ->
            val mapView = org.maplibre.android.maps.MapView(context)
            mapView.onCreate(null)
            mapView.getMapAsync(mapReadyCallback)
            mapView
        },
    )
}
```

3. アプリを構築して実行し、機能をプレビューします。

これで、アプリの画面にマーカーが表示されるようになりました。この場合は、動かない静止画像です。これは、検索するマップビューの構造を表示するために使用されます。次の手順では、その場所に検索を追加します。

場所のリバースジオコーディング検索をアプリケーションに追加するには

1. プロジェクトウィンドウで、ツリービューで gradle to `libs.versions.toml` file を開きます。これで編集する `libs.versions.toml` ファイルが開きます。次に、以下のバージョンとライブラリデータを `libs.versions.toml` ファイルに追加します。

```
[versions]
 ...
 okhttp = "4.12.0"

 [libraries]
 ...
 com-squareup-okhttp3 = { group = "com.squareup.okhttp3", name = "okhttp",
 version.ref = "okhttp" }

 [plugins]
 ...
```

2. `libs.versions.toml` ファイルの編集が完了したら、プロジェクトを再同期 AndroidStudio する必要があります。`libs.versions.toml` 編集ウィンドウの上部で、同期するように AndroidStudio に指示します。「今すぐ同期」を選択して、続行する前にプロジェクトを同期します。
3. プロジェクトウィンドウで、ツリービューで Gradle スクリプトを開き、アプリケーションモジュールの `build.gradle` ファイルを選択します。これで編集する `build.gradle` ファイルが開きます。
4. ファイルの下部にある依存関係セクションに、次の依存関係を追加します。

```
dependencies {
    ...
    implementation(libs.com.squareup.okhttp3)
 }
```

5. Gradle の依存関係の編集が完了したら、プロジェクトを再同期 AndroidStudio する必要があります。`build.gradle` 編集ウィンドウの上部で、同期するように AndroidStudio 求められます。続行する前にプロジェクトを同期SyncNowするには、 を選択します。

6. ツリービューで、データ をリクエストディレクトリに追加し、ReverseGeocodeRequest.ktデータクラスを作成します。クラスに次のコードを追加します。

```
import com.google.gson.annotations.SerializedName

data class ReverseGeocodeRequest(
    @SerializedName("Language")
    val language: String,
    @SerializedName("MaxResults")
    val maxResults: Int,
    @SerializedName("Position")
    val position: ListDouble
)
```

7. ツリービューで、データをレスポンスディレクトリに追加し、ReverseGeocodeResponse.ktデータクラスを作成します。その中に次のコードを追加します。

```
import com.google.gson.annotations.SerializedName

data class ReverseGeocodeResponse(
    @SerializedName("Results")
    val results: ListResult
)

data class Result(
    @SerializedName("Place")
    val place: Place
)

data class Place(
    @SerializedName("Label")
    val label: String
)
```

8. プロジェクトウィンドウから、ツリービューで App、Java、######を開き、ui フォルダ create viewModel ディレクトリ内の ui フォルダに移動します。
9. viewModel ディレクトリ内で MainViewModel.kt ファイルを作成します。
10. 以下のコードを MainViewModel.kt ファイルに追加します。

```
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.setValue
import androidx.lifecycle.ViewModel
import com.amazon.testmapapp.data.request.ReverseGeocodeRequest
import com.amazon.testmapapp.data.response.ReverseGeocodeResponse
import com.google.gson.Gson
import java.io.IOException
import okhttp3.Call
import okhttp3.Callback
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.OkHttpClient
import okhttp3.Request
import okhttp3.RequestBody.Companion.toRequestBody
import okhttp3.Response
import org.maplibre.android.geometry.LatLng
import org.maplibre.android.maps.MapLibreMap

class MainViewModel : ViewModel() {
    var label by mutableStateOf("")
    var isLabelAdded: Boolean by mutableStateOf(false)
    var client = OkHttpClient()
    var mapLibreMap: MapLibreMap? = null

    fun reverseGeocode(latLng: LatLng, apiKey: String) {
        val region = "YOUR_AWS_REGION"
        val indexName = "YOUR_AWS_PLACE_INDEX"
        val url =
            "https://places.geo.${region}.amazonaws.com/places/v0/indexes/
${indexName}/search/position?key=${apiKey}"

        val requestBody = ReverseGeocodeRequest(
            language = "en",
            maxResults = 1,
            position = listOf(latLng.longitude, latLng.latitude)
        )
        val json = Gson().toJson(requestBody)

        val mediaType = "application/json".toMediaTypeOrNull()
        val request = Request.Builder()
            .url(url)
            .post(json.toRequestBody(mediaType))
            .build()
```

```
        client.newCall(request).enqueue(object : Callback {
            override fun onFailure(call: Call, e: IOException) {
                e.printStackTrace()
            }

            override fun onResponse(call: Call, response: Response) {
                if (response.isSuccessful) {
                    val jsonResponse = response.body?.string()

                    val reverseGeocodeResponse =
                        Gson().fromJson(jsonResponse,
 ReverseGeocodeResponse::class.java)

                    val responseLabel =
 reverseGeocodeResponse.results.firstOrNull()?.place?.label

                    if (responseLabel != null) {
                        label = responseLabel
                        isLabelAdded = true
                    }
                }
            }
        })
    }
}
```

11. まだ開いていない場合は、前の手順と同様にMapLoadScreen.kt ファイルを開きます。次のコードを追加します。これにより、Compose Text ビューが作成され、その場所にリバースジオコーディングの検索結果が表示されます。

```
// ...other imports
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.material3.Text
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.testTag
```

```
import androidx.compose.ui.semantics.contentDescription
import androidx.compose.ui.semantics.semantics
import androidx.compose.ui.unit.sp
import com.amazon.testmapapp.ui.viewModel.MainViewModel

@Composable
fun MapLoadScreen(
    mapReadyCallback: OnMapReadyCallback,
    mainViewModel: MainViewModel,
) {
    Box(
        modifier = Modifier
            .fillMaxWidth()
            .fillMaxHeight(),
    ) {
        MapView(mapReadyCallback)
        Box(
            modifier = Modifier
                .align(Alignment.Center),
        ) {
            Image(
                painter = painterResource(id = R.drawable.red_marker),
                contentDescription = "marker",
                modifier = Modifier
                    .size(40.dp)
                    .align(Alignment.Center),
            )
        }
        if (mainViewModel.isLabelAdded) {
            Column(
                modifier = Modifier.fillMaxSize(),
                verticalArrangement = Arrangement.Bottom
            ) {
                Box(
                    modifier = Modifier
                        .fillMaxWidth()
                        .background(Color.White),
                ) {
                    Text(
                        text = mainViewModel.label,
                        modifier = Modifier
                            .padding(16.dp)
                            .align(Alignment.Center)
                            .testTag("label")
```

```
                            .semantics {
                                contentDescription = "label"
                            },
                        fontSize = 14.sp,
                    )
                }
                Spacer(modifier = Modifier.height(80.dp))
            }
        }
    }
}

@Composable
fun MapView(mapReadyCallback: OnMapReadyCallback) {
    AndroidView(
        factory = { context ->
            val mapView = org.maplibre.android.maps.MapView(context)
            mapView.onCreate(null)
            mapView.getMapAsync(mapReadyCallback)
            mapView
        },
    )
}
```

12. アプリの java の のパッケージ名フォルダで AndroidStudio、 `MainActivity.kt` ファイルを開きます。図のようにコードを変更します。

```
// ...other imports
import androidx.activity.viewModels
import com.amazon.testmapapp.ui.viewModel.MainViewModel

 class MainActivity : ComponentActivity(), OnMapReadyCallback,
 MapLibreMap.OnCameraMoveStartedListener, MapLibreMap.OnCameraIdleListener {

     private val mainViewModel: MainViewModel by viewModels()
     private val region = "YOUR_AWS_REGION"
     private val mapName = "YOUR_AWS_MAP_NAME"
     private val apiKey = "YOUR_AWS_API_KEY"
     override fun onCreate(savedInstanceState: Bundle?) {
         MapLibre.getInstance(this)
         super.onCreate(savedInstanceState)
         setContent {
             TestMapAppTheme {
```

```
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    MapLoadScreen(this, mainViewModel)
                }
            }
        }
    }

    override fun onMapReady(map: MapLibreMap) {
        map.setStyle(
            Style.Builder()
                .fromUri(
                    "https://maps.geo.$region.amazonaws.com/maps/v0/maps/$mapName/
style-descriptor?key=$apiKey"
                ),
        ) {
            map.uiSettings.isAttributionEnabled = true
            map.uiSettings.isLogoEnabled = false
            map.uiSettings.attributionGravity = Gravity.BOTTOM or Gravity.END
            val initialPosition = LatLng(47.6160281982247, -122.32642111977668)
            map.cameraPosition = CameraPosition.Builder()
                .target(initialPosition)
                .zoom(14.0)
                .build()

            map.addOnCameraMoveStartedListener(this)
            map.addOnCameraIdleListener(this)
            map.cameraPosition.target?.let { latLng ->
                mainViewModel.reverseGeocode(
                    LatLng(
                        latLng.latitude,
                        latLng.longitude
                    ), apiKey
                )
            }
        }
    }
   override fun onCameraMoveStarted(p0: Int) {
       mainViewModel.label = ""
       mainViewModel.isLabelAdded = false
   }
```

```
    override fun onCameraIdle() {
        if (!mainViewModel.isLabelAdded) {
            mainViewModel.mapLibreMap?.cameraPosition?.target?.let { latLng ->
                mainViewModel.reverseGeocode(
                    LatLng(
                        latLng.latitude,
                        latLng.longitude
                    ), apiKey
                )
            }
        }
    }
}
```

このコードはマップビューで動作します。仮想カメラの位置は、 のマップビューを定義します MapLibre。マップを動かすことは、その仮想カメラを動かすことと考えることができます。

- ViewModel にはラベル変数があります。この変数はデータを構成テキストビューに設定します。
- onMapReady: この関数が更新され、2 つの新しいイベントが登録されます。
- onCameraMove イベントは、ユーザーがマップを移動するたびに発生します。地図を移動するときは、ユーザーが地図を移動し終わるまで検索を非表示にするのが一般的です。
- onCameraIdle イベントは、ユーザーがマップの移動を一時停止したときに発生します。このイベントは、リバースジオコード関数を呼び出して、マップの先頭で検索します。
- `reverseGeocode(latLng: LatLng, apiKey: String)`: イベント で呼び出されるこの関数は onCameraIdle、マップの先頭で場所を検索し、ラベルを更新して結果を表示します。カメラターゲットを使用します。カメラターゲットは、マップ (カメラが見ている場所) の縮小を定義します。

13. ファイルを保存し、アプリを構築して実行し、機能するかどうかを確認します。

検索機能を備えたクイックスタートアプリケーションが完了しました。

アプリケーションへの追跡の追加

サンプルアプリケーションに追跡を追加するには、次の手順に従います。

1. SDK の依存関係の追跡と認証をプロジェクトに追加します。
2. AndroidManifest.xml ファイルに許可とサービスエントリを含めます。

3. 構成で追跡ボタンコードの開始/停止を設定します。
4. LocationTracker オブジェクトを作成するためのコードを追加し、追跡を開始および停止します。
5. Android Emulator を使用してテストルートを作成します。

1. SDK の依存関係の追跡と認証をプロジェクトに追加します。

   a. プロジェクトウィンドウで gradle を開き、ツリービューで libs.versions.toml ファイルを開きます。これで編集する libs.versions.toml ファイルが開きます。次に、以下のバージョンとライブラリデータを libs.versions.toml ファイルに追加します。

```
[versions]
 ...
 auth = "0.0.1"
 tracking = "0.0.1"

 [libraries]
 ...
 auth = { group = "software.amazon.location", name = "auth", version.ref =
 "auth" }
 tracking = { module = "software.amazon.location:tracking", version.ref =
 "tracking" }

 [plugins]
 ...
```

   b. libs.versions.toml ファイルの編集が完了したら、プロジェクトを再同期 AndroidStudio する必要があります。libs.versions.toml 編集ウィンドウの上部で、同期するように AndroidStudio に指示します。「今すぐ同期」を選択して、続行する前にプロジェクトを同期します。

   c. プロジェクトウィンドウで、ツリービューで Gradle スクリプトを開き、アプリケーションモジュールの build.gradle ファイルを選択します。これで編集する build.gradle ファイルが開きます。

   d. ファイルの下部にある依存関係セクションに、次の依存関係を追加します。

```
dependencies {
    ...
    implementation(libs.auth)
    implementation(libs.tracking)
```

```
}
```

e. Gradle の依存関係の編集が完了したら、プロジェクトを再同期 AndroidStudio する必要があります。build.gradle 編集ウィンドウの上部で、 が同期するように AndroidStudio 促します。続行する前にプロジェクトを同期SyncNowするには、 を選択します。

2. AndroidManifest.xml ファイルに許可とサービスエントリを含めます。

   - に正しいアクセス許可とサービスエントリを含めるにはAndroidManifest.xml file、次のコードで ファイルを更新します。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AndroidQuickStartApp"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.AndroidQuickStartApp">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

3. Compose で追跡ボタンコードの開始/停止を設定します。

a. ic_pause および ic_play という名前の描画可能な の下に、再生と一時停止の 2 つのイメージを res で追加します。 からイメージにアクセスすることもできますGitHub。

b. まだ開いていない場合は、前の手順と同様にMapLoadScreen.kt ファイルを開きます。次のコードを追加します。これにより構成ボタンビューが作成され、クリックすると追跡を開始および停止できます。

```
// ...other imports
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults

@Composable
fun MapLoadScreen(
    mapReadyCallback: OnMapReadyCallback,
    mainViewModel: MainViewModel,
    onStartStopTrackingClick: () -> Unit
) {
    Box(
        modifier = Modifier
            .fillMaxWidth()
            .fillMaxHeight(),
    ) {
        MapView(mapReadyCallback)
        Box(
            modifier = Modifier
                .align(Alignment.Center),
        ) {
            Image(
                painter = painterResource(id = R.drawable.red_marker),
                contentDescription = "marker",
                modifier = Modifier
                    .size(40.dp)
                    .align(Alignment.Center),
            )
        }
        if (mainViewModel.isLabelAdded) {
            Column(
                modifier = Modifier.fillMaxSize(),
                verticalArrangement = Arrangement.Bottom
            ) {
                Box(
                    modifier = Modifier
                        .fillMaxWidth()
```

```
                        .background(Color.White),
                ) {
                    Text(
                        text = mainViewModel.label,
                        modifier = Modifier
                            .padding(16.dp)
                            .align(Alignment.Center)
                            .testTag("label")
                            .semantics {
                                contentDescription = "label"
                            },
                        fontSize = 14.sp,
                    )
                }
                Spacer(modifier = Modifier.height(80.dp))
            }
        }
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(bottom = 16.dp),
            horizontalAlignment = Alignment.CenterHorizontally,
            verticalArrangement = Arrangement.Bottom,
        ) {
            Button(
                onClick = onStartStopTrackingClick,
                modifier = Modifier
                    .padding(horizontal = 16.dp)
            ) {
                Text(
                    text = if
(mainViewModel.isLocationTrackingForegroundActive) "Stop tracking" else "Start
tracking",
                    color = Color.Black
                )
                Spacer(modifier = Modifier.size(ButtonDefaults.IconSpacing))
                Image(
                    painter = painterResource(id = if
(mainViewModel.isLocationTrackingForegroundActive) R.drawable.ic_pause else
R.drawable.ic_play),
                    contentDescription = if
(mainViewModel.isLocationTrackingForegroundActive) "stop_tracking" else
"start_tracking"
                )
```

```
            }
        }
    }
}

@Composable
fun MapView(mapReadyCallback: OnMapReadyCallback) {
    AndroidView(
        factory = { context ->
            val mapView = org.maplibre.android.maps.MapView(context)
            mapView.onCreate(null)
            mapView.getMapAsync(mapReadyCallback)
            mapView
        },
    )
}
```

4. LocationTracker オブジェクトを作成するためのコードを追加し、追跡を開始および停止します。

   a. `MainViewModel.kt` ファイル内に次のコードを追加します。

```
...
var isLocationTrackingForegroundActive: Boolean by mutableStateOf(false)
var locationTracker: LocationTracker? = null
```

   b. 以下のコードを `MainActivity.kt` ファイルに追加します。

```
// ...other imports
import software.amazon.location.auth.AuthHelper
import software.amazon.location.auth.LocationCredentialsProvider
import software.amazon.location.tracking.LocationTracker
import software.amazon.location.tracking.aws.LocationTrackingCallback
import software.amazon.location.tracking.config.LocationTrackerConfig
import software.amazon.location.tracking.database.LocationEntry
import software.amazon.location.tracking.filters.DistanceLocationFilter
import software.amazon.location.tracking.filters.TimeLocationFilter
import software.amazon.location.tracking.util.TrackingSdkLogLevel

class MainActivity : ComponentActivity(), OnMapReadyCallback,
    MapLibreMap.OnCameraMoveStartedListener, MapLibreMap.OnCameraIdleListener {

    private val mainViewModel: MainViewModel by viewModels()
```

```
    private val poolId = "YOUR_AWS_IDENTITY_POOL_ID"
    private val trackerName = "YOUR_AWS_TRACKER_NAME"
    private val region = "YOUR_AWS_REGION"
    private val mapName = "YOUR_AWS_MAP_NAME"
    private val apiKey = "YOUR_AWS_API_KEY"
    private val coroutineScope = MainScope()
    private lateinit var locationCredentialsProvider:
LocationCredentialsProvider
    private lateinit var authHelper: AuthHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        MapLibre.getInstance(this)
        super.onCreate(savedInstanceState)
        setContent {
            TestMapAppTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colorScheme.background
                ) {
                    MapLoadScreen(this, mainViewModel, onStartStopTrackingClick
= {
                        if (mainViewModel.isLocationTrackingForegroundActive) {
                            mainViewModel.isLocationTrackingForegroundActive =
false
                            mainViewModel.locationTracker?.stop()
                        } else {
                            if (checkLocationPermission(this))
return@MapLoadScreen
                            mainViewModel.isLocationTrackingForegroundActive =
true

mainViewModel.locationTracker?.start(locationTrackingCallback = object :
                                LocationTrackingCallback {
                                override fun
onLocationAvailabilityChanged(locationAvailable: Boolean) {
                                }

                                override fun onLocationReceived(location:
LocationEntry) {
                                }

                                override fun onUploadSkipped(entries:
LocationEntry) {
                                }
```

```
                                override fun onUploadStarted(entries:
ListLocationEntry) {
                                }

                                override fun onUploaded(entries:
ListLocationEntry) {
                                }

                            })
                        }
                    })
                }
            }
        }
        authenticateUser()
    }

    private fun authenticateUser() {
        coroutineScope.launch {
            authHelper = AuthHelper(applicationContext)
            locationCredentialsProvider =
authHelper.authenticateWithCognitoIdentityPool(
                poolId,
            )
            locationCredentialsProvider.let {
                val config = LocationTrackerConfig(
                    trackerName = trackerName,
                    logLevel = TrackingSdkLogLevel.DEBUG,
                    latency = 1000,
                    frequency = 5000,
                    waitForAccurateLocation = false,
                    minUpdateIntervalMillis = 5000,
                )
                mainViewModel.locationTracker = LocationTracker(
                    applicationContext,
                    it,
                    config,
                )

mainViewModel.locationTracker?.enableFilter(TimeLocationFilter())

mainViewModel.locationTracker?.enableFilter(DistanceLocationFilter())
            }
```

```
        }
    }

    private fun checkLocationPermission(context: Context) =
 ActivityCompat.checkSelfPermission(
        context,
        Manifest.permission.ACCESS_FINE_LOCATION,
    ) != PackageManager.PERMISSION_GRANTED &&
 ActivityCompat.checkSelfPermission(
        context,
        Manifest.permission.ACCESS_COARSE_LOCATION,
    ) != PackageManager.PERMISSION_GRANTED

    override fun onMapReady(map: MapLibreMap) {
        map.setStyle(
            Style.Builder()
                .fromUri(
                    "https://maps.geo.$region.amazonaws.com/maps/v0/maps/
$mapName/style-descriptor?key=$apiKey"
                ),
        ) {
            mainViewModel.mapLibreMap = map
            map.uiSettings.isAttributionEnabled = true
            map.uiSettings.isLogoEnabled = false
            map.uiSettings.attributionGravity = Gravity.BOTTOM or Gravity.END
            val initialPosition = LatLng(47.6160281982247, -122.32642111977668)
            map.cameraPosition = CameraPosition.Builder()
                .target(initialPosition)
                .zoom(14.0)
                .build()

            map.addOnCameraMoveStartedListener(this)
            map.addOnCameraIdleListener(this)
            map.cameraPosition.target?.let { latLng ->
                mainViewModel.reverseGeocode(
                    LatLng(
                        latLng.latitude,
                        latLng.longitude
                    ), apiKey
                )
            }
        }
    }
```

```
    override fun onCameraMoveStarted(p0: Int) {
        mainViewModel.label = ""
        mainViewModel.isLabelAdded = false
    }

    override fun onCameraIdle() {
        if (!mainViewModel.isLabelAdded) {
            mainViewModel.mapLibreMap?.cameraPosition?.target?.let { latLng ->
                mainViewModel.reverseGeocode(
                    LatLng(
                        latLng.latitude,
                        latLng.longitude
                    ), apiKey
                )
            }
        }
    }
}
```

上記のコードは、 でLocationTrackerオブジェクトを作成する方法AuthHelperと、 で追跡を開始および停止する方法を示しています LocationTracker。

- `authenticateUser()`: このメソッドは AuthHelper および LocationTracker オブジェクトを作成します。
- `onStartStopTrackingClick`: このコールバックは、ユーザーが追跡の開始/停止ボタンをクリックするとトリガーされます。このボタンは、追跡 SDK で追跡を開始/停止します。

5. Android Emulator を使用してテストルートを作成します。

   a. Android Studio を使用して AVD を起動してエミュレータを開きます。

   b. エミュレータツールバーのその他 (3 つのドット) アイコンをクリックして、拡張コントロールを開きます。

   c. サイドバーから Location を選択して Location を開きます。

   d. GPX データを使用してルートを作成するか、マップをクリックして送信元データと送信先データを選択します。

   e. 「ROUTE を再生」をクリックしてシミュレーションを開始し、GPS ルートのシミュレーションを開始します。

   f. アプリケーションを実行し、シミュレートされたルートの処理方法を監視してアプリケーションをテストします。

これは Android クイックスタートアプリケーションの完全なデモです。

次のステップ

このアプリケーションのソースコードは、 で入手できますGitHub。

Amazon Location をさらに活用するには、以下のリソースをご覧ください。

- Amazon Location Service の概念をより深く知る
- Amazon Location の特徴や機能の使用方法に関する詳細情報はこちら
- Amazon Location を使用したコード例を見て、このサンプルを発展させて、より複雑なアプリケーションを構築する方法はこちら

iOS アプリケーションの作成

このセクションでは、ある場所で検索し、フォアグラウンドで追跡できる iOS アプリケーションを作成します。まず、Amazon Location リソースとアプリケーションの Amazon Cognito ID を作成します。

トピック

- アプリ用の Amazon Location リソースを作成する
- アプリケーションの認証の設定
- ベース iOS アプリケーションの作成
- 初期コードのセットアップ
- アプリケーションにインタラクティブマップを追加
- アプリケーションに検索を追加する
- アプリケーションへの追跡の追加
- 次のステップ

アプリ用の Amazon Location リソースを作成する

まだ持っていない場合は、アプリケーションが利用される Amazon Location リソースを作成する必要があります。アプリケーションにマップを表示するマップリソース、マップ上の位置を検索する場所インデックス、マップ全体のオブジェクトを追跡するトラッカーを作成します。

位置情報リソースをアプリケーションに追加

1. 使用するマップスタイルを選択します。

   a. Amazon Location コンソールの マップページで、[マップを作成] を選択してマップスタイルをプレビューします。

   b. 新しいマップリソースの [名前] と [説明] を追加します。マップリソースに使用する名前をメモしておきます。チュートリアルの後半でスクリプトファイルを作成するときに必要になります。

   c. マップを選択します。

   > **Note**
   >
   > マップスタイルを選択すると、利用するマップデータプロバイダーも選択されます。配送車両や従業員など、ビジネスで使用する資産をアプリケーションで追跡またはルーティングする場合、使用できる位置情報プロバイダーは HERE のみとなります。詳細については、「AWS サービス規約」のセクション 82 を参照してください。

   d. Amazon Location の利用規約に同意し、[マップを作成] を選択します。選択したマップを、拡大、縮小、または任意の方向への画面移動をすることができます。

   e. 新しいマップリソースに表示される Amazon リソースネーム (ARN) をメモします。チュートリアルの後半で、これを使って正しい認証を作成します。

2. 使用するプレースインデックスを選択します。

   a. Amazon Location コンソールの [プレースインデックス」ページで、「プレイスーンデックスを作成」を選択します。

   b. 新しいプレースインデックスリソースの 名前と 説明を追加します。プレースインデックスリソースに入力した名前をメモしておきます。チュートリアルの後半でスクリプトファイルを作成するときに必要になります。

   c. データプロバイダーを選択します。

   > **Note**
   >
   > ほぼすべてのケースでは、すでに選択したマッププロバイダーと一致するデータプロバイダーを選択します。これにより、検索が地図と一致するようになります。

配送車両や従業員など、ビジネスで使用する資産をアプリケーションで追跡またはルーティングする場合、使用できる位置情報プロバイダーは HERE のみとなります。詳細については、「AWS サービス規約」のセクション 82 を参照してください。

d. [データストレージオプション] を選択します。このチュートリアルでは結果は保存されないため、いいえ、1 回のみを選択することができます。

e. Amazon Location の利用規約に同意し、[プレースインデックスを作成] を選択します。

f. 新しいプレースインデックスリソースに表示される ARN をメモします。このチュートリアルの次のセクションで、これを使って正しい認証を作成します。

3. Amazon Location コンソールを使用してトラッカーを作成するには。

a. Amazon Location Service コンソール を開きます。

b. 左のナビゲーションペインから、[トラッカー] を選択します。

c. トラッカーを作成を選択します。

d. すべての必須フィールドに入力します。

e. 位置フィルタリング では、デフォルト設定 を使用することをお勧めしますTimeBased。

f. 終了するトラッカーの作成を選択します。

アプリケーションの認証の設定

このチュートリアルで作成したアプリケーションは匿名で使用できます。つまり、ユーザーはアプリケーション AWS を使用するためにサインインする必要はありません。しかし、Amazon Location Service API を使用するには認証が必要です。Amazon Cognito を使用して、匿名ユーザーに認証と承認を提供します。このチュートリアルでは、Amazon Cognito を使用してアプリケーションを認証します。

> **Note**
>
> Amazon Location Service で Amazon Cognito 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 Location Service へのアクセスを許可する。

以下のチュートリアルでは、 で作成したマップ、場所インデックス、トラッカーの認証を設定する方法と、Amazon Location のアクセス許可を設定する方法を示します。

追跡用の IAM ポリシーを作成する

1. 管理者権限を持つユーザーを使用して、https：//console.aws.amazon.com/iam/で IAM コンソールにサインインします。
2. ナビゲーションペインで、ポリシー を選択します。
3. コンテンツペインで、[ポリシーの作成] を選択します。
4. JSON オプションを選択し、この JSON ポリシーをコピーして JSON テキストボックスに貼り付けます。

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "geo:GetMapTile",
                    "geo:GetMapStyleDescriptor",
                    "geo:GetMapSprites",
                    "geo:GetMapGlyphs",
                    "geo:SearchPlaceIndexForPosition",
                    "geo:GetDevicePositionHistory",
                    "geo:BatchUpdateDevicePosition"
                ],
                "Resource": [
                    "arn:aws:geo:{Region}:{Account}:map/{MapName}",
                    "arn:aws:geo:{Region}:{Account}:place-index/{IndexName}",
                    "arn:aws:geo:{Region}:{Account}:tracker/{TrackerName}"
                ]
            }
        ]
    }
```

これは 追跡のポリシーの例です。独自のポリシーに例を使用するには、`Region`、、`Account`、`IndexNameMapName`および TrackerNameプレースホルダーを置き換えます。

> **Note**
>
> 認証されていない ID プールはセキュリティで保護されていないインターネットサイトで公開されることを目的としていますが、標準で時間制限のある AWS 認証情報と交換されることに注意してください。
> 認証されていないアイデンティティプールに関連付けられている IAM ロールの範囲を適切に設定することが重要です。Amazon Location Service で Amazon Cognito でポリシーを使用し、適切にスコープ設定する方法の詳細については、「Amazon Location Service へのアクセス権の付与」を参照してください。

5. 確認と作成ページで、ポリシー名フィールドの名前を指定します。ポリシーによって付与されたアクセス許可を確認し、ポリシーの作成を選択して作業を保存します。

新しいポリシーが管理ポリシーの一覧に表示され、アタッチの準備ができます。

追跡用の認証を設定する

1. Amazon Cognito コンソール でマップアプリケーションの認証を設定します。
2. ID プールページを開きます。

> **Note**
>
> 作成するプールは、前のセクションで作成した Amazon Location Service リソースと同じ AWS アカウントと AWS リージョンに存在する必要があります。

3. ID プールの作成 を選択します。
4. ID プールの信頼を設定するステップから開始します。ユーザーアクセス認証では、ゲストアクセス を選択し、次を押します。
5. アクセス許可の設定ページで、既存の IAM ロールを使用する を選択し、前のステップで作成した IAM ロールの名前を入力します。準備ができたら、 の横にある を押して次のステップに進みます。
6. 「プロパティの設定」ページで、ID プールの名前を指定します。次に、次へ を押します。
7. 確認と作成ページで、存在するすべての情報を確認してから、アイデンティティプールの作成を押します。
8. ID プールページを開き、先ほど作成した ID プールを選択します。次に、ブラウザスクリプトで後 IdentityPoolId で使用する をコピーまたは書き留めます。

ベース iOS アプリケーションの作成

このチュートリアルでは、マップを埋め込む iOS アプリケーションを作成し、ユーザーがマップ上の場所にあるものを見つけられるようにします。

まず、Xcode のプロジェクトウィザードを使用して Swift アプリケーションを作成しましょう。

空のアプリケーションを作成するには (Xcode)

1. Xcode を開き、メニューからファイル 、新規 、新規プロジェクト を選択します。
2. iOS タブから、アプリ を選択し、次へ を選択します。
3. インターフェイスフィールドの入力 に、製品名 、組織識別子 、および を指定しますSwiftUI。次へ を選択して選択を確定します。
4. プロジェクトを保存する場所を選択し、作成ボタンを押して空のアプリケーションを作成します。

ベースアプリケーションを作成したら、サンプルアプリケーションに必要なパッケージをインストールする必要があります。

必要な依存関係のインストール

1. Xcode でプロジェクトを右クリックし、パッケージの追加... を選択します。これにより、パッケージウィンドウが開き、プロジェクトにパッケージを追加できます。
2. パッケージウィンドウで、次のパッケージを追加します。

   - Maplibre ネイティブパッケージの場合は、次の URL を使用します: https://github.com/maplibre/maplibre-gl-native-distribution。URL から、、 maplibre-gl-native-distributionのパッケージを追加しますMapbox。
   - Amazon Location 認証 iOS SDK の場合は、https://github.com/aws-geospatial/amazon-location-mobile-auth-sdk-ios という URL を使用します。URL から、、 amazon-location-mobile-auth-sdk-iosのパッケージを追加しますAmazonLocationiOSAuthSDK。
   - Amazon Location 追跡 iOS SDK の場合は、https://github.com/aws-geospatial/amazon-location-mobile-tracking-sdk-ios という URL を使用します。URL から、、 amazon-location-mobile-tracking-sdk-iosのパッケージを追加しますAmazonLocationiOSTrackingSDK。

初期コードのセットアップ

アプリでロケーション許可を有効にする

1. Xcode プロジェクトを開きます。
2. プロジェクトの Info.plist ファイルを見つけます。
3. アプリの要件に基づいて、ロケーション許可に必要なキーを追加します。キーは次のとおりです。

   - NSLocationWhenInUseUsageDescription: アプリケーションが使用中にロケーションアクセスを必要とする理由の説明。
   - NSLocationAlwaysAndWhenInUseUsageDescription: アプリが継続的なロケーションアクセスを必要とする理由の説明。

次に、アプリでリソース値を設定する必要があります。という名前の新しいファイルを追加Config.xcconfigし、Amazon コンソールで以前に作成した値を入力します。

```
REGION =
INDEX_NAME =
MAP_NAME =
IDENTITY_POOL_ID =
TRACKER_NAME =
```

1. 左側のナビゲーターセクションで、プロジェクトを選択します。
2. ターゲットセクションで、アプリを選択し、情報タブをクリックします。
3. 次のような値を持つ情報プロパティを追加します。
4. 以下の内容の Config.swift ファイルを追加します。これにより、バンドル情報ファイルから設定値が読み取られます。

```
import Foundation

enum Config {
    static let region = Bundle.main.object(forInfoDictionaryKey: "Region") as!
 String
    static let mapName = Bundle.main.object(forInfoDictionaryKey: "MapName") as!
 String
```

```
    static let indexName = Bundle.main.object(forInfoDictionaryKey: "IndexName")
 as! String
    static let identityPoolId = Bundle.main.object(forInfoDictionaryKey:
 "IdentityPoolId") as! String
    static let trackerName = Bundle.main.object(forInfoDictionaryKey:
 "TrackerName") as! String
}
```

5. という名前の新しいフォルダViewModelを作成し、その中にTrackingViewModel.swiftファイルを追加します。

```
import SwiftUI
import AmazonLocationiOSAuthSDK
import MapLibre

final class TrackingViewModel : ObservableObject {
    @Published var trackingButtonText = NSLocalizedString("StartTrackingLabel",
 comment: "")
    @Published var trackingButtonColor = Color.blue
    @Published var trackingButtonIcon = "play.circle"
    @Published var region : String
    @Published var mapName : String
    @Published var indexName : String
    @Published var identityPoolId : String
    @Published var trackerName : String
    @Published var showAlert = false
    @Published var alertTitle = ""
    @Published var alertMessage = ""
    @Published var centerLabel = ""

    var clientIntialised: Bool
    var client: LocationTracker!
    var authHelper: AuthHelper
    var credentialsProvider: LocationCredentialsProvider?
    var mlnMapView: MLNMapView?
    var mapViewDelegate: MapViewDelegate?
    var lastGetTrackingTime: Date?
    var trackingActive: Bool

    init(region: String, mapName: String, indexName: String, identityPoolId:
 String, trackerName: String) {
        self.region = region
        self.mapName = mapName
```

```
        self.indexName = indexName
        self.identityPoolId = identityPoolId
        self.trackerName = trackerName
        self.authHelper = AuthHelper()
        self.trackingActive = false
        self.clientIntialised = false
    }

    func authWithCognito(identityPoolId: String?) {
        guard let identityPoolId =
 identityPoolId?.trimmingCharacters(in: .whitespacesAndNewlines)
        else {
            alertTitle = NSLocalizedString("Error", comment: "")
            alertMessage = NSLocalizedString("NotAllFieldsAreConfigured", comment:
 "")
            showAlert = true
            return
        }
        credentialsProvider =
 authHelper.authenticateWithCognitoUserPool(identityPoolId: identityPoolId)
        initializeClient()
    }

    func initializeClient() {
        client = LocationTracker(provider: credentialsProvider!, trackerName:
 trackerName)
        clientIntialised = true
    }
}
```

アプリケーションにインタラクティブマップを追加

次に、アプリケーションにマップコントロールを追加します。このチュートリアルでは、 MapLibre と AWS API を使用して、アプリケーションでマップビューを管理します。マップコントロール自体は MapLibre GL Native iOS ライブラリの一部です。

1. 次のコードを使用して、Views フォルダの下に `MapView.swift` ファイルを追加します。

```
import SwiftUI
import MapLibre

struct MapView: UIViewRepresentable {
```

```
    var onMapViewAvailable: ((MLNMapView) -> Void)?
    var mlnMapView: MLNMapView?
    var trackingViewModel: TrackingViewModel

    func makeCoordinator() -> MapView.Coordinator {
        return Coordinator(self, trackingViewModel: trackingViewModel)
    }

    func makeUIView(context: Context) -> MLNMapView {
        let styleURL = URL(string: "https://maps.geo.
\(trackingViewModel.region).amazonaws.com/maps/v0/maps/
\(trackingViewModel.mapName)/style-descriptor")
        let mapView = MLNMapView(frame: .zero, styleURL: styleURL)
        mapView.autoresizingMask = [.flexibleWidth, .flexibleHeight]
        mapView.setZoomLevel(15, animated: true)
        mapView.showsUserLocation = true
        mapView.userTrackingMode = .follow
        context.coordinator.mlnMapView = mapView
        mapView.delegate = context.coordinator

        mapView.logoView.isHidden = true
        context.coordinator.addCenterMarker()

        onMapViewAvailable?(mapView)
        trackingViewModel.mlnMapView = mapView
        return mapView
    }

    func updateUIView(_ uiView: MLNMapView, context: Context) {
    }

    class Coordinator: NSObject, MLNMapViewDelegate, MapViewDelegate {
        var control: MapView
        var mlnMapView: MLNMapView?
        var trackingViewModel: TrackingViewModel
        var centerMarker: MLNPointAnnotation?

        public init(_ control: MapView, trackingViewModel: TrackingViewModel) {
            self.control = control
            self.trackingViewModel = trackingViewModel
            super.init()
            self.trackingViewModel.mapViewDelegate = self
        }
```

```
        func mapViewDidFinishRenderingMap(_ mapView: MLNMapView, fullyRendered:
 Bool) {
            if(fullyRendered) {
                mapView.accessibilityIdentifier = "MapView"
                mapView.isAccessibilityElement = false
            }
        }

        func addCenterMarker() {
            guard let mlnMapView = mlnMapView else {
                return
            }

            let centerCoordinate = mlnMapView.centerCoordinate
            let marker = MLNPointAnnotation()
            marker.coordinate = centerCoordinate
            marker.accessibilityLabel = "CenterMarker"
            mlnMapView.addAnnotation(marker)
            centerMarker = marker

            trackingViewModel.reverseGeocodeCenter(centerCoordinate:
 centerCoordinate, marker: marker)
        }

        func mapView(_ mapView: MLNMapView, regionDidChangeAnimated animated: Bool)
 {
            if let marker = centerMarker {
                                DispatchQueue.main.asyncAfter(deadline: .now() + 1.0)
 {
                    mapView.deselectAnnotation(marker, animated: false)
                    marker.coordinate = mapView.centerCoordinate
                    let centerCoordinate = mapView.centerCoordinate
                    self.trackingViewModel.reverseGeocodeCenter(centerCoordinate:
 centerCoordinate, marker: marker)
                }
            }
        }
    }
}
```

2. ViewModel フォルダの下に AWSSignatureV4Delegate ファイルを追加します。このファイルは、マップをレンダリングするためのすべての MapView http リクエストで署名するために使用されます。

```
import MapLibre
import AmazonLocationiOSAuthSDK

class AWSSignatureV4Delegate : NSObject, MLNOfflineStorageDelegate {
    private let awsSigner: AWSSigner

    init(credentialsProvider: LocationCredentialsProvider) {
        self.awsSigner = DENY LIST ERROR , serviceName: "geo")
        super.init()
    }

    func offlineStorage(_ storage: MLNOfflineStorage, urlForResourceOf kind:
 MLNResourceKind, with url: URL) -> URL {
        if url.host?.contains("amazonaws.com") != true {
            return url
        }
        let signedURL = awsSigner.signURL(url: url, expires: .hours(1))

        return signedURL
    }
}
```

3. ビューフォルダにUserLocationView.swiftファイルを追加します。これにより、マップをユーザーの位置の中央に配置するボタンが追加されます。

```
import SwiftUI

struct UserLocationView: View {
    @ObservedObject var trackingViewModel: TrackingViewModel
    var body: some View {
            Button(action: {
                trackingViewModel.locateMe()
            }) {
                Image(systemName: "scope")
                    .resizable()
                    .frame(width: 24, height: 24)
                    .padding(5)
                    .background(Color.white)
                    .foregroundColor(.blue)
                    .clipShape(RoundedRectangle(cornerRadius: 8))
                    .shadow(color: Color.black.opacity(0.3), radius: 3, x: 0, y: 2)
            }
```

```
            .accessibility(identifier: "LocateMeButton")
            .padding(.trailing, 10)
            .padding(.bottom, 10)
            .frame(maxWidth: .infinity, alignment: .trailing)
    }
}
```

4. 次のコードで TrackingView.swift ファイルを追加します。

```
import SwiftUI

struct TrackingView: View {
    @ObservedObject var trackingViewModel: TrackingViewModel
    var body: some View {
        ZStack(alignment: .bottom) {
            MapView(trackingViewModel: trackingViewModel)
            VStack {
                UserLocationView(trackingViewModel: trackingViewModel)
            }
        }
        .onAppear() {
            if !trackingViewModel.identityPoolId.isEmpty {
                trackingViewModel.authWithCognito(identityPoolId:
 trackingViewModel.identityPoolId)
            }
        }
    }
}
```

これで、アプリケーションを構築できます。これを実行するには、Xcode でエミュレートするデバイスを設定するか、デバイスでアプリを使用する必要がある場合があります。このアプリを使用して、マップコントロールの動作を確認します。マップをドラッグしてパンし、ピンチしてズームできます。マップコントロールの仕組みを自分で変更して、アプリケーションのニーズに合わせてカスタマイズできます。

アプリケーションに検索を追加する

次に、リバースジオコーディング検索をアプリケーションに追加します。ここでは、ある場所にある項目を見つけます。iOS アプリの使用を簡素化するために、画面の中央を検索します。新しい場所を見つけるには、検索したい場所にマップを移動します。マップの中央にマーカーを配置して、検索している場所を示します。

1. リバースジオコーディング検索に関連する次のコードを TrackingViewModel`.swift` ファイルに追加します。

```
func reverseGeocodeCenter(centerCoordinate: CLLocationCoordinate2D, marker:
 MLNPointAnnotation) {
    let position = [NSNumber(value: centerCoordinate.longitude), NSNumber(value:
 centerCoordinate.latitude)]
    searchPositionAPI(position: position, marker: marker)
}

func searchPositionAPI(position: [Double], marker: MLNPointAnnotation) {
    if let amazonClient = authHelper.getLocationClient() {
        Task {
            let searchRequest = SearchPlaceIndexForPositionInput(indexName:
 indexName, language: "en" , maxResults: 10, position: position)
            let searchResponse = try? await amazonClient.searchPosition(indexName:
 indexName, input: searchRequest)
            DispatchQueue.main.async {
                self.centerLabel = searchResponse?.results?.first?.place?.label ??
 ""
                self.mlnMapView?.selectAnnotation(marker, animated: true,
 completionHandler: {})
            }
        }
    }
}
```

2. マップビューの中央ロケーションのアドレスを示す次のコードでTrackingView.swiftファイルを更新します。

```
import SwiftUI

struct TrackingView: View {
    @ObservedObject var trackingViewModel: TrackingViewModel
    var body: some View {
        ZStack(alignment: .bottom) {
            if trackingViewModel.mapSigningIntialised {
                MapView(trackingViewModel: trackingViewModel)
                VStack {
                    UserLocationView(trackingViewModel: trackingViewModel)
                    CenterAddressView(trackingViewModel: trackingViewModel)
                }
            }
```

```
            else {
                Text("Loading...")
            }
        }
        .onAppear() {
            if !trackingViewModel.identityPoolId.isEmpty {
                Task {
                    do {
                        try await trackingViewModel.authWithCognito(identityPoolId:
 trackingViewModel.identityPoolId)
                    }
                    catch {
                        print(error)
                    }
                }
            }
        }
    }
}
```

アプリケーションへの追跡の追加

アプリケーションの最後のステップは、アプリケーションに追跡機能を追加することです。この場合、アプリに追跡の開始、追跡の停止、トラッカーポイントの取得と表示を追加します。

1. プロジェクトに TrackingBottomView.swift ファイルを追加します。ユーザーの位置の追跡を開始および停止するボタンがあり、マップ上に追跡ポイントを表示します。

```
import SwiftUI

struct TrackingBottomView: View {
    @ObservedObject var trackingViewModel: TrackingViewModel
    var body: some View {
           Button(action: {
               Task {
                   if(trackingViewModel.trackingButtonText ==
 NSLocalizedString("StartTrackingLabel", comment: "")) {
                       trackingViewModel.startTracking()
                   } else {
                       trackingViewModel.stopTracking()
                   }
               }
```

```
            }) {
                HStack {
                    Spacer()
                    Text("Tracking")
                        .foregroundColor(trackingViewModel.trackingButtonColor)
                        .background(.white)
                        .cornerRadius(15.0)

                    Image(systemName: trackingViewModel.trackingButtonIcon)
                        .resizable()
                        .frame(width: 24, height: 24)
                        .padding(5)
                        .background(.white)
                        .foregroundColor(trackingViewModel.trackingButtonColor)

                }
            }
            .accessibility(identifier: "TrackingButton")
            .background(.white)
            .clipShape(RoundedRectangle(cornerRadius: 8))
            .padding(.trailing, 10)
            .padding(.bottom, 40)
            .frame(width: 130, alignment: .trailing)
            .shadow(color: Color.black.opacity(0.3), radius: 3, x: 0, y: 2)
        }
}
```

2. 次のコードでTrackingView.swiftファイルを更新する

```
import SwiftUI

struct TrackingView: View {
    @ObservedObject var trackingViewModel: TrackingViewModel
    var body: some View {
        ZStack(alignment: .bottom) {
            if trackingViewModel.mapSigningIntialised {
                MapView(trackingViewModel: trackingViewModel)
                VStack {
                    UserLocationView(trackingViewModel: trackingViewModel)
                    CenterAddressView(trackingViewModel: trackingViewModel)
                    TrackingBottomView(trackingViewModel: trackingViewModel)
                }
            }
            else {
```

```
                Text("Loading...")
            }
        }
        .onAppear() {
            if !trackingViewModel.identityPoolId.isEmpty {
                Task {
                    do {
                        try await trackingViewModel.authWithCognito(identityPoolId:
 trackingViewModel.identityPoolId)
                    }
                    catch {
                        print(error)
                    }
                }
            }
        }
    }
}
```

3. ファイルに次のコードを追加しますTrackingViewModel.swift。これらの関数は、追跡の開始と停止を担当します。また、ユーザーロケーションのアクセス許可が拒否されると、エラーアラートも表示されます。

4. フォアグラウンド追跡コピーを実装するには、次のコード例を貼り付けます。

```
func showLocationDeniedRationale() {
        alertTitle = NSLocalizedString("locationManagerAlertTitle", comment: "")
        alertMessage = NSLocalizedString("locationManagerAlertText", comment: "")
        showAlert = true
    }

    // Required in info.plist: Privacy - Location When In Use Usage Description
    func startTracking() {
        do {
            print("Tracking Started...")
            if(client == nil) {
                initializeClient()
            }
            try client.startTracking()
            DispatchQueue.main.async { [self] in
                self.trackingButtonText = NSLocalizedString("StopTrackingLabel",
 comment: "")
                self.trackingButtonColor = .red
                self.trackingButtonIcon = "pause.circle"
```

```
                trackingActive = true
            }
        } catch TrackingLocationError.permissionDenied {
            showLocationDeniedRationale()
        } catch {
            print("error in tracking")
        }
    }

    func stopTracking() {
        print("Tracking Stopped...")
        client.stopTracking()
        trackingButtonText = NSLocalizedString("StartTrackingLabel", comment: "")
        trackingButtonColor = .blue
        trackingButtonIcon = "play.circle"
        trackingActive = false
    }
```

> **Note**
>
> `startTracking` は、ユーザーの場所のアクセス許可を要求します。アプリケーションは、使用中または 1 回のみのアクセス許可を使用する必要があります。そうしないと、アプリケーションはアクセス許可拒否エラーをスローします。

追跡場所を取得して表示するには、次の手順に従います。

1. ユーザーのデバイスからロケーションを取得するには、開始日時と終了日時を指定する必要があります。1 回の呼び出しで最大 100 個の追跡場所が返されますが、100 個を超える追跡場所がある場合、`nextToken` 値が返されます。指定された開始時刻と終了時刻の追跡ポイントをロードするには、「nextToken」で後続のgetTrackerDevice「Location」呼び出しを呼び出す必要があります。

```
func getTrackingPoints(nextToken: String? = nil) async throws {
       guard trackingActive else {
           return
       }
       // Initialize startTime to 24 hours ago from the current date and time.
       let startTime: Date = Date().addingTimeInterval(-86400)
       var endTime: Date = Date()
       if lastGetTrackingTime != nil {
```

```
            endTime = lastGetTrackingTime!
        }
        let result = try await client?.getTrackerDeviceLocation(nextToken:
nextToken, startTime: startTime, endTime: endTime)
        if let trackingData = result {

            lastGetTrackingTime = Date()
            let devicePositions = trackingData.devicePositions

            let positions = devicePositions!.sorted { (pos1:
LocationClientTypes.DevicePosition, pos2: LocationClientTypes.DevicePosition) ->
Bool in
                guard let date1 = pos1.sampleTime,
                      let date2 = pos2.sampleTime else {
                    return false
                }
                return date1 < date2
            }

            let trackingPoints = positions.compactMap { position ->
CLLocationCoordinate2D? in
                guard let latitude = position.position!.last, let longitude =
position.position!.first else {
                    return nil
                }
                return CLLocationCoordinate2D(latitude: latitude, longitude:
longitude)
            }
            DispatchQueue.main.async {
                self.mapViewDelegate!.drawTrackingPoints( trackingPoints:
trackingPoints)
            }
            if let nextToken = trackingData.nextToken {
                try await getTrackingPoints(nextToken: nextToken)
            }
        }
    }
```

2. 次に、 MapView.swift ファイル内のコードを次のコードに置き換えます。

```
import SwiftUI
import MapLibre

struct MapView: UIViewRepresentable {
```

```
    var onMapViewAvailable: ((MLNMapView) -> Void)?
    var mlnMapView: MLNMapView?
    var trackingViewModel: TrackingViewModel

    func makeCoordinator() -> MapView.Coordinator {
        return Coordinator(self, trackingViewModel: trackingViewModel)
    }

    func makeUIView(context: Context) -> MLNMapView {
        let styleURL = URL(string: "https://maps.geo.
\(trackingViewModel.region).amazonaws.com/maps/v0/maps/
\(trackingViewModel.mapName)/style-descriptor")
        let mapView = MLNMapView(frame: .zero, styleURL: styleURL)
        mapView.autoresizingMask = [.flexibleWidth, .flexibleHeight]
        mapView.setZoomLevel(15, animated: true)
        mapView.showsUserLocation = true
        mapView.userTrackingMode = .follow
        context.coordinator.mlnMapView = mapView
        mapView.delegate = context.coordinator

        mapView.logoView.isHidden = true
        context.coordinator.addCenterMarker()

        onMapViewAvailable?(mapView)
        trackingViewModel.mlnMapView = mapView
        return mapView
    }

    func updateUIView(_ uiView: MLNMapView, context: Context) {
    }

    class Coordinator: NSObject, MLNMapViewDelegate, MapViewDelegate {
        var control: MapView
        var mlnMapView: MLNMapView?
        var trackingViewModel: TrackingViewModel
        var centerMarker: MLNPointAnnotation?

        public init(_ control: MapView, trackingViewModel: TrackingViewModel) {
            self.control = control
            self.trackingViewModel = trackingViewModel
            super.init()
            self.trackingViewModel.mapViewDelegate = self
        }
```

```
        func mapViewDidFinishRenderingMap(_ mapView: MLNMapView, fullyRendered:
Bool) {
            if(fullyRendered) {
                mapView.accessibilityIdentifier = "MapView"
                mapView.isAccessibilityElement = false
            }
        }

        func addCenterMarker() {
            guard let mlnMapView = mlnMapView else {
                return
            }

            let centerCoordinate = mlnMapView.centerCoordinate
            let marker = MLNPointAnnotation()
            marker.coordinate = centerCoordinate
            marker.accessibilityLabel = "CenterMarker"
            mlnMapView.addAnnotation(marker)
            centerMarker = marker

            trackingViewModel.reverseGeocodeCenter(centerCoordinate:
centerCoordinate, marker: marker)
        }

        func mapView(_ mapView: MLNMapView, regionDidChangeAnimated animated: Bool)
{
            if let marker = centerMarker {
                DispatchQueue.main.asyncAfter(deadline: .now() + 1.0) {
                    mapView.deselectAnnotation(marker, animated: false)
                    marker.coordinate = mapView.centerCoordinate
                    let centerCoordinate = mapView.centerCoordinate
                    self.trackingViewModel.reverseGeocodeCenter(centerCoordinate:
centerCoordinate, marker: marker)
                }
            }
        }

        func mapView(_ mapView: MLNMapView, viewFor annotation: MLNAnnotation) ->
MLNAnnotationView? {
            guard let pointAnnotation = annotation as? MLNPointAnnotation else {
                return nil
            }

            let reuseIdentifier: String
```

```
            var color: UIColor = .black
            if pointAnnotation.accessibilityLabel == "Tracking" {
                reuseIdentifier = "TrackingAnnotation"
                color = UIColor(red: 0.00784313725, green: 0.50588235294, blue:
0.58039215686, alpha: 1)
            } else if pointAnnotation.accessibilityLabel == "LocationChange" {
                reuseIdentifier = "LocationChange"
                color = .gray
            } else {
                reuseIdentifier = "DefaultAnnotationView"
            }

            var annotationView =
mapView.dequeueReusableAnnotationView(withIdentifier: reuseIdentifier)

            if annotationView == nil {
                if reuseIdentifier != "DefaultAnnotationView" {
                    annotationView = MLNAnnotationView(annotation: annotation,
reuseIdentifier: reuseIdentifier)
                    //If point annotation is an uploaded Tracking point the radius
is 20 and color is blue, otherwise radius is 10 and color is gray
                    let radius = pointAnnotation.accessibilityLabel == "Tracking" ?
20:10
                    annotationView?.frame = CGRect(x: 0, y: 0, width: radius,
height: radius)
                    annotationView?.backgroundColor = color
                    annotationView?.layer.cornerRadius = 10

                    if pointAnnotation.accessibilityLabel == "Tracking" {
                        annotationView?.layer.borderColor = UIColor.white.cgColor
                        annotationView?.layer.borderWidth = 2.0
                        annotationView?.layer.shadowColor = UIColor.black.cgColor
                        annotationView?.layer.shadowOffset = CGSize(width: 0,
height: 2)
                        annotationView?.layer.shadowRadius = 3
                        annotationView?.layer.shadowOpacity = 0.2
                        annotationView?.clipsToBounds = false
                    }
                }
                else {
                    return nil
                }
            }
```

```
            return annotationView
        }

        func mapView(_ mapView: MLNMapView, didUpdate userLocation:
MLNUserLocation?) {
            if (userLocation?.location) != nil {
                if trackingViewModel.trackingActive {
                    let point = MLNPointAnnotation()
                    point.coordinate = (userLocation?.location!.coordinate)!
                    point.accessibilityLabel = "LocationChange"
                    mapView.addAnnotation(point)
                    Task {
                        do {
                            try await trackingViewModel.getTrackingPoints()
                        }
                        catch {
                            print(error)
                        }
                    }
                }
            }
        }

        func checkIfTrackingAnnotationExists(on mapView: MLNMapView, at
coordinates: CLLocationCoordinate2D) -> Bool {
            let existingAnnotation = mapView.annotations?.first(where: { annotation
in
                guard let annotation = annotation as? MLNPointAnnotation else
{ return false }
                return annotation.coordinate.latitude == coordinates.latitude &&
                annotation.coordinate.longitude == coordinates.longitude &&
annotation.accessibilityLabel == "Tracking" })
            return existingAnnotation != nil
        }

        public func drawTrackingPoints(trackingPoints: [CLLocationCoordinate2D]?) {
            guard let mapView = mlnMapView, let newTrackingPoints =
trackingPoints, !newTrackingPoints.isEmpty else {
                return
            }

            let uniqueCoordinates = newTrackingPoints.filter { coordinate in
                !checkIfTrackingAnnotationExists(on: mapView, at: coordinate)
            }
```

```
            let points = uniqueCoordinates.map { coordinate -> MLNPointAnnotation
 in
                let point = MLNPointAnnotation()
                point.coordinate = coordinate
                point.accessibilityLabel = "Tracking"
                return point
            }
            mapView.addAnnotations(points)
        }
    }
}

protocol MapViewDelegate: AnyObject {
    func drawTrackingPoints(trackingPoints: [CLLocationCoordinate2D]?)
}
```

文字列値 をローカライズするには、次の手順を使用します。

1. という名前の新しいファイルを作成して追加しますLocalizable.xcstrings。
2. Localizable.xcstrings ファイルを右クリックし、ソースコード として開きます。
3. その内容を以下に置き換えます。

```
{
  "sourceLanguage" : "en",
  "strings" : {
    "Cancel" : {
      "extractionState" : "manual",
      "localizations" : {
        "en" : {
          "stringUnit" : {
            "state" : "translated",
            "value" : "Cancel"
          }
        }
      }
    },
    "Error" : {
      "extractionState" : "manual",
      "localizations" : {
        "en" : {
```

```
        "stringUnit" : {
          "state" : "translated",
          "value" : "Error"
        }
      }
    }
  },
  "Loading..." : {

  },
  "locationManagerAlertText" : {
    "extractionState" : "manual",
    "localizations" : {
      "en" : {
        "stringUnit" : {
          "state" : "translated",
          "value" : "Allow \\\"Quick Start App\\\" to use your location"
        }
      }
    }
  },
  "locationManagerAlertTitle" : {
    "extractionState" : "manual",
    "localizations" : {
      "en" : {
        "stringUnit" : {
          "state" : "translated",
          "value" : "We need your location to detect your location in map"
        }
      }
    }
  },
  "NotAllFieldsAreConfigured" : {
    "extractionState" : "manual",
    "localizations" : {
      "en" : {
        "stringUnit" : {
          "state" : "translated",
          "value" : "Not all the fields are configured"
        }
      }
    }
  },
  "OK" : {
```

```
      "extractionState" : "manual",
      "localizations" : {
        "en" : {
          "stringUnit" : {
            "state" : "translated",
            "value" : "OK"
          }
        }
      }
    },
    "StartTrackingLabel" : {
      "localizations" : {
        "en" : {
          "stringUnit" : {
            "state" : "translated",
            "value" : "Start Tracking"
          }
        }
      }
    },
    "StopTrackingLabel" : {
      "localizations" : {
        "en" : {
          "stringUnit" : {
            "state" : "translated",
            "value" : "Stop Tracking"
          }
        }
      }
    },
    "Tracking" : {

    }
  },
  "version" : "1.0"
}
```

4. ファイルを保存し、アプリを構築して実行し、機能をプレビューします。
5. ロケーション許可を付与し、追跡ボタンをタップします。アプリはユーザーロケーションのアップロードを開始し、Amazon Location トラッカーにアップロードします。また、ユーザーの位置変更、追跡ポイント、現在の住所がマップに表示されます。

クイックスタートアプリケーションが完了しました。このチュートリアルでは、以下の iOS アプリケーションを作成する方法を説明します。

- ユーザーが操作できるマップを作成します。
- マップビューを変更するユーザーに関連するいくつかのマップイベントを処理します。
- Amazon Location Service API を呼び出します。具体的には、Amazon Location の searchByPosition API を使用して、特定の場所でマップを検索します。

次のステップ

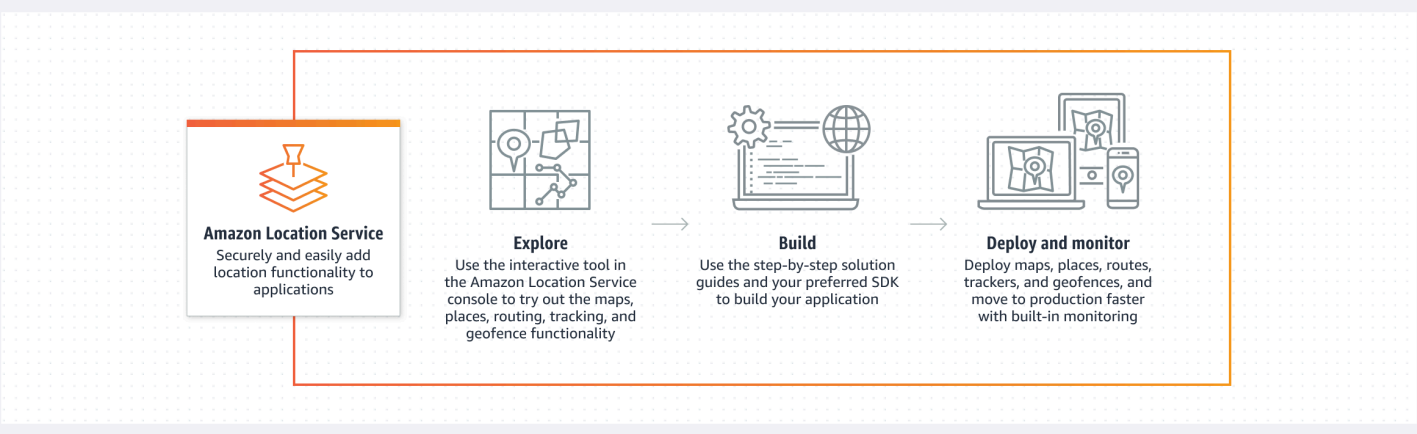
このアプリケーションのソースコードは、 で入手できますGitHub。

Amazon Location をさらに活用するには、以下のリソースをご覧ください。

- Amazon Location Service の概念をより深く知る
- Amazon Location の特徴や機能の使用方法に関する詳細情報はこちら
- Amazon Location を使用したコード例を見て、このサンプルを発展させて、より複雑なアプリケーションを構築する方法はこちら

Amazon Location Service の概念

Amazon Location Service を使用すると、アプリケーションに位置データを安全に追加できます。Amazon Location コンソールにある視覚的でインタラクティブなツールを使用して、機能の一部を試してみてください。探索ツールを使用すると、デフォルトマップを操作したり、関心のある地点を検索したり、対象地域の周囲にジオフェンスを描画したり、デバイスの位置情報をトラッカーに送信するシミュレーションを行ったりできます。



作成する準備ができたら、リソースを作成し、さまざまなマップスタイルやデータプロバイダーから選択します。その後、開発環境に合った SDK をインストールし、このガイドの手順に従って Amazon Location API を使用できます。さらに、Amazon CloudWatch と を使用してモニタリングを統合できます AWS CloudTrail。

このセクションのトピックでは、Amazon Location のコアコンセプトの概要を説明し、独自のアプリケーションで位置情報を使い始めるための準備を整えます。

トピック

- Amazon Location の概要
- マップ
- 場所検索
- ルート
- ジオフェンスとトラッカー
- Amazon Location Service を使用する一般的なユースケース
- データプロバイダーとは何ですか？
- Amazon Location のリージョンとエンドポイント
- Amazon Location Service のサービスクォータ

Amazon Location の概要

Amazon Location Service は、 AWS リソースを通じて口ケーションベースの機能やデータプロバイダーへのアクセスを提供します。Amazon Location には、必要な機能の種類に応じて 5 種類の AWS リソースが用意されています。さまざまなリソースを組み合わせて、完全な口ケーションベースのアプリケーションを作成します。Amazon Location コンソール、Amazon Location API、または SDK を使用して、これらのリソースを 1 つ以上作成できます。

各リソースは、使用する基盤となるデータプロバイダー (該当する場合) を定義し、そのタイプに関連する機能へのアクセスを提供します。

例：

- Amazon Location Service Maps では、マッププロバイダーからマップを選択して、モバイルアプリケーションまたはウェブアプリケーションで使用できます。
- Amazon Location Service Places では、関心のある地点の検索、部分テキストの入力、ジオコーディング、リバースジオコーディングを行うデータプロバイダーを選択できます。
- Amazon Location Service Routes では、データプロバイダーを選択し、 up-to-date 道路やライブ交通情報に基づいてルートを検索し、移動時間を見積もることができます。
- Amazon Location Service Geofences では、対象地域を仮想境界として定義できます。その後、その位置を基準にして位置を評価し、入退室イベントの通知を受け取ることができます。
- Amazon Location Service Trackers では、お使いのデバイスから位置情報の更新を受け取ります。トラッカーをジオフェンスコレクションにリンクすると、位置の更新がすべてジオフェンスと照合されて自動的に評価されます。

IAM ポリシーを使用して、Amazon Location リソースへのアクセスを管理および承認できます。また、リソースをリソースグループに整理して、リソースの数が増えるにつれてタスクを管理および自動化することもできます。 AWS リソースの管理の詳細については、「AWS Resource Groups とは」を参照してください。AWS Resource Groups のユーザーガイド。

位置は、全地球測位システム (GPS) サービスの標準座標参照系として一般的に使用されている世界測地系 (WGS 84) に従った緯度と経度の座標を使用して定義されます。

次のセクションでは、Amazon Location のコンポーネントがどのように機能するかについて説明します。

マップ

Amazon Location Service Map リソースを使用すると、マップの基になるベースマップデータにアクセスできます。Map リソースとマップレンダリングライブラリを使用して、インタラクティブマップをアプリケーションに追加します。アプリケーションの必要に応じて、マーカー (またはピン)、ルート、ポリゴンエリアなどの他の機能をマップに追加できます。

> **Note**
>
> マップリソースの実際の使用方法については、「アプリケーションで Amazon Location マップを使用する」を参照してください。

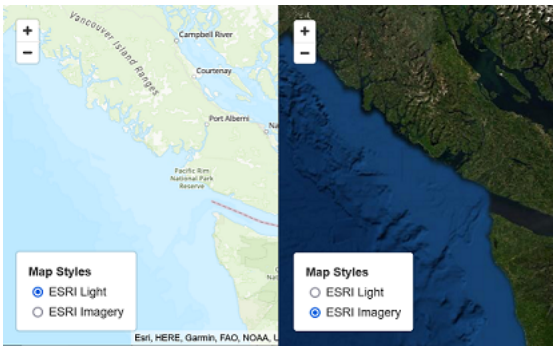
以下は、マップリソースを作成および使用する方法の概要です。

1. データプロバイダーからマップスタイルを選択して、 AWS アカウントにマップリソースを作成します。
2. その後、開発環境とアプリケーションに合った SDK を選択してインストールできます。使用可能なオプションの詳細については、Amazon Location へのアクセスに関するトピックを参照してください。
3. アプリケーションにマップを表示するには、マップリソースを Amplify、 MapLibreTangram などのレンダリングライブラリと組み合わせます。詳細については、このガイドの「マップの使用」を参照してください。
4. その後、Amazon CloudWatch や AWS CloudTrail Amazon Location などの のサービスを使用してモニタリングを統合できます。詳細については、「Amazon による Amazon Location Service のモニタリング CloudWatch」および「 での口グ記録とモニタリング」を参照してください。

マップスタイル

マップリソースを作成するときは、そのリソースのマップスタイルを選択する必要があります。マップスタイルは、レンダリングされたマップの外観を定義します。例えば、次の画像は、同じデータ

プロバイダーを Amazon Location の異なるマップリソースの 2 つの異なるスタイルで表現したものです。1 つのスタイルは、マップ内のベクターデータに基づく一般的な道路スタイルです。もう 1 つは、衛星画像を示すラスターデータです。マップを拡大または縮小するとスタイルが変わる場合がありますが、通常、スタイルには一貫したテーマがあります。スタイル情報をマップレンダリングライブラリに渡す前に、スタイル情報の一部または全部をオーバーライドできます。



政治的見解

Amazon Location Service 特定のマップスタイルでは、追加の政治的見解がサポートされています。

> **Note**
>
> 政治的見解は、適用法に準拠して使用する必要があります。これには、Amazon Location Service を通じてアクセスする地図、画像、その他のデータ、およびサードパーティのコンテンツを利用できる国または地域のマッピングに関する法律が含まれます。

以下のマップスタイルは、インド (IND) の政治的見解に対応しています。

- Esri マップスタイル:
  - Esri Navigation
  - Esri Light
  - Esri Street Map
  - Esri Dark Gray Canvas
  - Esri Light Gray Canvas
- Open Data マップスタイル:
  - Open Data Standard Light
  - Open Data Standard Dark
  - Open Data Visualization Light

- Open Data Visualization Dark

Amazon Location Service コンソールでは、表示されるスタイルをフィルタリングして、インドの政治的見解を支持するスタイルのみを表示できます。

カスタムレイヤー

カスタムレイヤーは、マップスタイルに対して有効化できる追加のレイヤーです。現在、VectorEsriNavigation マップスタイルのみが`POI`カスタムレイヤーをサポートしています。

`POI` カスタムレイヤーを有効にすると、ショップ、サービス、レストラン、アトラクション、その他の名所など、さまざまな場所の豊富なセットがマップに追加されます。デフォルトでは、カスタムレイヤーは `unset` です。詳細については、「Location API リファレンスMapConfiguration」の「」を参照してください。

マップレンダリング

アプリケーションでマップをレンダリングするには、通常、マップレンダリングライブラリを使用します。ライブラリにはよく使われるオプションがいくつかあります。

- MapLibre - MapLibre は、インタラクティブマップをレンダリングするためのオープンソースライブラリであり、Amazon Location Service からマップをレンダリングするための推奨方法です。 には、データソース (Amazon Location マップリソースなど) からラスターデータとベクトルデータをレンダリングする機能 MapLibre が含まれています。を拡張 MapLibre して、マップ上に独自のデータを描画できます。
- Amplify — Amplify は、ウェブ、iOS、Android などのアプリケーションを構築するためのオープンソースフレームワークです。アプリケーションで Amplify を使用している場合は、Amazon Location 機能を含むように拡張できます。Amplify には、レンダリングマップなどの Amazon Location ベースのアプリケーションを作成するためのライブラリが含まれています。Amplify は MapLibre を使用してマップをレンダリングしますが、Amazon Location Service に固有の追加機能を提供して、より効率的に使用できるようにします。また、検索やその他の機能も追加します。
- Tangram – Tangram は、 と同様にインタラクティブマップをレンダリングする代替オープンソースライブラリです MapLibre。

マップレンダリングライブラリは、ランタイムに Amazon Location Service からデータを取得し、選択したマップリソースに基づいてマップデータをレンダリングします。マップリソースは、使用するデータプロバイダーとマップスタイルを定義します。

以下の画像は、Amazon Location Service でマップリソースをマップレンダリングライブラリとともに使用して最終的なマップを作成する方法を示しています。

1. Amazon Location Service でマップリソースを作成するには、 AWS Management Console または を使用します AWS CLI。これにより、使用するデータプロバイダーとマップスタイルが定義されます。
2. アプリケーションには、マップレンダリングライブラリが含まれています。マップレンダリングライブラリには、使用するマップリソースの名前を指定します。マップレンダリングライブラリは、Amazon Location からそのマップリソースのデータとスタイル情報を取得し、マップを画面上にレンダリングします。

マップの用語

マップリソース

選択したプロバイダーのマップデータにアクセスできます。マップリソースを使用して、マップデータと、マップ上でのフィーチャのレンダリング方法を指定するスタイル記述子を含むマップタイルを取得します。

ベースマップ

マップに地理情報を提供し、ベクタータイルレイヤーとして格納します。タイルレイヤーには、道路名、建物、土地利用などの地理的コンテキストが含まれており、視覚的に参照できます。

ベクトル

ベクターデータは、ポイント、ライン、ポリゴンで構成されるシェイプデータです。道路、場所、地域を地図上に保存して表示するためによく使用されます。ベクターシェイプは、マップ上のマーカーのアイコンとしても使用できます。

Raster

ラスターデータはグリッドで構成された画像データで、通常は色で構成されています。地形、衛星画像、ヒートマップなど、連続したデータをマップ上に格納して表示するためによく使用されます。ラスター画像は画像やアイコンとしても使用できます。

Map Style

ベクターデータには、データのレイヤーを描画して最終的なマップを作成する方法に関する情報が本質的に含まれていません。マップスタイルは、データの色やその他のスタイル情報を定義して、レンダリング時の外観を定義します。マップリソースには、マップのスタイル情報が含まれます。

Amazon Location Service は、Mapbox GL スタイル仕様に準拠したスタイルを提供します。

Vector タイル

ベクターシェイプを使用してマップデータを格納するタイル形式。このデータにより、最適なパフォーマンスを実現するためにファイルサイズを小さく抑えながら、表示解像度に合わせて調整し、さまざまな方法でフィーチャを選択的にレンダリングできるマップが作成されます。

サポートされているベクターファイル形式: Mapbox Vector Tiles (MVT)。

Glyph ファイル

エンコードされた Unicode 文字を含むバイナリファイル。マップレンダラーがラベルを表示するために使用します。

Sprite ファイル

JSON ファイル内の場所の説明を含む小さなラスター画像を含む Portable Network Graphic (PNG) 画像ファイル。マップレンダラーがマップ上にアイコンやテクスチャをレンダリングするために使用されます。

場所検索

Amazon Location Service の主な機能は、地理位置情報を検索する機能です。Amazon Location は、プレースインデックスリソースを通じてこの機能を提供します。

> **Note**
>
> プレースインデックスリソースを使用して実際に検索する方法の詳細については、「Amazon Location を使用して場所と位置情報を検索する」を参照してください。

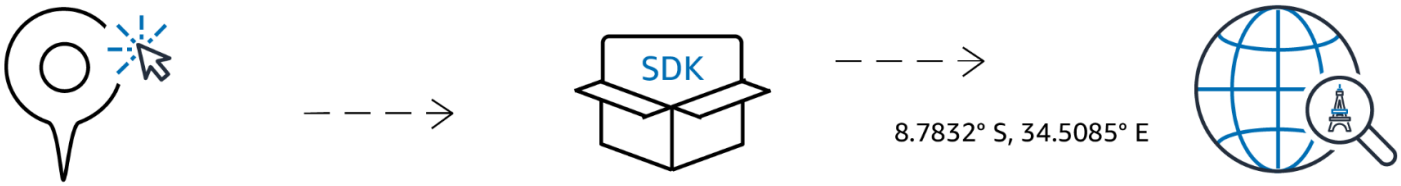
Place Index API を使用して以下を検索できます。

- レストランやランドマークなど、興味のあるポイント。名前で検索したり、検索したい場所を指定したりすると、関連性の高い順に並んだオプションのリストが表示されます。
- 住所。その住所の緯度と経度が表示されます。これはジオコーディングと呼ばれます。
- 緯度と経度の位置で、関連する住所やその場所に関するその他の情報を受け取ります。これはリバースジオコーディングと呼ばれます。
- 部分的またはスペルが間違っている自由形式のテキストクエリ (通常はユーザーが入力したもの)。これはオートコンプリート、オートサジェスト、またはファジーマッチングと呼ばれます。

Place Index には、検索に使用するデータプロバイダーが含まれます。

> **Note**
>
> 地図データやその他の位置情報 (正確な位置情報を含む) は、データプロバイダーによって異なる場合があります。ベストプラクティスとして、Place Index、マップ、その他の Amazon Location リソースには同じデータプロバイダを使用してください。例えば、Place Index によって返される場所が、マップリソースによって提供される同じ場所の位置と一致しない場合、マップ上の間違った場所と思われる場所にマーカーを配置できます。

以下は、プレースインデックスリソースを作成および使用する方法です。



1. まず、データプロバイダーを選択して、 AWS アカウントに場所インデックスリソースを作成します。

2. その後、開発環境とアプリケーションに合った SDK を選択してインストールできます。使用可能なオプションの詳細については、Amazon Location へのアクセスに関するトピックを参照してください。
3. Amazon Location Places API を使い始めましょう。詳細については、場所検索の使用に関するトピックを参照してください。
4. その後、Amazon CloudWatch や などのサービスを使用してモニタリングを統合できます AWS CloudTrail。詳細については、「the section called “によるモニタリング CloudWatch”」および「the section called “Amazon Location CloudTrail での の使用”」を参照してください。

ジオコーディングの概念

Amazon Location Place Index には、`SearchPlaceIndexForText`検索するテキストを指定できるアクションがあります。例えば、次のように検索できます。

- 場所 — **`Paris`** を検索すると、フランスの都市の位置が返される場合があります。
- 企業 — **`coffee shop`** を検索すると、名前や所在地を含むコーヒーショップのリストが返される場合があります。検索する場所や境界ボックスを指定して、結果をより関連性の高いものにすることもできます。この場合、ワシントン州シアトルのダウンタウンに店舗を指定すると、その地域のコーヒーショップが返されます。
- 住所 — **`1600 Pennsylvania Ave, Washington D.C.`** を検索すると、米国のホワイトハウス (その住所) の場所が返されます。

この方法でテキストを検索することを一般に「ジオコーディング」と呼び、住所または場所の地理的位置を検索します。

Amazon Location Service には、`SearchPlaceIndexForPosition` という リバースジオコーディング アクションも用意されています。これは地理的位置を取得し、住所、勤務先、またはその場所にあるものに関するその他の情報を返します。

検索結果

Amazon Location Service で検索リクエストが成功すると、1 つ以上の結果が返されます。各結果には、結果の名前または説明であるラベルが含まれます。例えば、**`coffee shop`** を検索すると、「Hometown Cafe」というコーヒーショップが見つかったというラベル `Hometown Cafe` の付いた結果が返される場合があります。検索結果には通常、構造化された住所 (住所番号、ユニット、番

地、郵便番号などのプロパティを含む) も含まれます。データプロバイダーによっては、国やタイムゾーンなどの他のメタデータも含まれる場合があります。

会社名やカテゴリ (**coffee shop** など) で検索する場合、返されたすべての結果を地図上に表示したい場合があります。住所検索では、最初の結果だけを自動的に使用したい場合があります。関連性については、次のトピックを参照してください。

複数の結果と関連性

テキストで検索する場合、Amazon Location Service は複数の結果を見つけることがよくあります。例えば、**Paris** を検索すると、フランスの都市だけでなく、テキサス州の都市も返される場合があります。結果は、データプロバイダーによって決定された関連性で並び替えられます。

> **Note**
>
> 結果は、すべてのプロバイダーから関連性の高い順に返されます。データプロバイダーとして Esri または Grab を選択した場合、結果には関連性の値が含まれます。これを使用して 1 つのリクエストの結果間の相対的な関連性を把握できます。

国名や検索する場所などの追加情報を指定すると、結果の順序が変わったり、結果の数が減ったり、返される結果セットが変更されたりする可能性があります。例えば、テキサスの場所を指定して **Paris** を検索すると、`Paris, Texas` よりも `Paris, France` が最初の結果として返される可能性が高くなります。

インタラクティブなアプリケーションでは、関連性を利用して上位の結果を受け入れるかどうかを決定したり、返された複数の結果を区別するようユーザーに求めたりできます。最初の結果の関連性が高い場合は、それを正解として受け入れるだけでもかまいません。関連性の高い結果が複数ある場合や、関連性の高い結果がない場合は、結果を一覧表示して、ユーザーが最適な結果を選択できるようにするとよいでしょう。

結果に対処する

同じ `SearchPlaceIndexForText` アクションを使用して Amazon Location Service で住所を検索できます。入力する情報が多いほど、返される住所は指定した住所と一致する可能性が高くなります。例えば、**123 Main St** は、**123 Main St, Anytown, California, 90210** より正しい結果が見つかる可能性は低くなります。

住所には、番地、通り、都市、地域、郵便番号などの複数の属性があります。これらの属性を使用して、できるだけ多くの要素に一致する住所を Place Index から検索します。見つかる属性が多いほど、その一致の関連性が高いと見なされ、返される可能性が高くなります。

> **Note**
>
> 住所結果との関連性は、結果が入力とどの程度一致するかによって決まります。これは、一致した属性の数だけでなく、結果が入力とどの程度一致するかによっても異なります。例えば、**`123 Main St`** という入力は、`Maine St` が唯一の結果である場合よりも、データ内に `Main St` が見つかった場合の関連性が高くなります。`Maine St` は引き続き返されますが、関連性の値が低くなる可能性があります。

検索結果には、住所全体のラベル (`123 Main St, Anytown, California, 90210`) だけでなく、返された住所の個々の構造化属性も含まれます。これは、例えばデータベースの住所フィールドに値を入力したり、結果を調べて見つかった場所の都市、地域、または郵便番号を検索したりできるので便利です。

Interpolation

Place Index データ内の住所には、完全に一致する住所が含まれています。例えば、次の図に示すように、通り `9th street` があり、1 つのブロックに 2 つの家 `220` と `240` があるとします。

データプロバイダーは、これら 2 つの既知の住所を使用して位置情報データを作成します。この 2 つの住所を検索すると、見つかります。データプロバイダーが地図データを作成した後に、最初の 2 つの住所の間に新しい家が追加されたとしましょう。この新しい家には住所 230 が割り当てられます。**230 S 9th St** を検索しても、データプロバイダーは引き続き結果を見つけます。既知の住所を使用する代わりに、既知の住所を補間し、それらの住所から新しい住所の位置を推定します。この場合、230 は 220 と 240 の中間 (通りの同じ側) にあると仮定し、それに基づいておおよその位置を返します。

> **Note**
>
> データプロバイダーは、位置情報データを定期的に新しい住所で更新します。この場合、230 S 9th St はデータプロバイダーデータに追加されますが、通常、新しい住所が作成されたが、まだデータに追加されていない期間があります。

この場合、データプロバイダーは、新しい住所がまだデータに含まれていないため、その新しい住所が世界中に存在するかどうかはわかりませんが、保有している情報からできる限り最善の回答を提供します。この結果は補間と呼ばれ、データプロバイダーが結果で返すことができます。`interpolated` が `false` を返した場合、それは既知のアドレスです。`true` が返された場合、おおよその住所です。返されない場合、データプロバイダーは結果が補間によるものかどうかについての情報を提供していないことになります。

> **Important**
>
> データプロバイダーは、まったく存在しない住所の補間結果を返すこともあります。例えば、ユーザーが **232 S 9th St** を入力すると、プロバイダーは存在しない住所を検索し、230 に近い 240 側の位置を返します。補間された住所は正しい場所にたどり着くのに役立ちますが、既知の住所ではないことを覚えておくとよいでしょう。

ジオコード結果の保存

プレースインデックスリソースを作成するときは、[データストレージオプション] (API で `IntendedUse` と呼ばれる) を指定する必要があります。これは、単独使用 または 保存結果のいずれかに設定できます。これは結果の使用目的について尋ねています。結果を (キャッシュ目的であっても) 保存する場合は、単独使用 オプションではなく、保存オプションを選択する必要があります。

> **Note**
>
> 保存オプション (「はい」のラベルが付いている場合、結果はコンソールに保存されるか、または `CreatePlaceIndex` API `storage` で選択) を選択した場合、Amazon Location Service は結果を自動的に保存しません。これは、結果を保存する予定であることを示しています。

Amazon Location Service へのクエリ結果をどのように使用するかを検討する際には、適用される AWS サービス条件 を常に把握しておく必要があります。

場所の用語

プレースインデックスリソース

検索クエリをサポートするデータソースを選択できます。例えば、対象とする地点、住所、または座標を検索できます。検索クエリがプレースインデックスリソースに送信されると、リソースの設定済みデータソースを使用して処理されます。

ジオコーディング

ジオコーディングは、テキスト入力を受け取り、それを Place Index で検索して、位置とともに結果を返すプロセスです。

リバースジオコーディング

リバースジオコーディングは、位置を取得し、その位置に関する情報 (その位置の住所、都市、事業所など) を Place Index から返すプロセスです。

Relevance

関連性とは、結果が入力とどの程度一致するかです。これは正確さの尺度ではありません。

Interpolation

補間とは、既知の住所位置をガイドポイントとして使用して、未知の住所を見つけるプロセスです。

ISO 3166 国コード

Amazon Location Service Places では、国際標準化機構 (ISO) 3166 国コードを使用して国または地域を参照します。

特定の国または地域のコードを検索するには、ISO Online Browsing Platform を使用してください。

ルート

このセクションでは、Amazon Location Service を使用したルーティングに関する概念の概要を提供します。

> **Note**
>
> 実際にルートリソースを使用する方法については、「Amazon Location Service によるルートの計算」を参照してください。

ルート計算リソース

ルート計算リソースを使用すると、選択したデータプロバイダーからの up-to-date 道路網とライブトラフィック情報に基づいてルートを検索し、移動時間を見積もることができます。

Routes API を使用すると、任意の 2 つの場所間のルートの移動時間、距離、ジオメトリをアプリケーションでリクエストできる機能を構築できます。また、Routes API を使用して、複数の出発地と目的地間の移動時間と距離を 1 回のリクエストでリクエストし、マトリックスを計算することもできます。

以下に示しているのは、ルート計算リソースを作成および使用する方法です。

1. まず、データプロバイダーを選択して、 AWS アカウントにルート計算リソースを作成します。
2. その後、開発環境とアプリケーションに合った SDK を選択してインストールできます。

3. Amazon Location Routes API の使用を開始します。ルーティング API の使用方法については、「Amazon Location Service によるルートの計算」を参照してください。
4. その後、Amazon CloudWatch や などのサービスを使用してモニタリングを統合できます AWS CloudTrail。詳細については、「Amazon による Amazon Location Service のモニタリング CloudWatch」および「 でのログ記録とモニタリング」を参照してください。

ルートの計算

Amazon Location のルート計算リソースには、2 つの地理的位置 (出発地と目的地) 間のルートを作成するために使用できる `CalculateRoute` と呼ばれるアクションを提供します。計算されたルートには、マップ上にルートを描画するためのジオメトリと、ルートの全体の時間と距離が含まれます。

ウェイポイントを使用する

ルートリクエストを作成するときに、ルートにウェイポイントを追加できます。これらは出発地と目的地の間にあるポイントで、ルート上のストップとして機能します。ルートは指定された各ウェイポイントを経由して計算されます。リクエスト内の 1 つの地点から次の地点までのルートは、`Leg` と呼ばれます。各区間には、ルートのその部分の距離、時間、ジオメトリが含まれます。

> **Note**
>
> ウェイポイントは、リクエストで指定された順序でルーティングされます。最短経路の順序は変わりません。最短経路を見つける方法については、ルートの準備セクションを参照してください。

ルートを計算するには、1 回のリクエストに最大 25 のウェイポイントを含めることができます。

交通量と出発時刻

Amazon Location Service、ルートを計算する際にトラフィックを考慮します。考慮されるトラフィックは、指定した時間に基づいています。今すぐ出発するように指定することも、特定の出発時間を指定することもできます。その場合、指定した時刻の交通状況に合わせて調整され、ルートの結果に影響します。

> **Note**
>
> 出発時刻とルート応答時間を使用して到着時刻を計算し、例えばドライバーの到着を推定できます。

Amazon Location に交通量を考慮させたくない場合は、出発時刻を指定したり、今すぐ出発を指定したりしないでください。これにより、そのルートに最適な交通状況を想定したルートが計算されます。

移動モードのオプション

Amazon Location Service を使用してルートを計算するときに、移動モードを設定できます。デフォルトの移動モードは自動車ですが、「トラック」と「徒歩」のどちらかを交互に選択できます。

自動車モードまたはトラックモードのいずれかを指定すると、追加のオプションも指定できます。

自動車モード では、有料道路やフェリーを避けたいように指定できます。これにより、フェリーや有料道路を避けようとしますが、目的地までの唯一の方法である場合は、フェリーや有料道路に沿ってルーティングします。

トラックモード では、フェリーや有料道路を避けることもできますが、トラックのサイズと重量を指定して、トラックが通らないルートを避けることもできます。

ルートの準備

Amazon Location Service を利用すれば、ルートの計画および改善したソフトウェアへの入力ができます。出発地と目的地間のルートについて、所要時間や移動距離などのルートの計算が作成できます。これをルートマトリックスの作成と呼びます。

> **Note**
>
> ルートプランニングと最適化ソフトウェアで解決できるシナリオはさまざまです。例えば、プランニングソフトウェアは、ポイント間の時間と距離のセットを使用して、各ポイントで停止する最短経路を計算し、1 人のドライバーに効率的なルートを提供できます。また、計画ソフトウェアを使用して複数のトラックに停車地を分割して車両全体の効率を高めたり、各顧客が必要な時間内に訪問できるようにしたりすることもできます。Amazon Location はルーティング機能を効率的に提供し、計画ソフトウェアがタスクを完了できるようにします。

例えば、出発地の A と B、目的地の X と Y を入力すると、Amazon Location Service は A から X、A から Y、B から X、B から Y までのルートの移動時間と移動距離を計算してルートを出します。

1 つのルートを計算する場合と同様に、さまざまな交通手段、回避手段、交通状況を使用してルートを計算できます。例えば、車両は長さが 35 フィートのトラックであると入力すると、それらの規制に基づいて移動時間と移動距離を計算してルートを決めます。ルートマトリックスの計算にウェイポイントを含めることはできません。

返された結果 (そして計算されたルート数) は、出発地点の数を目的地の数の掛け算の結果です。料金は、サービスへのリクエストごとではなく、計算されたルートごとに請求されるため、10 か所出発地と 10 か所の目的地の掛け算の結果、つまり、ルートマトリックスは 100 ルートとして請求されます。

ルート用語

ルート計算リソース

選択したデータプロバイダーから取得したトラフィックと道路ネットワークデータを使用して、移動時間、距離を見積もり、地図上のルートをプロットできる AWS リソース。

ルート計算ツールのリソースを使用して、さまざまな交通手段、迂回路、交通状況のルートを計算します。

ルート

ルートには、出発位置、ウェイポイントの位置、目的地の位置からパスに沿って移動するときに使用される詳細が含まれています。

ルートの詳細には次のようなものがあります。

- ある位置から別の位置までの距離。
- ある位置から次の位置に移動するのにかかる時間。
- ルートのパスを表す LineString ジオメトリ。

ルートの詳細については、「Amazon Location Service Routes API リファレンス」の「 CalculateRoute オペレーションのレスポンス構文」を参照してください。

ルートマトリックス

一連の出発地点から一連の目的地までのルートのリスト。ルートプランニングや最適化ソフトウェアへの入力として役立ちます。

ルートマトリックスの計算の詳細については、「Amazon Location Service Routes API リファレンス」の「 CalculateRouteMatrix オペレーションの構文」を参照してください。

`LineString` ジオメトリ

Amazon Location ルートは 1 つ以上の区間 (ルート全体の中のあるウェイポイントから別のウェイポイントへのルート) で構成されます。各区間のジオメトリは、`LineString` で表されるポリラインです。`LineString` は順序付けられた位置の配列で、これを使用してマップ上にルートを描くことができます。

以下は 3 つの点を持つ `LineString` の例です。

```
[
    [-122.7565,49.0021],
    [-122.3394,47.6159],
    [-122.1082,45.8371]
]
```

Waypoint

ウェイポイントは、出発地と目的地の間のルート上のストップとして機能する中間位置です。ルート上のストップオーバー順序は、リクエストで指定したウェイポイント位置の順序に従います。

レッグ

片足とは、ある位置から別の位置への移動です。ポジションが道路上にない場合は、一番近い道路に移動します。ルート内の区間数は、位置の合計数より 1 つ少なくなります。

ウェイポイントのないルートは、出発地点から目的地までの 1 つの区間で構成されます。ウェイポイントが 1 つのルートは、出発地点からウェイポイントまで、そしてウェイポイントから目的地までの 2 つの区間で構成されます。

ステップ

ステップは区間のサブセクションです。各ステップには、そのレッグ内のそのステップの概要情報が表示されます。

ジオフェンスとトラッカー

このセクションでは、Amazon Location Service のジオフェンスとトラッカーの使用に関する概念の概要を説明します。ジオフェンスはポリゴンの境界線で、デバイスや位置がエリアに出入りしたとき

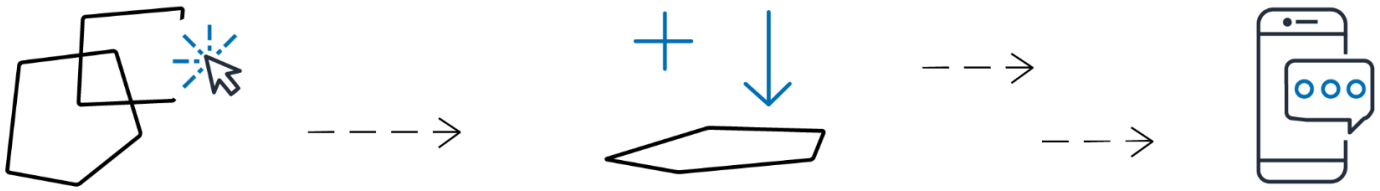
に通知を受けることができます。トラッカーリソースは、デバイスが移動するたびにその位置を保存および更新するために使用されます。

> **Note**
>
> ジオフェンスとトラッカーを実際に使用する方法の詳細については、「Amazon Location を使用して対象エリアをジオフェンシングする」を参照してください。

ジオフェンス

ジオフェンスコレクションリソースを使用すると、ジオフェンス (マップ上の仮想境界) を保存および管理できます。ジオフェンスコレクションリソースと照らし合わせて位置を評価し、位置の更新がジオフェンスコレクション内のいずれかのジオフェンスの境界を越えたときに通知を受け取ることができます。

以下に示しているのは、ジオフェンスコレクションリソースを作成および使用する方法です。



1. AWS アカウントにジオフェンスコレクションリソースを作成します。
2. そのコレクションにジオフェンスを追加します。Amazon Location コンソールのジオフェンスアップロードツールを使用するか、Amazon Location Geofences API を使用して追加できます。利用可能なオプションの詳細については、Amazon Location へのアクセス を参照してください。

   ジオフェンスはポリゴンまたは円で定義できます。ポリゴンを使用すると、デバイスが特定のエリアにいつ侵入したかを調べることができます。円を使用して、デバイスがポイントから特定の距離 (半径) 以内に入ったことを検出します。
3. すべてのジオフェンスに対して位置の評価を開始できます。ロケーションの更新が 1 つ以上のジオフェンスの境界を超えると、ジオフェンスコレクションリソースは Amazon で次のジオフェンスイベントタイプのいずれかを出力します EventBridge。
   - ENTER — 位置情報の更新がその境界を越えるジオフェンスを入力すると、ジオフェンスごとに 1 つのイベントが生成されます。

- EXIT — 位置情報の更新が終了して境界を越えるジオフェンスごとに 1 つのイベントが生成されます。

詳細については、「the section called “でのイベントへの対応 EventBridge”」を参照してください。Amazon CloudWatch や などのサービスを使用してモニタリングを統合することもできます AWS CloudTrail。詳細については、「the section called “によるモニタリング CloudWatch”」および「the section called “Amazon Location CloudTrail での の使用”」を参照してください。

例えば、大量のトラックを追跡していて、いずれかの倉庫の特定のエリアにトラックが入荷したときに通知を受け取りたい場合などです。各倉庫の周辺にジオフェンスを作成できます。その後、トラックから最新の位置情報が送られてきたら、Amazon Location Service を使用してそれらの位置を評価し、トラックがジオフェンスエリアに入った (または出た) かどうかを確認できます。

> **Note**
>
> 料金は、評価対象のジオフェンスコレクションの数によって決まります。請求額は、各コレクション内のジオフェンスの数に影響されません。各ジオフェンスコレクションには最大 50,000 のジオフェンスを含めることができるため、ジオフェンスの評価コストを削減するために、ジオフェンスをできるだけ少ないコレクションにまとめることをお勧めします。生成されるイベントには、コレクション内の個々のジオフェンスの ID とコレクションの ID が含まれます。

ジオフェンスイベント

監視している位置の位置は `DeviceId` と呼ばれる ID で参照されます (位置はデバイス位置と呼ばれます)。評価対象のデバイス位置のリストをジオフェンスコレクションリソースに直接送信することも、トラッカーを使用することもできます。トラッカーの詳細については、次のセクションを参照してください。

(Amazon 経由で EventBridge) イベントは、デバイスがジオフェンスに出入りする場合にのみ受信され、位置が変わるたびに受信されません。つまり、通常はデバイスの位置が更新されるたびに、イベントを受信して対応しなければならない頻度がはるかに少なくなります。

> **Note**
>
> 特定 `DeviceID` のデバイスの最初の位置評価では、そのデバイスは以前はどのジオフェンスにも存在していなかったと仮定します。そのため、最初の更新では、コレクション内のジオ

フェンス内にある場合はENTER イベントが生成され、そうでない場合はイベントが生成されません。

デバイスがジオフェンスに出入りしたかどうかを計算するために、Amazon Location Service はデバイスの以前の位置状態を保持する必要があります。この位置状態は 30 日間保存されます。デバイスの更新がない状態が 30 日経過すると、新しい位置情報の更新が最初の位置更新として扱われます。

トラッカー

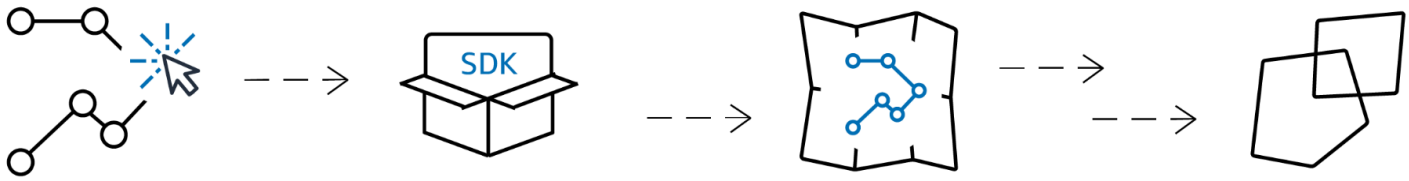
トラッカーは、複数のデバイスの位置更新情報を保存します。トラッカーを使用して、デバイスの現在位置や位置履歴を照会できます。更新情報を保存しますが、保存する前に位置をフィルタリングすることで、ストレージスペースを減らし、視覚的なノイズを減らします。

トラッカーリソースに保存される各位置更新には、位置精度の測定値と、保存したい位置またはデバイスに関する最大 3 つのフィールドのメタデータを含めることができます。メタデータはキーと値がペアとして保存されており、速度、方向、タイヤ空気圧、エンジン温度などの情報を保存できます。

> **Note**
>
> トラッカーストレージは、 AWS 所有キーで自動的に暗号化されます。管理する KMS キーを使用して別の暗号化レイヤーを追加して、自分だけがデータにアクセスできるようにすることができます。詳細については、「Amazon Location Service の保管中のデータ暗号化」を参照してください。

トラッカーの位置のフィルタリングと保存はそれだけでも便利ですが、トラッカーはジオフェンスと組み合わせると特に便利です。トラッカーは 1 つ以上のジオフェンスコレクションリソースにリンクでき、位置の更新はそれらのコレクション内のジオフェンスと照合して自動的に評価されます。フィルタリングを適切に使用すれば、ジオフェンスの評価コストも大幅に削減できます。

次の図は、トラッカーリソースを作成および使用する方法です。



1. まず、 AWS アカウントにトラッカーリソースを作成します。
2. 次に、位置情報の更新をトラッカーリソースに送信する方法を決定します。AWS SDK を使用して追跡機能をモバイルアプリケーションに統合します。または、MQTT を使用して追跡する step-by-step 指示に従って MQTT を使用することもできます。
3. トラッカーリソースを使用してロケーション履歴を記録し、地図上に視覚化できるようになりました。
4. トラッカーリソースを 1 つ以上のジオフェンスコレクションにリンクして、トラッカーリソースに送信されるすべての位置更新が、リンクされているすべてのジオフェンスコレクション内のすべてのジオフェンスに対して自動的に評価されるようにすることもできます。Amazon Location コンソールのトラッカーリソース詳細ページでリソースをリンクするか、Amazon Location Trackers API を使用してリソースをリンクできます。
5. その後、Amazon CloudWatch や などのサービスを使用してモニタリングを統合できます AWS CloudTrail。詳細については、「the section called “によるモニタリング CloudWatch”」および「the section called “Amazon Location CloudTrail での の使用”」を参照してください。

ジオフェンスでのトラッカーの使用

トラッカーをジオフェンスと組み合わせると、さらに機能が追加されます。Amazon Location コンソールまたは API を使用してトラッカーをジオフェンスコレクションに関連付けて、トラッカーロケーションを自動的に評価します。トラッカーが更新されたロケーションを受信するたびに、そのロケーションはコレクション内の各ジオフェンスに対して評価され、適切な ENTER イベントと EXITイベントが Amazon で生成されます EventBridge。トラッカーにフィルタリングを適用することもでき、フィルタリングによっては、意味のある位置の更新のみを評価することでジオフェンスの評価コストを削減できます。

位置の更新をすでに受信した後でトラッカーをジオフェンスコレクションに関連付けると、関連付け後の最初の位置更新は、ジオフェンス評価の初期更新として扱われます。ジオフェンス内にある場合は、ENTER イベントを受信します。どのジオフェンス内にもない場合は、以前の状態に関係なく EXIT イベントは受信されません。

位置フィルタリング

トラッカーは、送信されたポジションを自動的にフィルタリングできます。デバイスのロケーションの更新の一部を除外するのには、いくつかの理由があります。1 分おきにしかレポートを送信しないシステムの場合は、デバイスを時間でフィルタリングし、30 秒ごとに位置の保存と評価のみを行いたいと思うかもしれません。より頻繁に監視している場合でも、位置の更新をフィルタリングして GPS ハードウェアの雑音を取り除きたい場合があります。GPS 位置情報には本質的にノイズが多く

含まれます。その精度は 100% 完璧ではないため、静止しているデバイスでも少し動き回っているように見えます。低速では、デバイスがジオフェンスのエッジの近くにあると、このジッターによって視覚的に乱雑になり、誤った入退室イベントが発生する可能性があります。

位置フィルタリングは、位置の更新をトラッカーが受信すると同時に機能するため、デバイスパス内の視覚的なノイズ (ジッター) が減り、ジオフェンスの誤った入退イベントの数が減ります。また、位置更新の保存回数やジオフェンス評価のトリガー回数が減るため、コスト管理に役立ちます。

トラッカーには 3 つの位置フィルターオプションがあります。コスト管理と位置情報の更新におけるジッターの軽減に役立ちます。

- 精度ベース — 精度測定が可能なあらゆるデバイスで使用できます。ほとんどの GPS やモバイル機器がこの情報を提供します。各位置測定の精度は、GPS 衛星の受信状況、風景、Wi-Fi デバイスや Bluetooth デバイスの近接性など、さまざまな環境要因の影響を受けます。モバイル機器を含むほとんどのデバイスでは、測定と同時にその精度も推定できます。`AccuracyBased` フィルタリングでは、デバイスの移動距離が測定精度より少ない場合、Amazon Location は位置の更新を無視します。例えば、デバイスから 2 つの連続した更新を受信し、その精度が 5 m と 10 m だった場合、デバイスの移動距離が 15 m 未満であれば 2 回目の更新は無視されます。Amazon Location は、無視した更新をジオフェンスと比較して評価したり、保存したりしません。

  精度が提供されない場合はゼロとして扱われ、測定値は完全に正確であると見なされ、更新にはフィルタリングは適用されません。

  > **Note**
  >
  > 精度に基づくフィルタリングを使用すると、すべてのフィルタリングを削除できます。精度ベースのフィルタリングを選択したものの、すべての精度データをゼロに上書きしたり、精度を完全に無視した場合、Amazon Location は更新をフィルタリングしません。

  ほとんどのシナリオでは、位置の更新をフィルタリングするには精度ベースのフィルタリングが適しています。位置情報の追跡と不要な更新を除外することのバランスを取ることができるため、コストを削減できます。

- 距離ベース — 使用しているデバイスでは正確な測定値が得られないが、フィルタリングを活用してジッターを減らし、コストを管理したい場合に使用します。`DistanceBased` フィルタリングでは、デバイスの移動が 30 m (98.4 フィート) 未満の場合、ロケーションの更新は無視されます。`DistanceBased` 位置フィルターを使用すると、Amazon Location は無視した更新をジオフェンスと比較して評価したり、保存したりしません。

iOS および Android デバイスの平均精度を含め、ほとんどのモバイル機器の精度は 15m 以内です。ほとんどのアプリケーションでは、DistanceBased フィルタリングを使用することで、デバイスの軌跡を地図上に表示するときに位置が不正確になることによる影響や、デバイスがジオフェンスの境界近くにあるときに複数の連続した出入りイベントによるバウンス効果を減らすことができます。また、リンクされたジオフェンスと比較して評価するリクエストや、デバイスの位置を取得するリクエストが減るので、アプリケーションのコスト削減にも役立ちます。

距離ベースのフィルタリングは、フィルタリングしたいがデバイスでは正確な測定値が得られない場合や、精度ベースの場合よりも多くの更新を除外したい場合に便利です。

- 時間ベース — (デフォルト) デバイスが位置の更新を非常に頻繁に (30 秒に 1 回以上) 送信し、すべての更新を保存せずにほぼリアルタイムでジオフェンスの評価を行いたい場合に使用します。TimeBased フィルタリングでは、位置の更新ごとに、リンクされたジオフェンスコレクションと比較して評価しますが、すべてが保存されるわけではありません。更新頻度が 30 秒を超える場合、一意のデバイス ID ごとに 30 秒あたり 1 つの更新のみが保存されます。

  時間ベースのフィルタリングは、保存する位置の数は少ないが、位置の更新はすべて関連するジオフェンスコレクションと照合して評価したい場合に特に便利です。

> **Note**
>
> フィルタリング方法や位置更新の頻度を決める際には、トラッキングアプリケーションのコストに留意してください。位置の更新ごとに請求され、リンクされた各ジオフェンスコレクションと比較して評価する際にも 1 回請求されます。例えば、時間ベースのフィルタリングを使用する場合で、トラッカーが 2 つのジオフェンスコレクションにリンクされている場合は、位置の更新ごとに 1 回の位置更新リクエストと 2 回のジオフェンスコレクション評価としてカウントされます。デバイスの位置更新を 5 秒ごとに報告し、時間ベースのフィルタリングを使用している場合は、デバイスごとに 1 時間あたり 720 回の位置更新リクエストと 1,440 回のジオフェンス評価について課金されます。

ジオフェンスの用語

ジオフェンスコレクション

0 個以上のジオフェンスが含まれます。要求に応じて Entry イベントと Exit イベントを発行して、デバイスの位置をジオフェンスと照合して評価することで、ジオフェンスを監視できます。

ジオフェンス

マップ上の仮想境界を定義するポリゴンまたは円のジオメトリ。

ポリゴンジオメトリ

Amazon Location ジオフェンスは地理的領域の仮想境界であり、ポリゴンジオメトリまたは円として表されます。

円は、周囲に距離がある点です。デバイスが特定の場所から一定の距離内にある場合に通知を受け取りたい場合は、円を使います。

多角形は、1 つ以上の線状のリングで構成された配列です。デバイス通知に特定の境界を定義する場合は、多角形を使用します。線形リングは 4 つ以上の頂点の配列で、最初と最後の頂点が同じで閉じた境界を形成します。各頂点は *[#####]* という形の 2 次元の点で、経度と緯度の単位は度です。頂点は多角形の周囲を反時計回りの順序で並べる必要があります。

> **Note**
>
> Amazon Location Service、複数のリングを持つポリゴンをサポートしていません。これには、ホール、アイランド、マルチポリゴンが含まれます。Amazon Location では、時計回りに曲がっているポリゴンや反経線と交差するポリゴンもサポートしていません。

次は、1 つのリニア外部リングの例です。

```
[
  [
    [-5.716667, -15.933333],
    [-14.416667, -7.933333],
    [-12.316667, -37.066667],
    [-5.716667, -15.933333]
  ]
]
```

トラッカーの用語

トラッカーリソース

デバイスから位置情報の更新を受信する AWS リソース。トラッカーリソースは、現在および過去のデバイスの位置情報などの位置情報クエリをサポートします。トラッカーリソースをジオフェンスコレクションにリンクすると、リンクされたジオフェンスコレクション内のすべてのジオフェンスに対して位置情報の更新が自動的に評価されます。

位置データのトラッキング

トラッカーリソースは、時間の経過と共にデバイスに関する情報を保存します。この情報には一連の位置更新が含まれ、各更新には位置情報、時刻、およびオプションのメタデータが含まれます。メタデータには、位置の精度に加え、追跡対象車両の速度、方向、タイヤ空気圧、燃料残量、エンジン温度など、各位置に関する重要な情報を追跡するのに役立つ最大 3 つのキーと値のペアを含めることができます。トラッカーは、デバイスの位置履歴を 30 日間保持します。

位置フィルタリング

位置フィルタリングは、貴重な情報を提供しない位置の更新を、その更新が保存されるか、ジオフェンスに対して評価される前に除外することで、コストを管理し、追跡アプリケーションの品質を向上させるのに役立ちます。

`AccuracyBased`、`DistanceBased`、または `TimeBased` のフィルタリングを選択できます。デフォルトでは、位置フィルタリングは `TimeBased` に設定されています。

トラッカーリソースを作成または更新するとき、位置フィルタリングを設定できます。

RFC 3339 タイムスタンプフォーマット

Amazon Location Service ストラッカーは、日付と時刻に関しては国際標準化機構 (ISO) 8601 形式に従う RFC 3339 形式を使用しています。

形式は「YYYY-MM-DDThh:mm:ss.sssZ+00:00」です。

- `YYYY-MM-DD`— 日付形式を表します。
- `T`— 時刻の値もそれに続くことを示します。
- `hh:mm:ss.sss`— 24 時間形式の時刻を表します。
- `Z`— 使用されているタイムゾーンが UTC であることを示します。UTC タイムゾーンとは異なる場合もあります。
- `+00:00`— オプションで UTC タイムゾーンからの逸脱を示します。例えば、+ 01:00 は UTC + 1 時間であることを示します。

例

2020 年 7 月 2 日の午後 12 時 15 分 20 分については、UTC タイムゾーンに合わせて 1 時間が追加されます。

```
2020-07-02T12:15:20.000Z+01:00
```

Amazon Location Service を使用する一般的なユースケース

Amazon Location Service では、アセットトラッキングからロケーションベースのマーケティングまで、さまざまなアプリケーションを構築できます。一般的なユースケースは次のとおりです。

ユーザーエンゲージメントとジオマーケティング

位置データを使用して、ターゲットを絞った顧客へのマーケティングにおけるユーザーエンゲージメントを向上させるソリューションを構築します。例えば、モバイルアプリでコーヒーを注文した顧客が近くにいると、Amazon Location によって通知を促すイベントをトリガーできます。さらに、地域ターゲティング機能を構築して、小売業者が対象店舗の近くにいる顧客にディスカウントコードやデジタルチラシを送信できるようにすることもできます。

アセットトラッキング

アセットトラッキング機能を構築して、企業が自社の製品、人材、インフラの現在および過去の位置を把握できるようにしましょう。アセットトラッキング機能を使えば、リモートスタッフ配置の最適化、輸送中の貨物の安全確保、発送効率の最大化を実現するさまざまなソリューションを構築できます。

配送

ロケーション機能を配送アプリケーションに統合して、出発地、配送車両、配送先を保存、追跡、調整できます。例えば、Amazon Location 機能が組み込まれたフードデリバリーアプリケーションには、配達ドライバーが近くにいることをレストランに自動的に通知できる位置追跡機能とジオフェンシング機能があります。これにより、待ち時間が短縮され、配達される食品の品質を維持できます。

このトピックでは、Amazon Location を使用して構築できるアプリケーションのアーキテクチャと手順の概要を説明します。

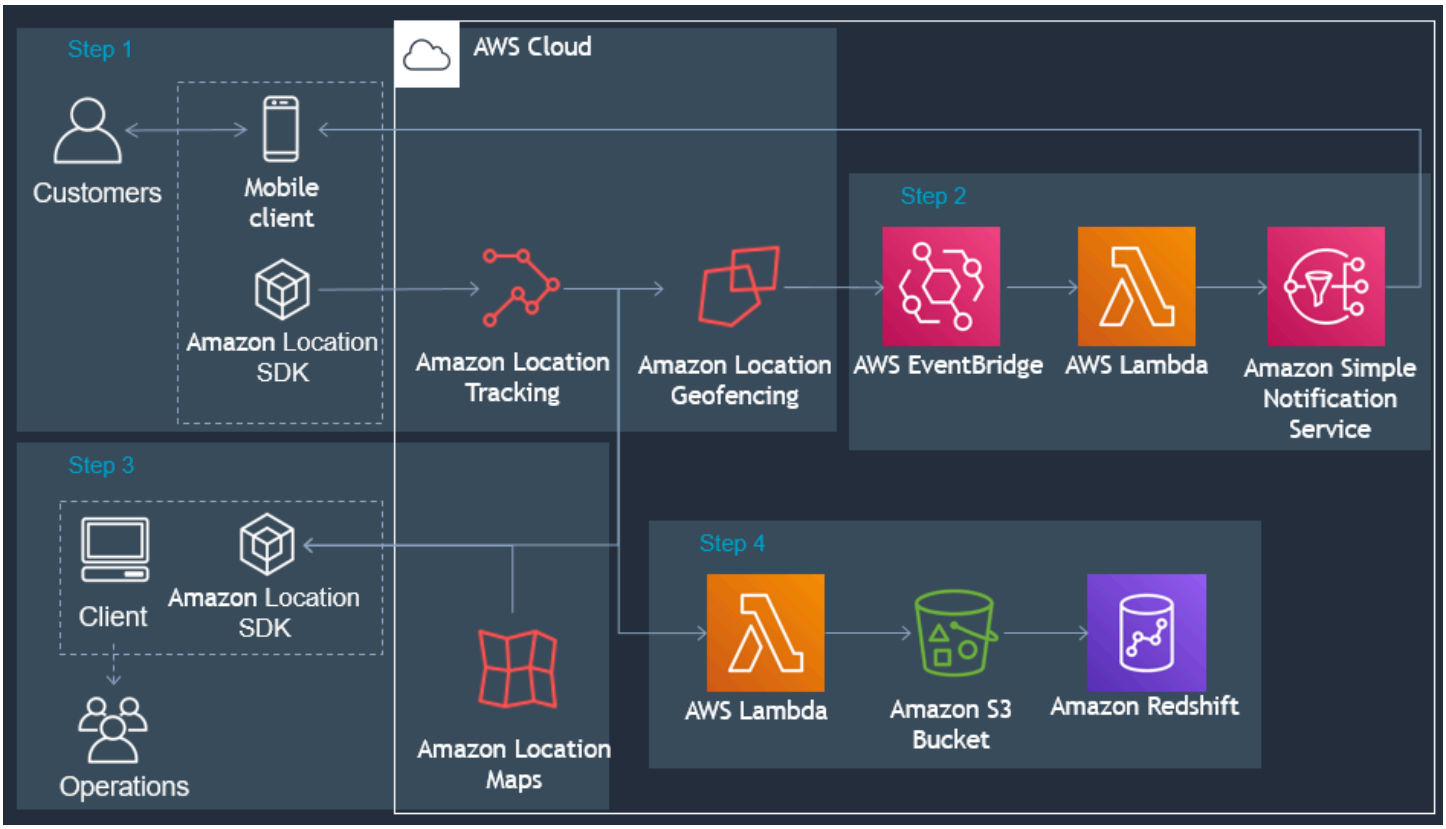
トピック

ユーザーエンゲージメントとジオマーケティングアプリケーション

以下は、Amazon Location を使用したユーザーエンゲージメントとジオマーケティングのアプリケーションアーキテクチャを示しています。

このアーキテクチャでは、次のことが可能になります。

- ターゲットの近さに基づいてイベントを開始することで、近くの顧客にオファーを送ったり、店舗を去ったばかりの顧客を引き付けたりできます (ジオターゲティングと呼ばれます)。
- 顧客のデバイスの位置を地図上で視覚化して、時間の経過に伴う傾向を監視します。
- 顧客のデバイス位置を保存して、時間をかけて分析できるようにする。
- ロケーション履歴を分析して、傾向や最適化の機会を特定します。



以下は、ユーザーエンゲージメントとジオマーケティングアプリケーションの構築に必要な手順の概要です。

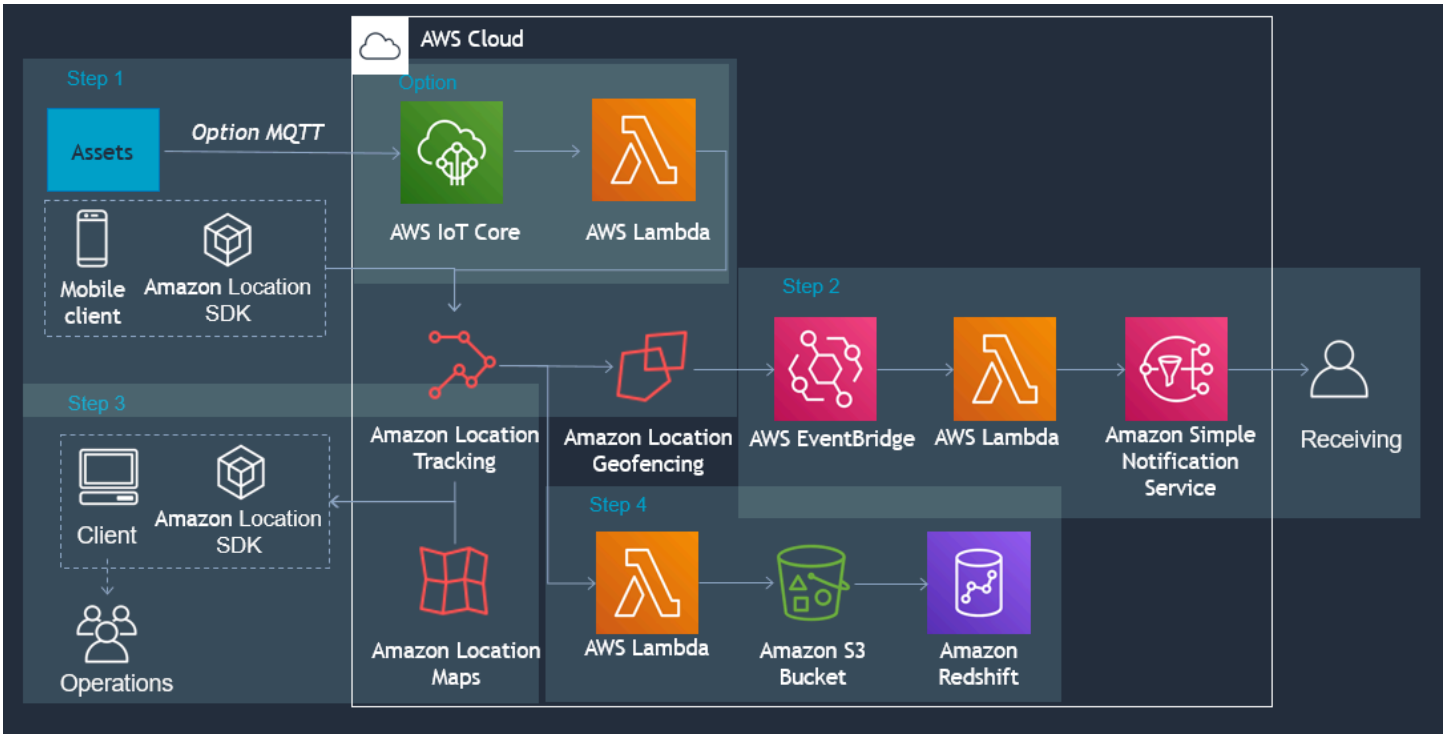
1. ジオフェンスコレクションにジオフェンスを作成し、トラッカーをそれらにリンクします。詳細については、「the section called “ジオフェンシングと追跡”」を参照してください。
2. ジオフェンスの対象エリアに出入りするお客様に通知を送信する EventBridge ように Amazon を設定します。詳細については、「the section called “でのイベントへの対応 EventBridge”」を参照してください。
3. 顧客の位置とジオフェンスを地図上に表示します。詳細については、「マップの使用」を参照してください。
4. 位置データを長期保存し、さらに分析できるようにします。
5. アプリケーションを構築したら、Amazon CloudWatch と AWS CloudTrail を使用してアプリケーションを管理できます。詳細については、「the section called “によるモニタリング CloudWatch”」および「the section called “Amazon Location CloudTrail での の使用”」を参照してください。

アセットトラッキングアプリケーション

以下は、Amazon Location を使用するアセットトラッキングアプリケーションアーキテクチャの例です。

このアーキテクチャでは、次のことが可能になります。

- 資産の位置を地図上に表示して全体像を把握できます。例えば、運用チームや計画チームに役立つように、過去の場所やイベントを含むヒートマップを表示できます。
- アセットの近さに基づいてイベントを開始し、荷物の到着に備えて荷物の到着に備えて処理時間を短縮するよう受領部門に通知します。
- 資産の場所を保存して、バックエンドアプリケーションでアクションを開始したり、データを時系列的に分析したりできます。
- ロケーション履歴を分析して、傾向や最適化の機会を特定します。



以下は、アセットトラッキングアプリケーションの構築に必要な手順の概要を示しています。

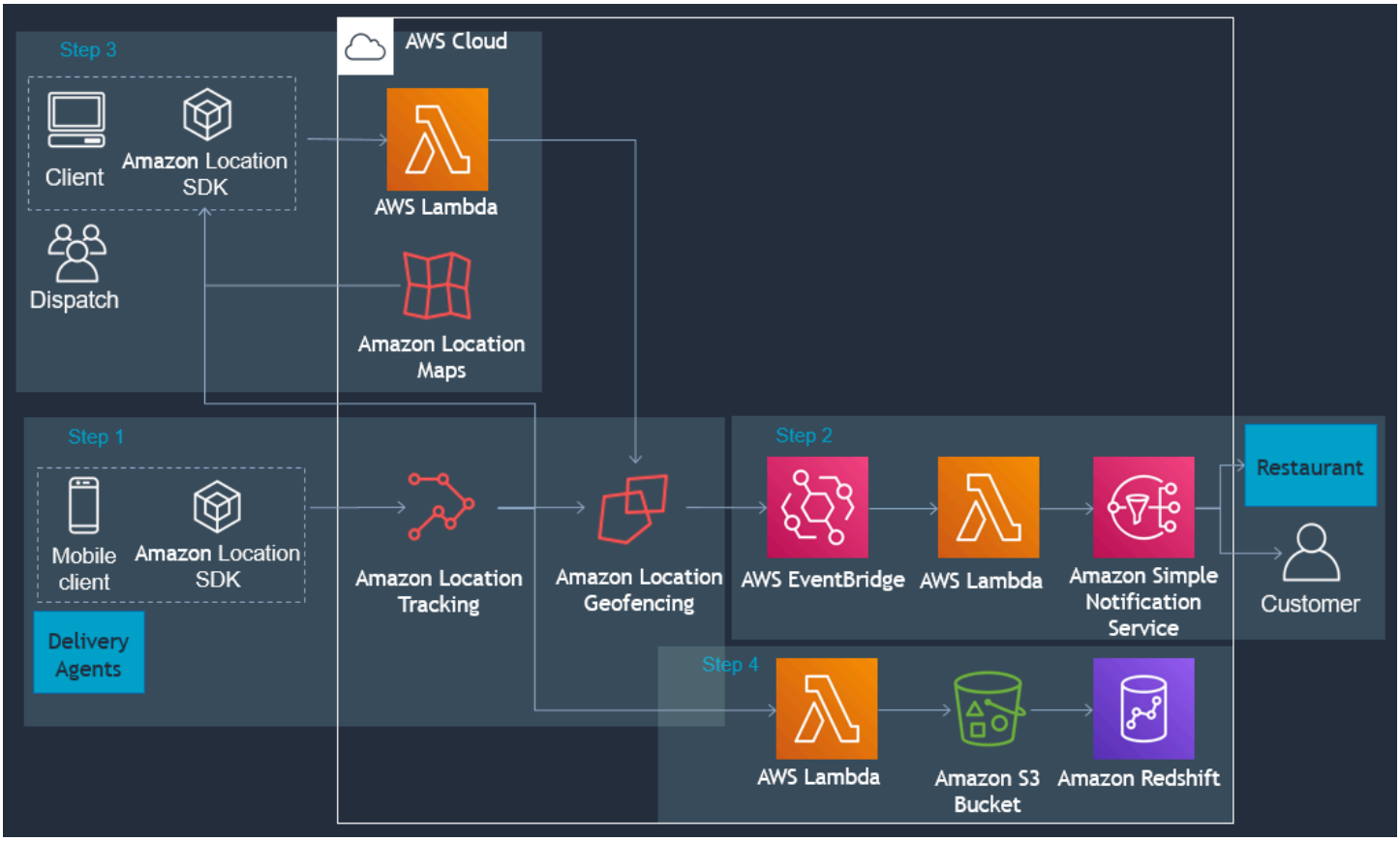
1. ジオフェンスコレクションにジオフェンスを作成し、トラッカーをそれらにリンクします。詳細については、「the section called “ジオフェンシングと追跡”」を参照してください。
2. 通知を送信するか、プロセスを開始する EventBridge ように Amazon を設定します。詳細については、「the section called “でのイベントへの対応 EventBridge”」を参照してください。
3. 追跡対象アセットとアクティブなジオフェンスを地図上に表示します。詳細については、「マップの使用」を参照してください。
4. さらに分析できるように、位置データを長期ストレージに保存します。
5. アプリケーションを構築したら、Amazon CloudWatch と AWS CloudTrail を使用してアプリケーションを管理できます。詳細については、「the section called “によるモニタリング CloudWatch”」および「the section called “Amazon Location CloudTrail での の使用”」を参照してください。

配信アプリケーション

以下は、Amazon Location を使用する配信アプリケーションアーキテクチャの例です。

このアーキテクチャでは、次のことが可能になります。

- 配達業者の近さに基づいてイベントを開始することで、集荷の準備が間に合い、配達が到着すると顧客に通知できるようになります。
- ドライバーの所在地だけでなく、乗車場所や降車場所もほぼリアルタイムで地図上に表示し、派遣チームに全体像を把握できます。
- 配送業者の位置情報を保存して、バックエンドアプリケーションで操作したり、時間をかけて分析したりできるようにします。
- ロケーション履歴を分析して、傾向や最適化の機会を特定します。



以下は、配信アプリケーションの構築に必要な手順の概要です。

1. ジオフェンスコレクションを作成し、追跡対象デバイスをコレクションにリンクします。詳細については、the section called “ジオフェンシングと追跡” を参照してください。
2. 注文の予約時にジオフェンスを自動的に追加および削除する AWS Lambda 関数を作成します。
3. 通知を送信するか、プロセスを開始する EventBridge ように Amazon を設定します。詳細については、「the section called “でのイベントへの対応 EventBridge”」を参照してください。

4. 追跡されたアセットとアクティブなジオフェンスを地図上に表示します。詳細については、「マップの使用」を参照してください。
5. 位置データを長期保存し、さらに分析できるようにします。
6. アプリケーションを構築したら、Amazon CloudWatch と AWS CloudTrail を使用してアプリケーションを管理できます。詳細については、「the section called “によるモニタリング CloudWatch”」および「the section called “Amazon Location CloudTrail での の使用”」を参照してください。

データプロバイダーとは何ですか？

Amazon Location Service を使用すると、第三者との契約や統合を必要とせずに、 AWS アカウントを通じて複数のデータプロバイダーの位置情報リソースにアクセスできます。これにより、サードパーティのアカウント、認証情報、ライセンス、請求を管理しなくても、アプリケーションの構築に集中できます。

次の Amazon Location Service はデータプロバイダーを使用します。

- マップ — マップリソースを作成 するときに、さまざまなマッププロバイダーからスタイルを選択します。マップリソースを使用して、データを視覚化するインタラクティブなマップを構築できます。
- 場所 — ジオコーディング、リバースジオコーディング、検索のクエリをサポートするプレースインデックスリソースを作成時に、データプロバイダーを選択してください。
- ルート — ルート計算リソースを作成する ときに、さまざまな地域やアプリケーションでのルート計算のクエリをサポートするデータプロバイダーを選択してください。Amazon Location Service では、選択したデータプロバイダーを使用して、 up-to-date 道路ネットワークデータ、ライブトラフィックデータ、計画的な閉鎖、過去のトラフィックパターンに基づいてルートを計算できます。

プロバイダはそれぞれ異なる方法でデータを収集し、管理します。また、世界の地域によって専門知識が異なる場合もあります。このセクションでは、データプロバイダーの詳細を示します。好みに応じて、どのデータプロバイダーでも選択できます。

Amazon Location Service データプロバイダーを使用する際は、利用規約を必ずお読みください。詳細については、「AWS サービス条件」を参照してください。Amazon Location がお客様のプライバシーをどのように保護するかについての詳細は、the section called “データプライバシー” セクションを参照してください。

データプロバイダーの対象範囲と機能

次の表は、各データプロバイダーの対象範囲と機能の概要を示しています。

データプロバイダー	地理カバレッジエリア	機能のカバレッジ	AWS リージョン
Esri	グローバル	マップ、場所、ルート	Amazon Location が利用可能なすべてのリージョン。
Grab	東南アジア	マップ、場所、ルート	アジアパシフィック (シンガポール)、ap-southeast-1 のみ。
HERE	グローバル	マップ、場所、ルート	Amazon Location が利用可能なすべてのリージョン。
オープンデータ	グローバル	マップ	Amazon Location が利用可能なすべてのリージョン。

各データプロバイダーの特定の機能の詳細については、「データプロバイダー別の機能」を参照してください。

データプロバイダーは、それぞれ異なる方法でデータを収集して生成します。対象地域について詳しくは、次のトピックで確認できます。

- カバレージ: Esri
- カバレッジ: Grab
- カバレッジ: HERE
- カバレッジ: Open Data

データに問題が発生し、データプロバイダーにエラーを報告したい場合は、以下のトピックを参照してください。

- Esri へのエラー報告
- GrabMaps データのエラーレポート
- HERE へのエラー報告
- エラー報告とオープンデータへの寄稿

マップスタイル

各データプロバイダーは、提供するマップデータをレンダリングするためのマップスタイルセットを提供しています。例えば、スタイルには衛星画像が含まれていたり、ナビゲーション用に道路を表示するように最適化されている場合があります。各プロバイダーのスタイルのリストと例については、以下のトピックを参照してください。

- Esri マップスタイル
- Grab マップスタイル
- HERE マップスタイル
- Open Data マップスタイル

各データプロバイダーに関する詳細情報

次のリンクでは、各データプロバイダーに関する詳細情報が提供されます。

- Esri
- GrabMaps
- HERE テクノロジーズ
- オープンデータ

Esri

Amazon Location Service は、Esri の位置情報サービスを使用して、 AWS お客様がマップ、ジオコード、ルートの効率的な計算を行うのに役立ちます。Esri の位置情報サービスは、高品質で権威のある ready-to-use 位置情報データで構築されており、地図学者、地理学者、デモグラファーの専門家チームによってキュレートされています。

追加の機能情報については、Amazon Location Service データプロバイダの「Esri」を参照してください。

トピック

Esri マップスタイル

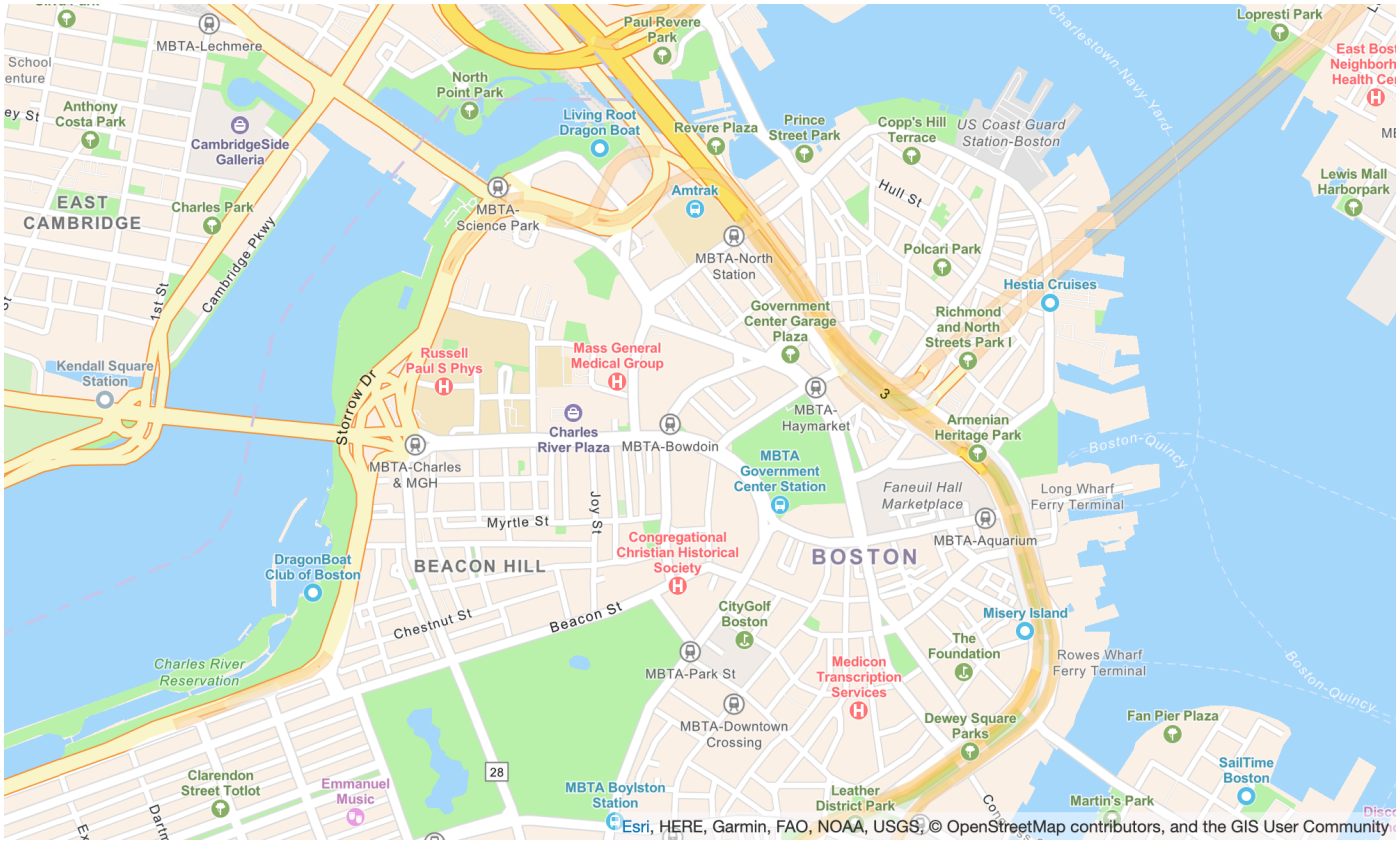
Amazon Location Service、マップリソースの作成 時に次の Esri マップスタイルをサポートします。

> **Note**
>
> このセクションに記載されていない Esri マップスタイルはサポートされていません。

Esri のベクタースタイルは代替 政治的見解 をサポートしています。

Esri Navigation

Esri Navigation



マップスタイル名: `VectorEsriNavigation`

このマップは、モバイルデバイスで日中に使用するように設計されたカスタムナビゲーションのマップスタイルでシンボル化された世界の詳細なベースマップを提供します。

この包括的な市街地図には、高速道路、主要道路、補助道路、鉄道、水域、都市、公園、ランドマーク、建物の敷地、行政区域が含まれています。このマップのベクタータイルレイヤーは、World Street Map やその他の Esri ベースマップに使用されているのと同じデータソースを使用して構築されています。`POI` レイヤーを に設定して有効にしCustomLayers、追加の場所データを活用します。

詳細については、Esri Web サイトの「Esri ワールドナビゲーション」をご参照ください。

> **Note**
>
> 上の図の `VectorEsriNavigation` マップでは、`POI` レイヤーが有効化されています。

[Fonts]

Amazon Location では、`GetMapGlyphs` を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

- Arial Italic
- Arial Regular
- Arial Bold
- Arial Unicode MS Bold
- Arial Unicode MS Regular

Esri Imagery

Esri Imagery

マップスタイル名: `RasterEsriImagery`

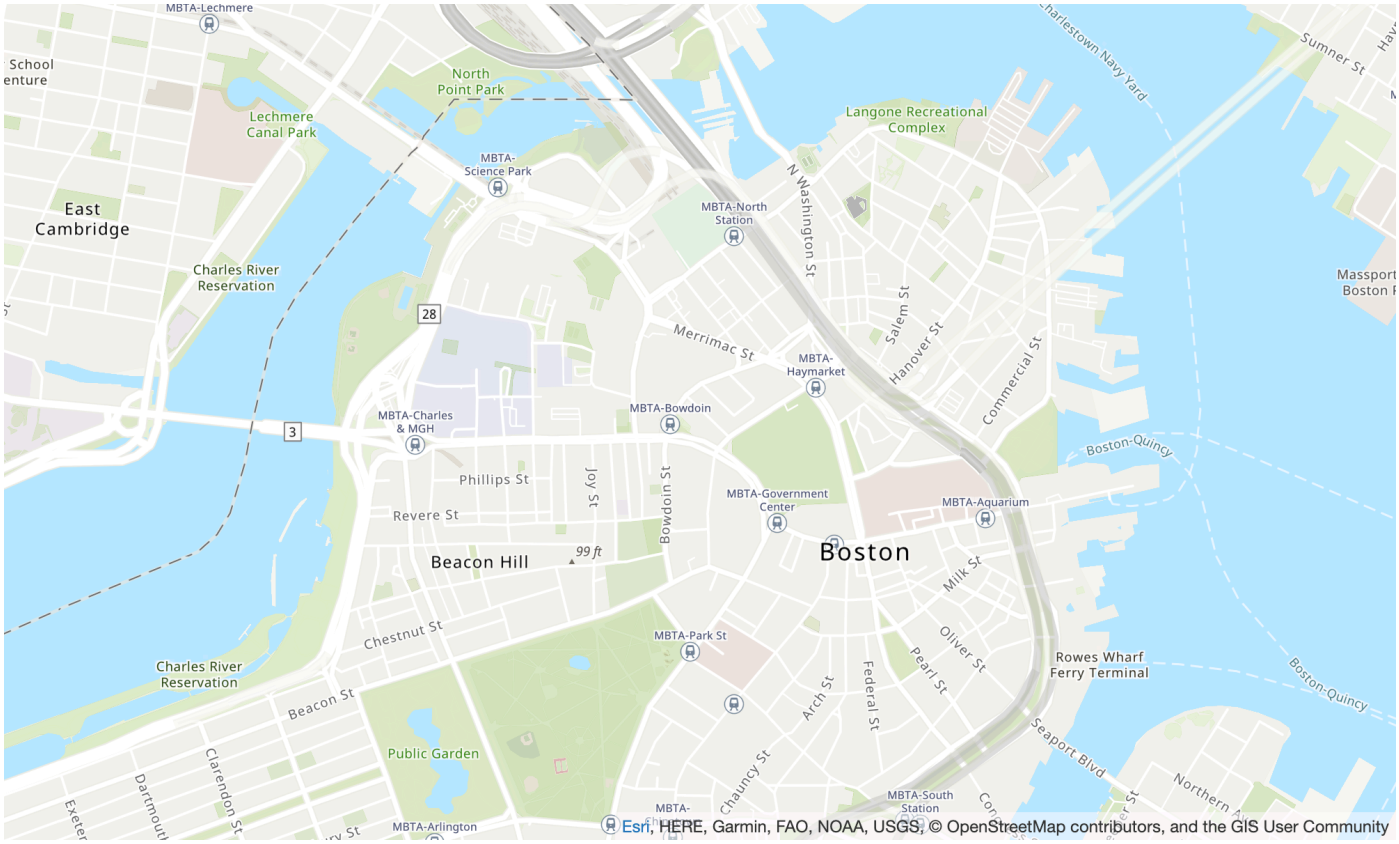
このマップは、世界の多くの地域で 1 メートル以上の衛星画像と航空画像を提供し、世界中で低解像度の衛星画像を提供します。

このマップには、世界中の小規模および中縮尺の 15 m 画像 (約 1:591 M から 1:72 k まで) と 2.5 m の SPOT 画像 (約 1:288 k ~ 1:72 k) が含まれています。このマップには、Maxar から撮影した米国本土と西ヨーロッパの一部の 0.5 m 解像度の画像が掲載されています。このマップには、世界の多くの地域にある Maxar サブメーター画像が追加されています。世界の他の地域では、GIS ユーザーコミュニティがさまざまな解像度の画像を提供しています。一部のコミュニティでは、1:280 程度の縮尺までの非常に高解像度の画像 (最小 0.03 m) が入手可能です。

詳細については、Esri Web サイトの「Esri World Imagery」を参照してください。

Esri Light

Esri Light



マップスタイル名: `VectorEsriTopographic`

これにより、従来の Esri マップスタイルでシンボル表示された詳細な世界ベースマップが提供されます。これには、高速道路、主要道路、補助道路、鉄道、水域、都市、公園、ランドマーク、建物の敷地、行政区域が含まれます。

このベースマップは、米国地質調査所 (USGS)、米国環境保護庁 (EPA)、米国国立公園局 (NPS)、国連食糧農業機関 (FAO)、カナダ天然資源省 (NRCAN)、HERE、Esri など、複数のデータプロバイダーからのさまざまな信頼できる情報源から編集されています。選択したエリアのデータは OpenStreetMap 寄稿者から取得されます。さらに、データは GIS コミュニティから提供されています。

[Fonts]

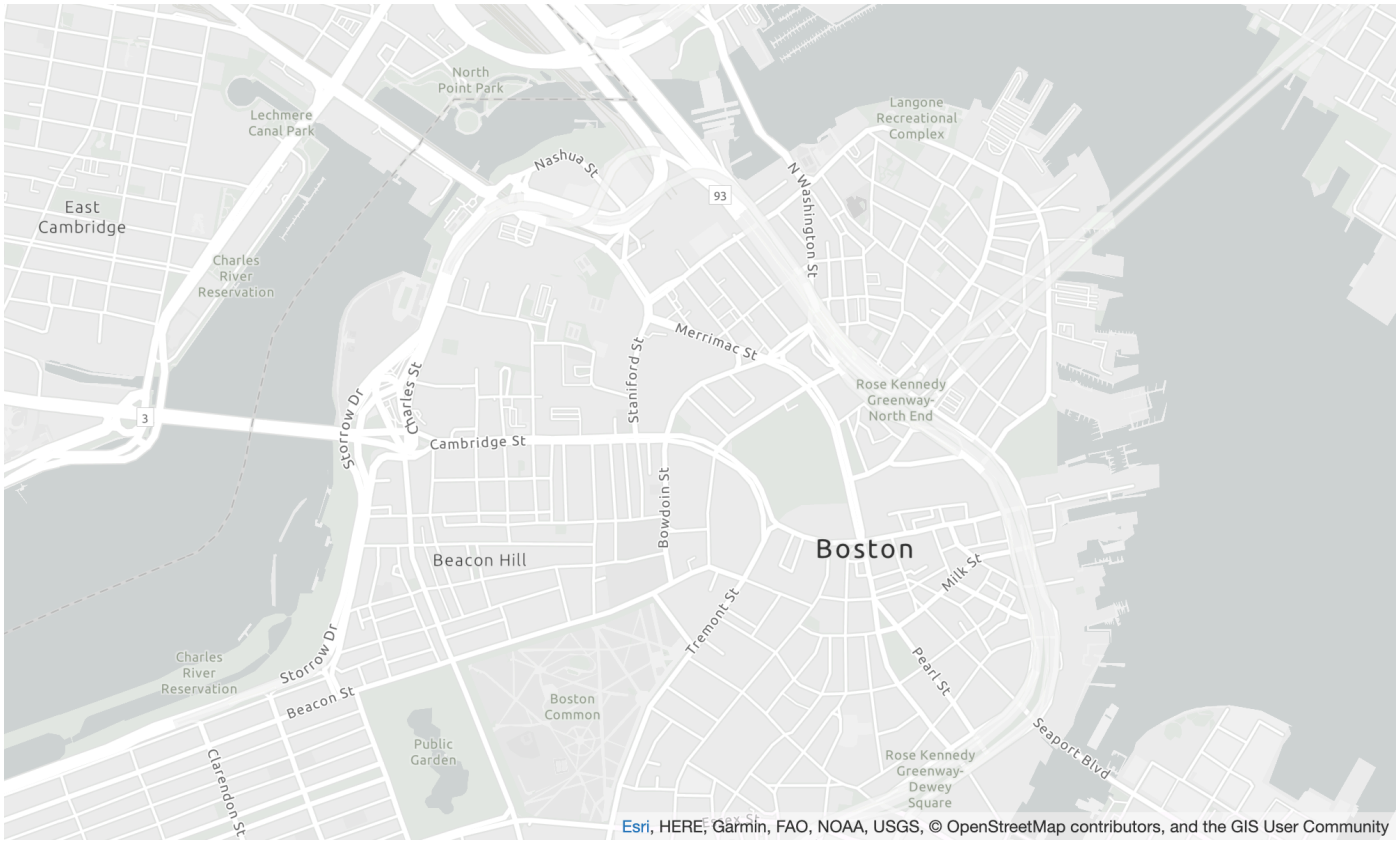
Amazon Location では、`GetMapGlyphs` を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

- Noto Sans Italic
- Noto Sans Regular
- Noto Sans Bold

- Noto Serif Regular
- Roboto Condensed Light Italic

Esri Light Gray Canvas

Esri Light Gray Canvas



マップスタイル名: `VectorEsriLightGrayCanvas`

このマップは、テーマコンテンツに注目を集めるように設計された、最小限の色、ラベル、機能を備えたライトグレーの二ュートラルな背景スタイルでシンボル化された世界の詳細なベースマップを提供します。

このベクタータイルレイヤーは、ライトグレーキャンバスやその他の Esri ベースマップに使用されているのと同じデータソースを使用して構築されています。マップには、高速道路、主要道路、補助道路、鉄道、水域、都市、公園、ランドマーク、建物の敷地、行政区域が含まれています。

詳細については、Esri Web サイトの「Esri ライトグレーキャンバス」をご参照ください。

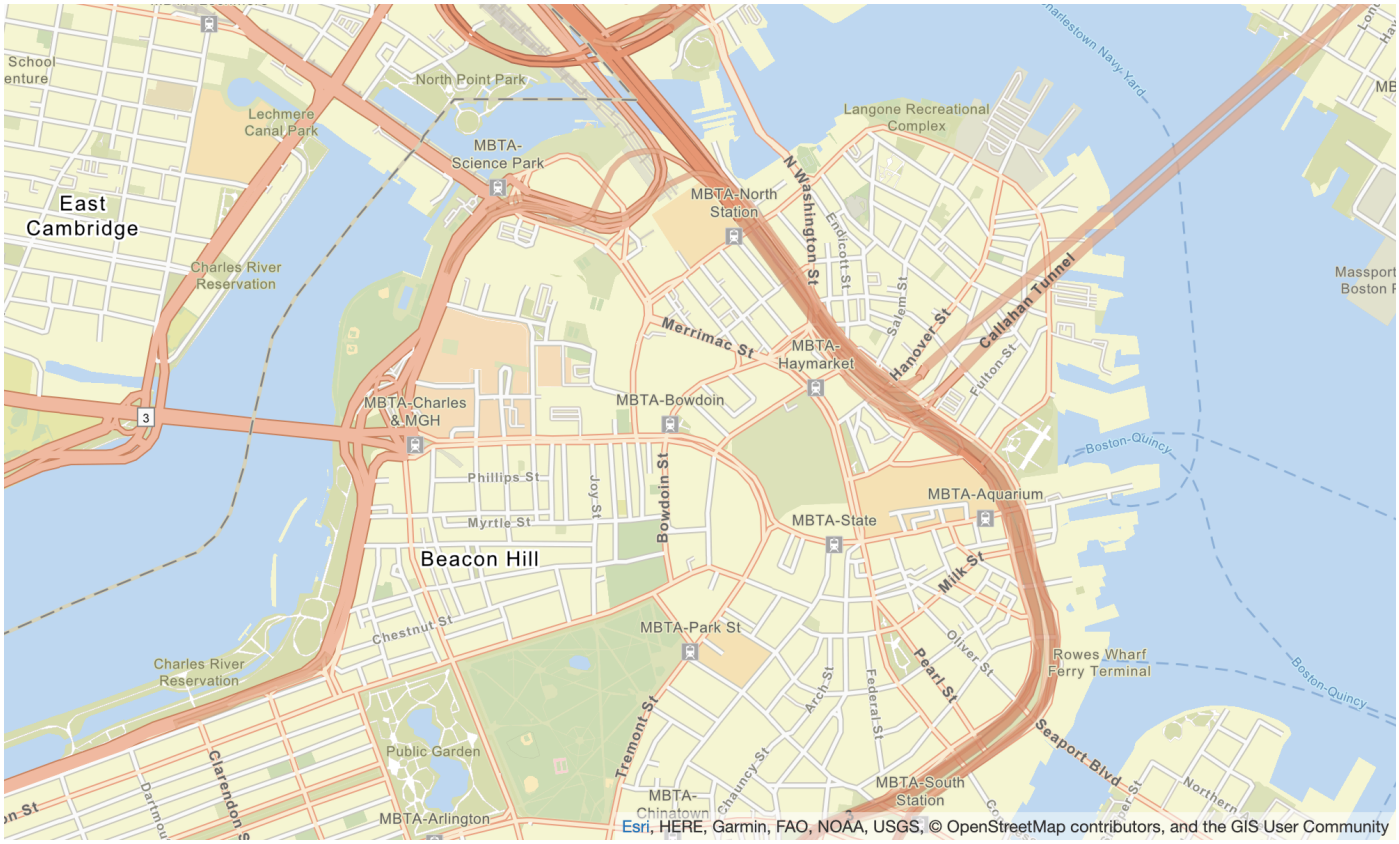
[Fonts]

Amazon Location では、GetMapGlyphs を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

- Ubuntu Italic
- Ubuntu Regular
- Ubuntu Light
- Ubuntu Bold

Esri Street Map

Esri Street Map



マップスタイル名: `VectorEsriStreets`

このマップは、モバイルデバイスで日中に使用するように設計されたカスタムナビゲーションのマップスタイルでシンボル化された世界の詳細なベースマップを提供します。

この包括的な市街地図には、高速道路、主要道路、補助道路、鉄道、水域、都市、公園、ランドマーク、建物の敷地、行政区域が含まれています。また、ショップ、サービス、レストラン、アトラクション、その他の名所など、さまざまな場所の豊富なセットも含まれています。このマッ

プのベクタータイルレイヤーは、World Street Map やその他の Esri ベースマップに使用されているのと同じデータソースを使用して構築されています。

詳細については、Esri Web サイトの「Esri World Street」をご参照ください。

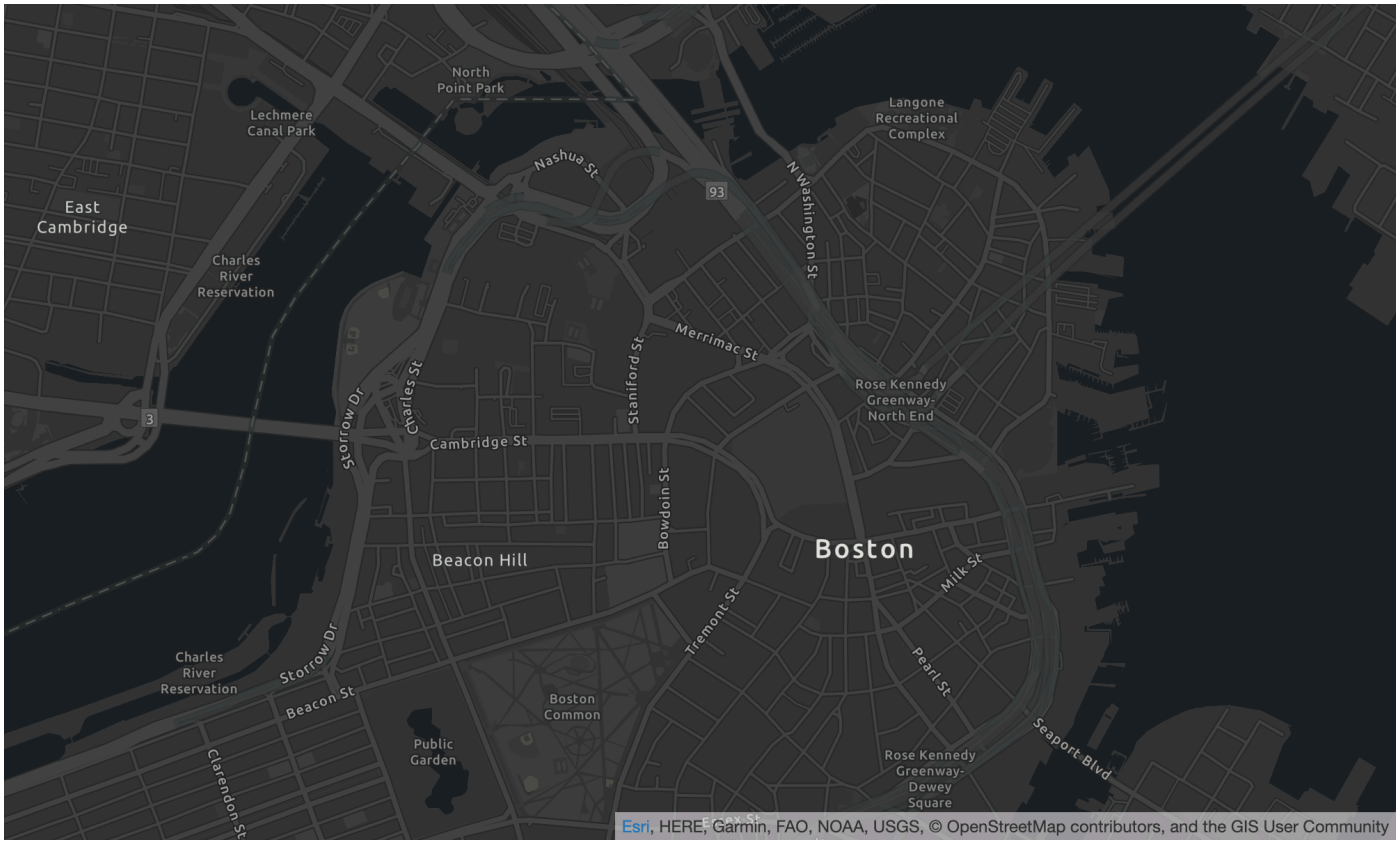
[Fonts]

Amazon Location では、`GetMapGlyphs` を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

- Arial Italic
- Arial Regular
- Arial Bold
- Arial Unicode MS Bold
- Arial Unicode MS Regular

Esri Dark Gray Canvas

Esri Dark Gray Canvas



マップスタイル名: `VectorEsriDarkGrayCanvas`

このマップは、テーマコンテンツに注目を集めるように設計された最小限の色、ラベル、機能を備えたダークグレーのニュートラルな背景スタイルでシンボル化された世界の詳細なベクターベースマップを提供します。

このマップには、高速道路、主要道路、補助道路、鉄道、水域、都市、公園、ランドマーク、建物の敷地、行政区域が含まれています。このマップのベクタータイルレイヤーは、Dark Gray Canvas ラスターマップやその他の Esri ベースマップに使用されているのと同じデータソースを使用して構築されています。

詳細については、Esri Web サイトの「Esri ダークグレーキャンバス」をご参照ください。

[Fonts]

Amazon Location では、`GetMapGlyphs` を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

- Ubuntu Medium Italic
- Ubuntu Medium
- Ubuntu Italic
- Ubuntu Regular
- Ubuntu Bold

カバレージ: Esri

Esri をデータプロバイダーとして使用すると、プレースインデックスリソースの作成時にジオコーディング、リバースジオコーディング、検索のクエリをサポートしたり、ルート計算リソースを作成時にルートを計算するクエリをサポートしたりできます。

Esri は、世界のさまざまな地域でさまざまなレベルのデータ品質を提供しています。対象地域のカバレッジに関する追加情報については、以下を参照してください。

- Esri のジオコーディングカバレッジの詳細
- 道路網と交通量に関する Esri の詳細

利用規約とデータアトリビューション: Esri

Esri のデータを使用する前に、Esri と に適用されるライセンス条項など、適用されるすべての法的要件に準拠できることを確認してください AWS。

AWS 要件の詳細については、「AWS サービス条件」を参照してください。

Esri の帰属ガイドラインについては、「Esri の データ帰属と利用規約」を参照してください。

Esri へのエラー報告

データに問題が発生し、エラーや不一致を Esri に報告したい場合は、Esri のテクニカルサポート記事「方法: ベースマップとジオコーディングに関するフィードバックの提供」を参照してください。

GrabMaps

Grab は東南アジア最大の配送会社で、数百万のドライバーパートナーと顧客を抱えています。彼らの傍受者である はGrabMaps up-to-date、それらの国/地域で独自の使用のためにマッピングデータを作成します。Amazon Location Service は、 GrabMapsの位置情報サービスを使用して、 AWS お客様がマップ、ジオコード、ルートを効果的に使用できるよう支援します。 GrabMapsの位置情報サービスは、特に東南アジアの国向けに、高品質で権威のある ready-to-use 口ケーションデータを提供するように構築されています。

その他の機能の詳細については、Amazon Location Service データプロバイダーのGrabMaps「」を参照してください。

> **Important**
>
> Grab は、東南アジアのみの地図を提供しており、アジアパシフィック (シンガポール) リージョン (ap-southeast-1) でのみ利用できます。詳細については、「国/地域と対象地域」を参照してください。

トピック

- Grab マップスタイル
- カバレッジ: Grab
- 国/地域と対象地域
- 利用規約とデータアトリビューション: Grab
- GrabMaps データのエラーレポート

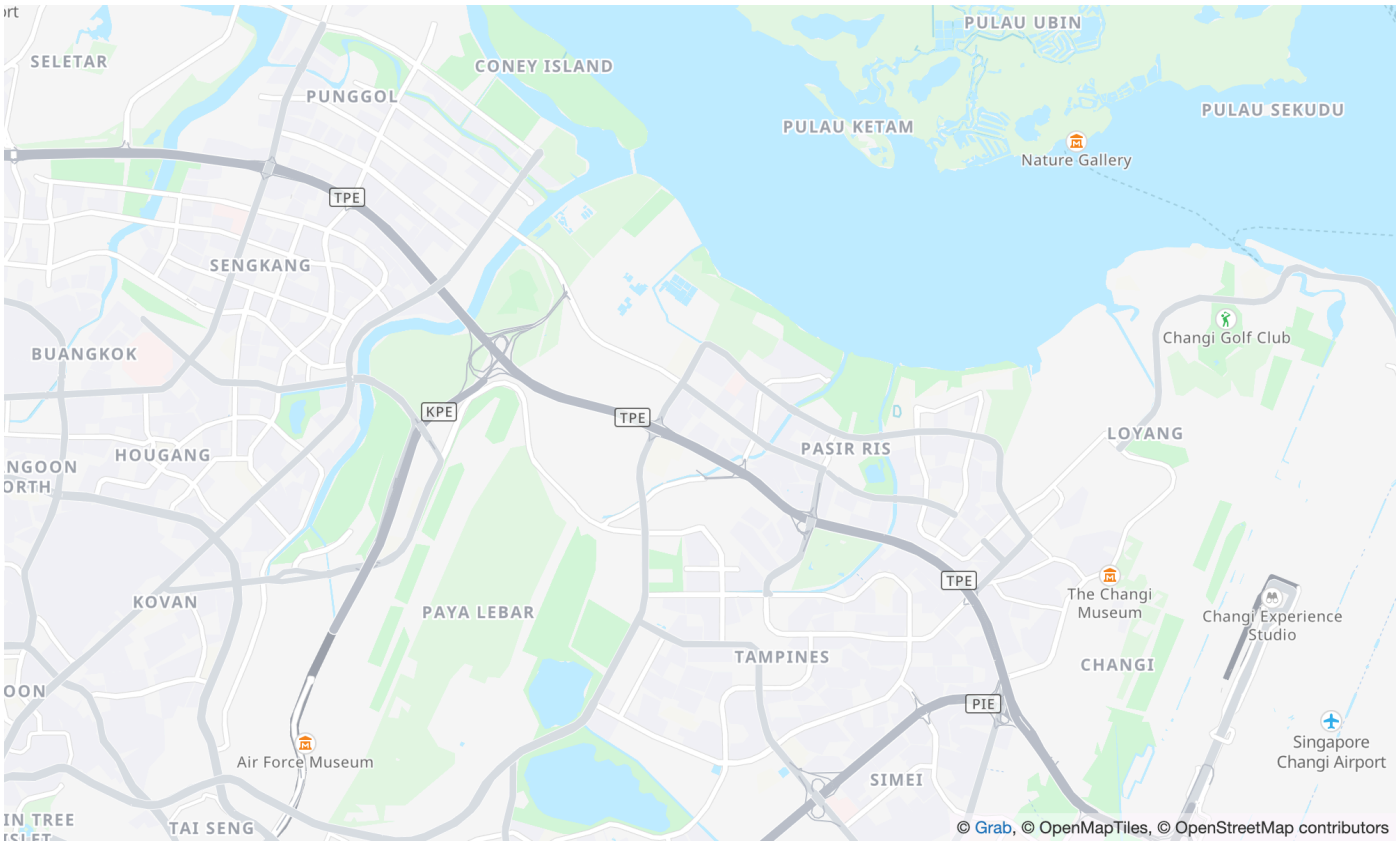
Grab マップスタイル

Amazon Location Service は、マップリソースの作成 時に次の Grab マップスタイルをサポートします。

> **Note**
>
> このセクションに記載されていない Grab マップスタイルは、現在サポートされていません。

Grab Standard Light Map

Grab Standard Light Map



マップスタイル名: `VectorGrabStandardLight`

東南アジアをカバーする詳細な土地利用の色分け、地域名、道路、ランドマーク、名所が記載された Grab の標準ベースマップ。

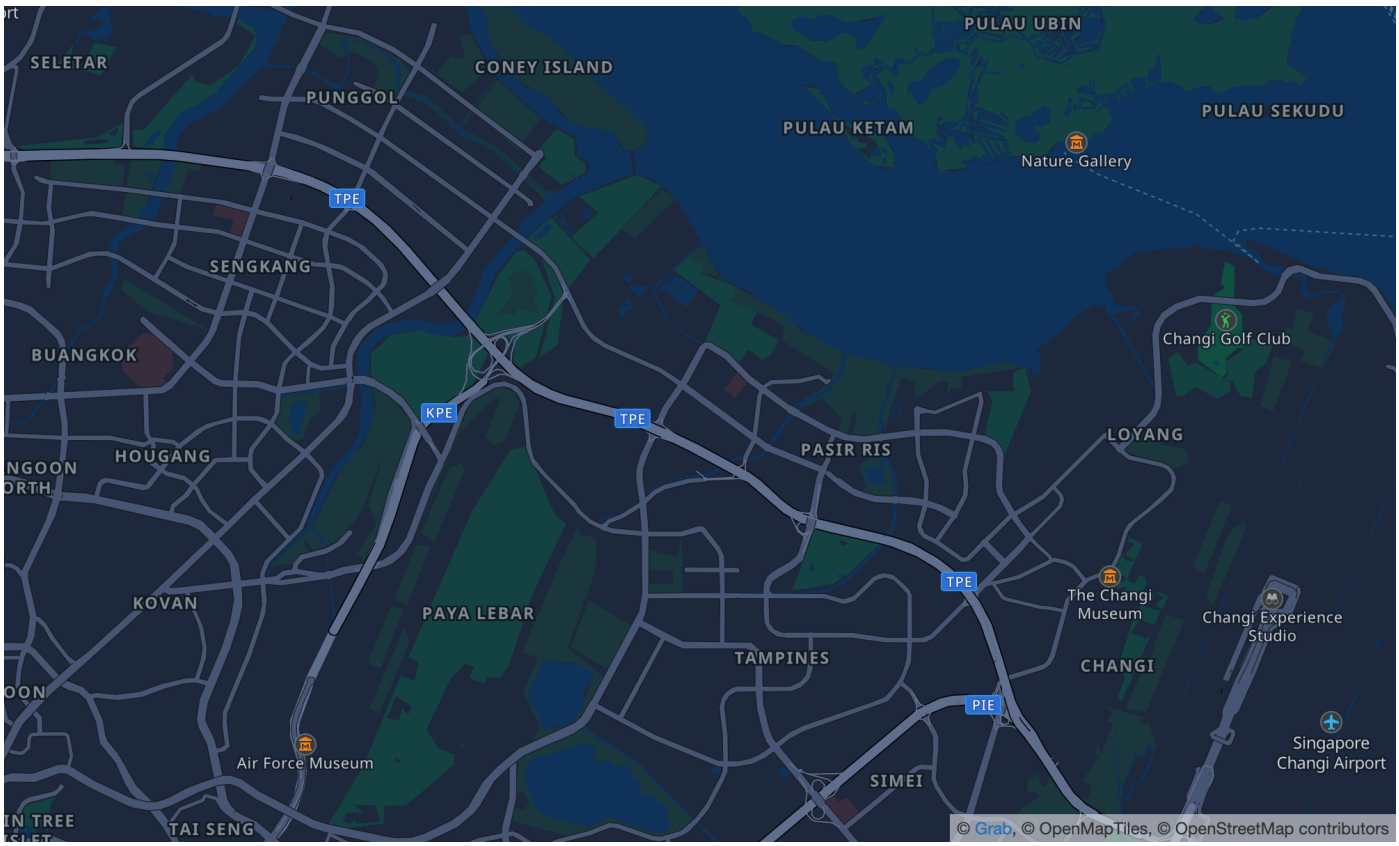
[Fonts]

Amazon Location では、GetMapGlyphs を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

- Noto Sans Regular
- Noto Sans Medium
- Noto Sans Bold

Grab Standard Dark Map

Grab Standard Dark Map



マップスタイル名: `VectorGrabStandardDark`

Grab の標準ベースマップのダークバージョン。東南アジアをカバーする詳細な土地利用の色分け、地域名、道路、ランドマーク、名所が含まれています。

[Fonts]

Amazon Location では、GetMapGlyphs を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

- Noto Sans Regular
- Noto Sans Medium
- Noto Sans Bold

カバレッジ: Grab

Grab をデータプロバイダーとして使用すると、プレースインデックスリソースの作成時にジオコーディング、リバースジオコーディング、検索のクエリをサポートしたり、ルート計算リソースを作成時にルートを計算するクエリをサポートしたりできます。

国/地域と対象地域

Grab は、東南アジアのみの地図を提供し、アジアパシフィック (シンガポール) リージョン (`ap-southeast-1`) でのみ利用できます。

Grab は、以下の国/地域の詳細データを提供しています。

- マレーシア
- フィリピン
- タイ
- シンガポール
- ベトナム
- インドネシア
- ミャンマー
- カンボジア

> **Note**
>
> これらの領域以外では、Grab をデータプロバイダーとして作成した Amazon Location Service リソースでは結果が得られません。これには検索結果やルートが含まれます。

Grab のマップは以下の範囲内にあります。

- 南 — 緯度 -21.943045533438166
- 西 — 経度 90.0

- 北 — 緯度 31.952162238024968
- 東 — 経度 146.25

ズームレベル 1–4 の場合、Grab にはグローバルカバレッジが含まれます。ズームレベル 5 以下の場合、マップタイルはこの境界ボックス内でのみ表示されます。

> **Note**
>
> この境界ボックスの外では、Grab をデータプロバイダーとして作成した Amazon Location Service のマップリソースはマップタイルを返しません。アプリケーションに 404 エラーが表示されないようにするには、を使用してマップの拡張を設定する MapLibre で説明されているように、境界ボックスを使用してマップを制限できます。

Grab ルーティング移動モード

ルーティングに関しては、Grab は前述のすべての国/地域に 車 と バイク のルーティングを提供しています。

Grab は トラック のルーティングには対応していません。

自転車 や ウォーキング ルートでは、Grab は以下の都市をサポートします。

- シンガポール
- ジャカルタ
- マニラ
- クランバレー
- バンコク
- ホーチミン市
- ハノイ

利用規約とデータアトリビューション: Grab

Grab のデータを使用する場合は、Grab および に適用されるライセンス条項を含む、適用されるすべての法的要件に従う必要があります AWS。

AWS 要件の詳細については、「AWS サービス条件」を参照してください。

GrabMapsの属性ガイドラインについては、Grab のデータ属性と利用規約のセクション 9.23 を参照してください。

GrabMaps データのエラーレポート

からのデータに問題が発生し GrabMaps、エラーや不一致を報告する場合は、 AWS テクニカルサポートにお問い合わせください。

HERE テクノロジーズ

Amazon Location Service は HERE Technologies の位置情報サービスを使用して、 AWS お客様がマップ、ジオコード、ルートの効率的な計算を行うのに役立ちます。HERE の位置データは、オープンで安全かつプライベートな位置情報中心のプラットフォームです。HERE の位置データを選択することで、 AWS クラウドにネイティブにデプロイされた、正確で新しく、堅牢なデータを選択することになります。

追加の機能情報については、Amazon Location Service データプロバイダー に関するこちらを参照してください。

トピック

- HERE マップスタイル
- カバレッジ: HERE
- 利用規約とデータアトリビューション HERE
- HERE へのエラー報告

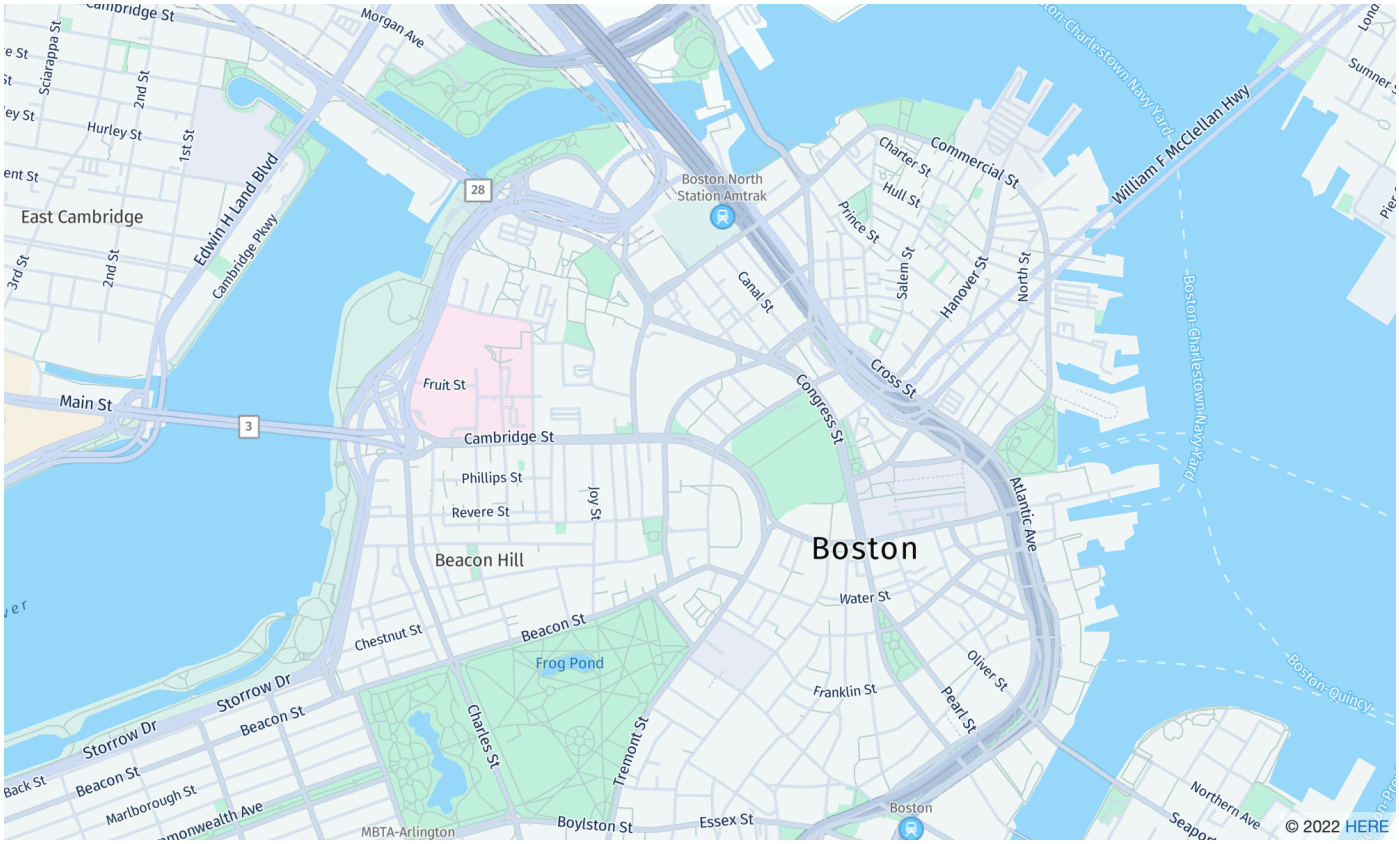
HERE マップスタイル

Amazon Location Service、マップリソースの作成 時に次の HERE マップスタイルをサポートします。

> **Note**
>
> このセクションに記載されていない HERE マップスタイルは、現在サポートされていません。

HERE Explore

HERE Explore



マップスタイル名: `VectorHereExplore`

HERE Explore

詳細で中立的な世界のベースマップ。市街地図には、高速道路、幹線道路、補助道路、鉄道、水域、都市、公園、ランドマーク、建物の敷地、行政区域が含まれます。デザインが完全に行われた日本のマップが含まれています。

[Fonts]

Amazon Location では、`GetMapGlyphs` を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

- Fira GO Italic
- Fira GO Regular
- Fira GO Bold
- Noto Sans CJK JP Light
- Noto Sans CJK JP Regular
- Noto Sans CJK JP Bold

HERE Imagery

HERE Imagery

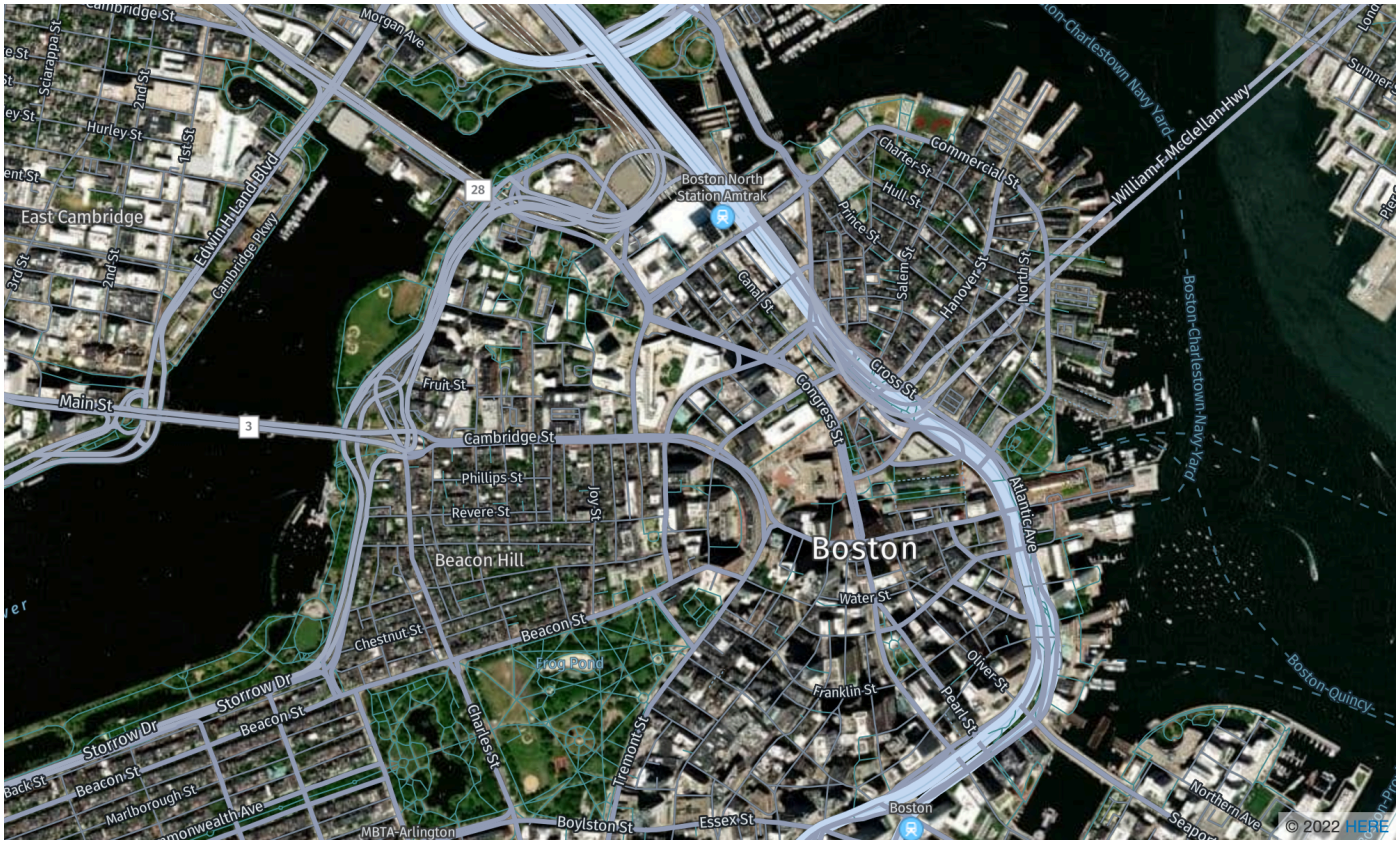
マップスタイル名: `RasterHereExploreSatellite`

HERE Imagery

HERE 画像は、全世界をカバーする高解像度の衛星画像を提供します。

HERE Hybrid

HERE Hybrid



マップスタイル名: `HybridHereExploreSatellite`

HERE Hybrid

HERE ハイブリッドスタイルでは、道路網、通りの名前、都市ラベルが衛星画像の上に表示されます。このスタイルでは、背景に衛星画像 (ラスタータイル)、上部に道路網とラベル (ベクタータイル) という 2 つのマップタイルがオーバーレイされます。このスタイルは、マップのレンダリングに必要なラスタータイルとベクタータイルの両方を自動的に取得します。

> **Note**
>
> ハイブリッドスタイルでは、表示されるマップをレンダリングするときにベクタータイルとラスタータイルの両方を使用します。つまり、ベクタータイルまたはラスタータイルのみを使用した場合よりも多くのタイルが取得されます。料金には、取得されたすべてのタイルが含まれます。

[Fonts]

Amazon Location では、GetMapGlyphs を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

- Fira GO Italic
- Fira GO Regular
- Fira GO Bold
- Noto Sans CJK JP Light
- Noto Sans CJK JP Regular
- Noto Sans CJK JP Bold

HERE Contrast (Berlin)

HERE コントラスト (ベルリン)

マップスタイル名: `VectorHereContrast`

HERE コントラスト (ベルリン)

3D と 2D レンダリングを融合させた世界の詳細な世界マップ。ハイコントラストの市街地図には、高速道路、幹線道路、補助道路、鉄道、水域、都市、公園、ランドマーク、建物の敷地、行政区域が含まれています。

[Fonts]

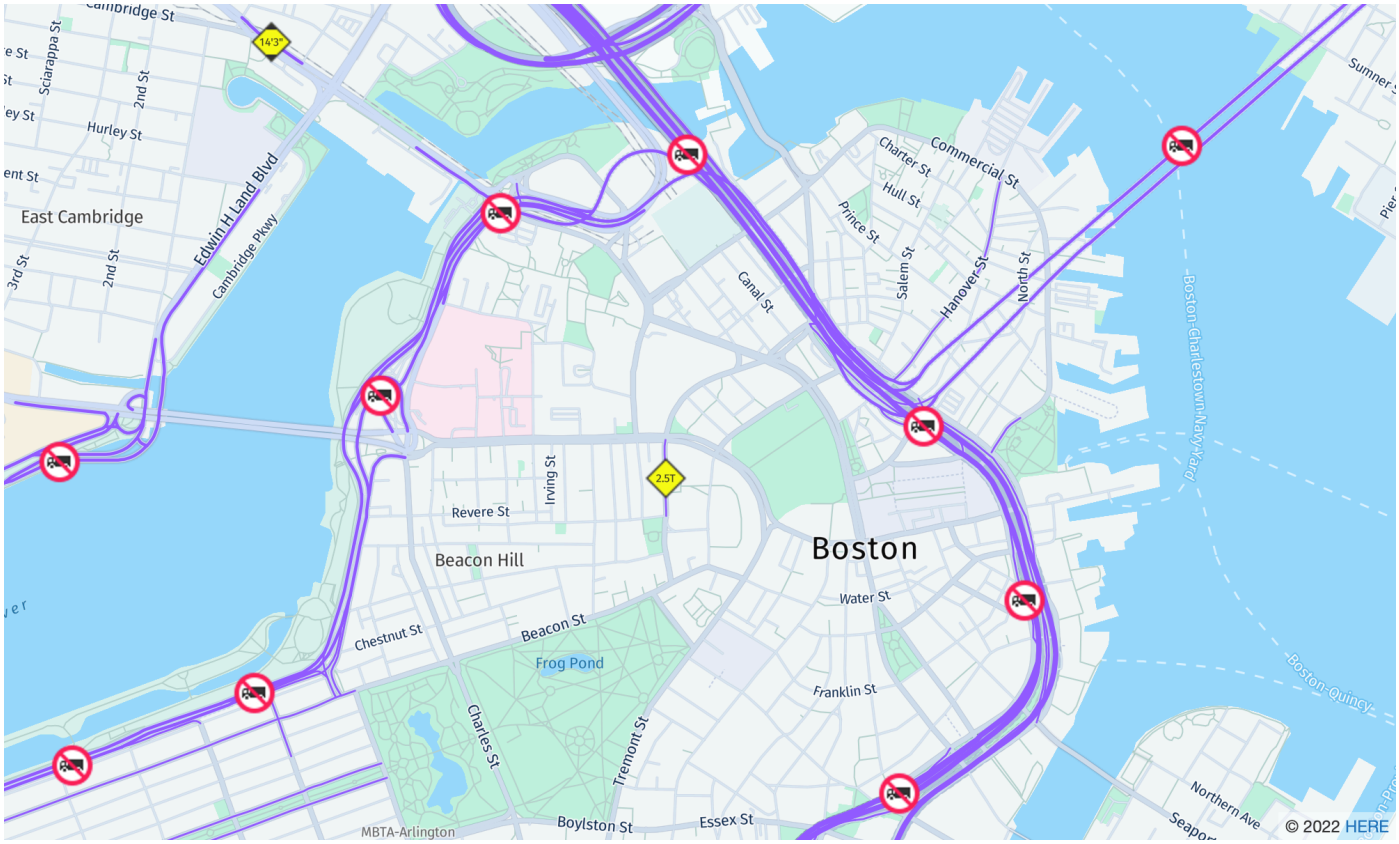
Amazon Location では、`GetMapGlyphs` を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

- Fira GO Regular
- Fira GO Bold

> **Note**
>
> このスタイルは `VectorHereBerlin` (ベルリンの地図はこちら) から改名されました。`VectorHereBerlin` は廃止されましたが、これを使用するアプリケーションでは引き続き機能します。

HERE Explore Truck

HERE Explore Truck



マップスタイル名: `VectorHereExploreTruck`

HERE Explore Truck

詳細で中立的な世界のベースマップ。HERE Explore スタイルをベースにしたストリートマップで、運送や物流のユースケースに対応します (幅/高さ/危険物など) をシンボルとアイコンでハイライト表示します。

[Fonts]

Amazon Location では、`GetMapGlyphs` を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

- Fira GO Italic
- Fira GO Regular
- Fira GO Bold
- Noto Sans CJK JP Light
- Noto Sans CJK JP Regular
- Noto Sans CJK JP Bold

世界のさまざまな地域のマップデータ品質に関する追加情報については、「HERE マップカバレッジ」をご参照ください。

カバレッジ: HERE

HERE をデータプロバイダーとして使用して、プレースインデックスリソースを作成 時にジオコーディング、リバースジオコーディング、検索のクエリをサポートしたり、ルート計算リソースを作成 時にルートを計算するクエリをサポートしたりできます。

HERE は、世界のさまざまな地域でさまざまなレベルのデータ品質を提供しています。対象地域の補償範囲の詳細については、次の内容を参照してください。

- HERE ジオコーディングカバレッジ
- HERE カールーティングカバレッジ
- HERE トラックルーティングカバレッジ

利用規約とデータアトリビューション HERE

HERE データを使用する前に、HERE および に適用されるライセンス条項など、適用されるすべての法的要件に準拠できることを確認してください AWS。ライセンス上の制限により、HERE を使用して日本国内の口ケーションのジオコーディング結果を保存することはできません。

AWS 要件の詳細については、「AWS サービス条件」を参照してください。

HERE のアトリビューションガイドラインに関する追加情報については、ロケーションおよびその他のコンテンツ に適用される HERE Technologies のサプライヤ規約のセクション 2 を参照してください。

HERE へのエラー報告

マップのエラーや不一致をここに報告するには、https://www.here.com/contact に移動して [マップエラーを報告] を選択します。

オープンデータ

Amazon Location Service、オープンデータプロバイダを介してオープンソースのマップデータへのアクセスを提供します。Open Data は、 OpenStreetMap (OSM)、Natural Earth などのオープンデータソースの夏マップ分布から構築されたグローバルベースマップを提供します。提供されているマッ

プは、物流、配送、ウェブ環境やモバイル環境でのデータ視覚化など、さまざまなアプリケーションやユースケースをサポートするように設計されています。100 万人を超える地図制作者を擁する OSM コミュニティは、1 日に何十万もの機能を更新しています。Amazon Location Service では、これらの編集を定期的に取り入れています。

追加の機能情報については、Amazon Location Service データプロバイダーの「オープンデータ」を参照してください。

トピック

- Open Data マップスタイル
- カバレッジ: Open Data
- 利用規約とデータアトリビューション: Open Data
- エラー報告とオープンデータへの寄稿

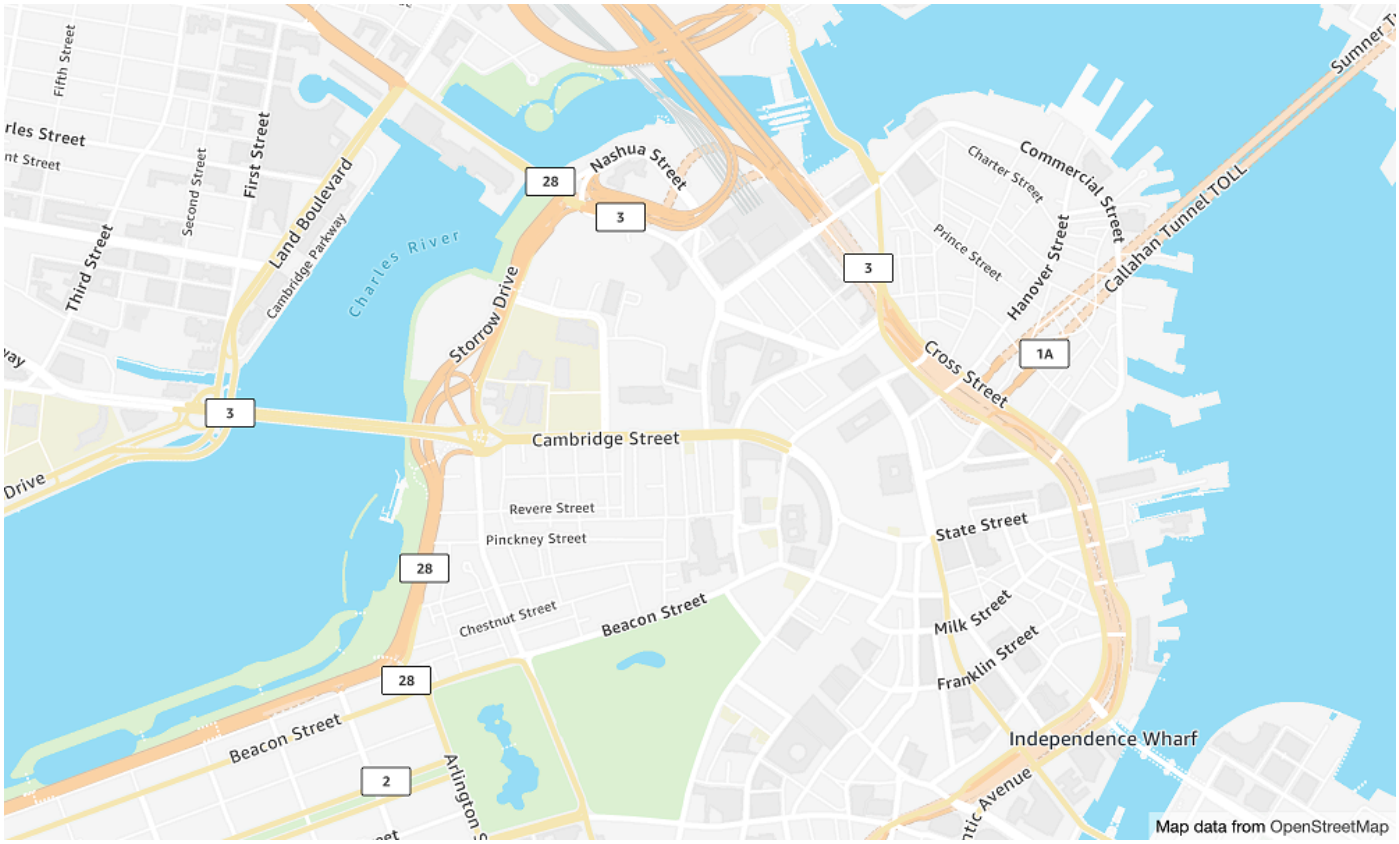
Open Data マップスタイル

Amazon Location Service は、マップリソースの作成 時に以下のマップスタイルをサポートします。

Open Data マップスタイルは代替 政治的見解 をサポートします。

Open Data Standard Light

Open Data Standard Light



マップスタイル名: `VectorOpenDataStandardLight`

これにより、ウェブサイトやモバイルアプリケーションでの使用に適した、ライトマップスタイルで世界の詳細なベースマップを提供します。これには、高速道路、主要道路、補助道路、鉄道、水域、都市、公園、ランドマーク、建物の敷地、行政区域が含まれます。

このベースマップは、 OpenStreetMap (OSM) 寄稿者からコンパイルされた OSM Daylight Map ディストリビューションに基づいています。OSM コミュニティには 180 万人以上のコントリビューターがいて、毎日 50 万以上の機能を更新しています。Amazon Location Service は、これらの編集を定期的に組み込みます。

[Fonts]

Amazon Location では、`GetMapGlyphs` を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

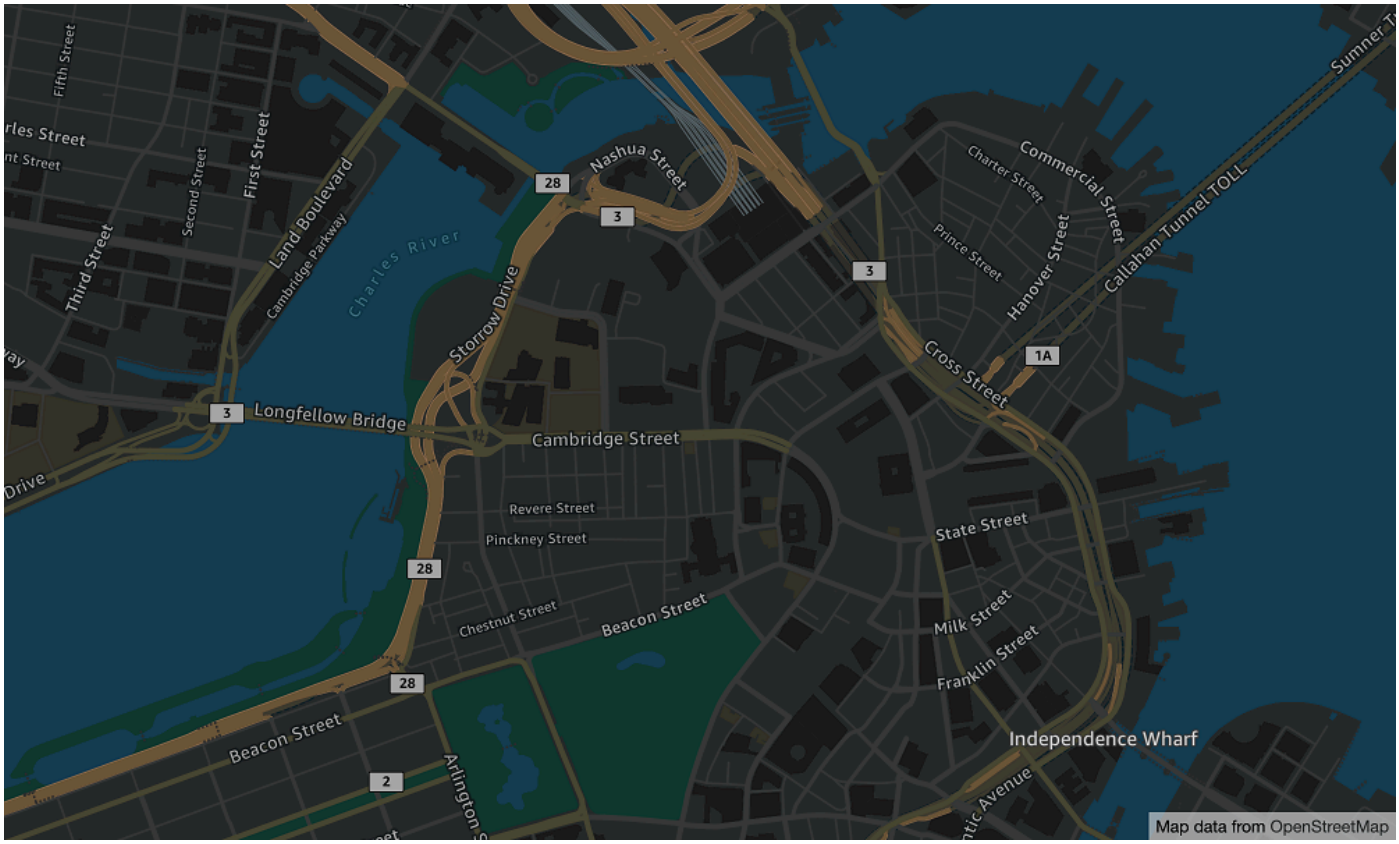
- Amazon Ember Bold、Noto Sans Bold
- Amazon Ember Condensed RC Bold、Noto Sans Bold
- Amazon Ember Condensed RC Regular、Noto Sans Regular

- Amazon Ember Medium、Noto Sans Medium
- Amazon Ember Regular Italic、Noto Sans Italic
- Amazon Ember Regular、Noto Sans Regular
- Amazon Ember Regular、Noto Sans Regular、Noto Sans Arabic Regular
- Amazon Ember Condensed RC Bold、Noto Sans Bold、Noto Sans Arabic Condensed Bold
- Amazon Ember Bold、Noto Sans Bold、Noto Sans Arabic Bold
- Amazon Ember Regular Italic、Noto Sans Italic、Noto Sans Arabic Regular
- Amazon Ember Condensed RC Regular、Noto Sans Regular、Noto Sans Arabic Condensed Regular
- Amazon Ember Medium、Noto Sans Medium、Noto Sans Arabic Medium

> **Note**
>
> `VectorOpenDataStandardLight` で使用されるフォントは、ほとんどのグリフには `Amazon Ember` を使用しますが、`Amazon Ember` でサポートされていないグリフには `Noto Sans` を使用する複合フォントです。

Open Data Standard Dark

Open Data Standard Dark



マップスタイル名: `VectorOpenDataStandardDark`

これは、ウェブサイトやモバイルアプリケーションでの使用に適した、世界中の詳細なベースマップを提供します。これには、高速道路、主要道路、補助道路、鉄道、水域、都市、公園、ランドマーク、建物の敷地、行政区域が含まれます。

このベースマップは、 OpenStreetMap (OSM) 寄稿者からコンパイルされた OSM Daylight Map ディストリビューションに基づいています。OSM コミュニティには 180 万人以上のコントリビューターがいて、毎日 50 万以上の機能を更新しています。Amazon Location Service は、これらの編集を定期的に組み込みます。

[Fonts]

Amazon Location では、`GetMapGlyphs` を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

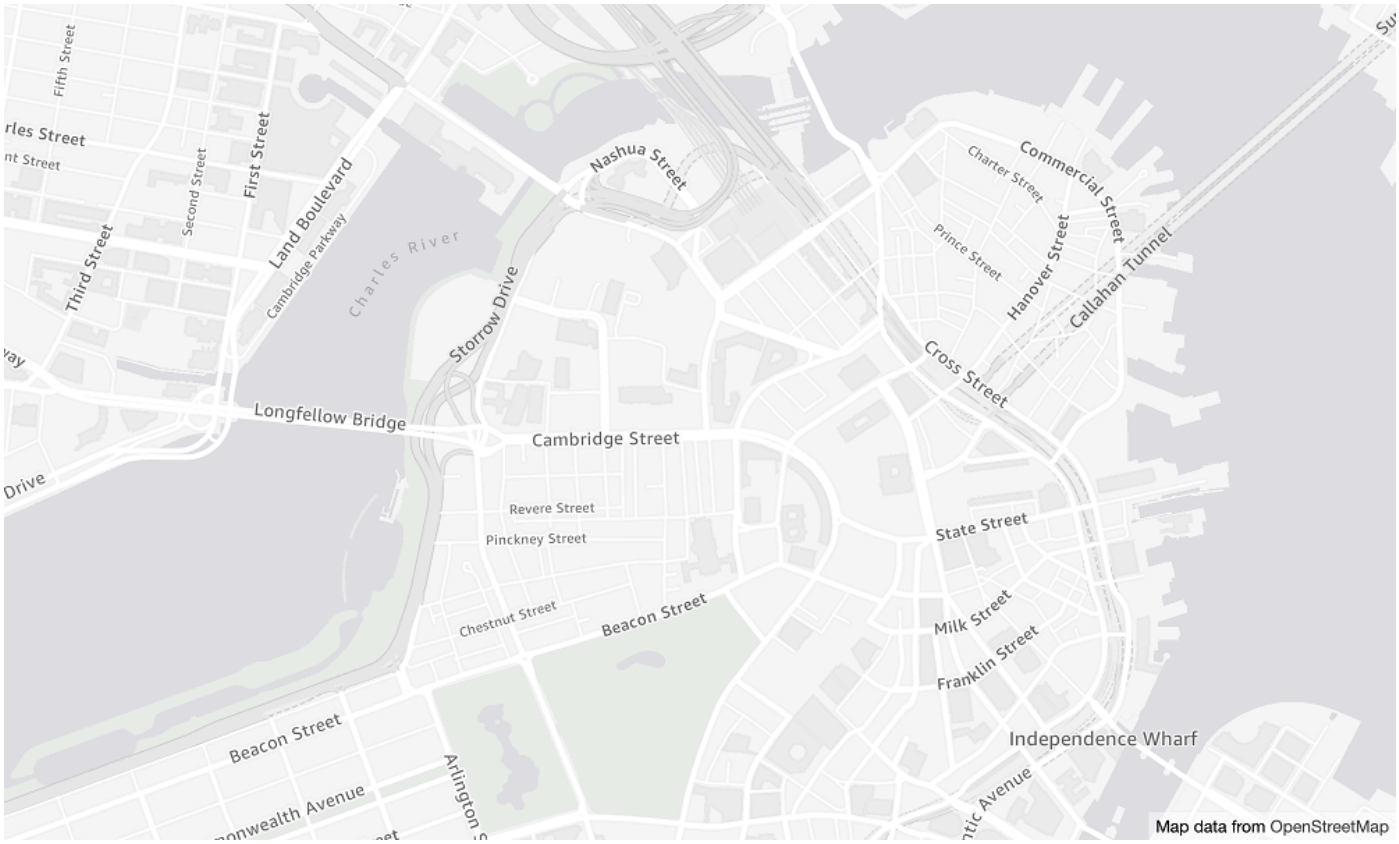
- Amazon Ember Bold、Noto Sans Bold
- Amazon Ember Condensed RC Bold、Noto Sans Bold
- Amazon Ember Condensed RC Regular、Noto Sans Regular

- Amazon Ember Medium、Noto Sans Medium
- Amazon Ember Regular Italic、Noto Sans Italic
- Amazon Ember Regular、Noto Sans Regular
- Amazon Ember Regular、Noto Sans Regular、Noto Sans Arabic Regular
- Amazon Ember Condensed RC Bold、Noto Sans Bold、Noto Sans Arabic Condensed Bold
- Amazon Ember Bold、Noto Sans Bold、Noto Sans Arabic Bold
- Amazon Ember Regular Italic、Noto Sans Italic、Noto Sans Arabic Regular
- Amazon Ember Condensed RC Regular、Noto Sans Regular、Noto Sans Arabic Condensed Regular
- Amazon Ember Medium、Noto Sans Medium、Noto Sans Arabic Medium

> **Note**
>
> `VectorOpenDataStandardDark` で使用されるフォントは、ほとんどのグリフには `Amazon Ember` を使用しますが、`Amazon Ember` でサポートされていないグリフには `Noto Sans` を使用する複合フォントです。

Open Data Visualization Light

Open Data Visualization Light



マップスタイル名: `VectorOpenDataVisualizationLight`

光をテーマにしたスタイルで、色が抑えられ、オーバーレイされたデータを理解しやすくなるような特徴が少なくなっています。

このベースマップは、 OpenStreetMap (OSM) 寄稿者からコンパイルされた OSM Daylight Map ディストリビューションに基づいています。OSM コミュニティには 180 万人以上のコントリビューターがいて、毎日 50 万以上の機能を更新しています。Amazon Location Service は、これらの編集を定期的に組み込みます。

[Fonts]

Amazon Location では、`GetMapGlyphs` を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

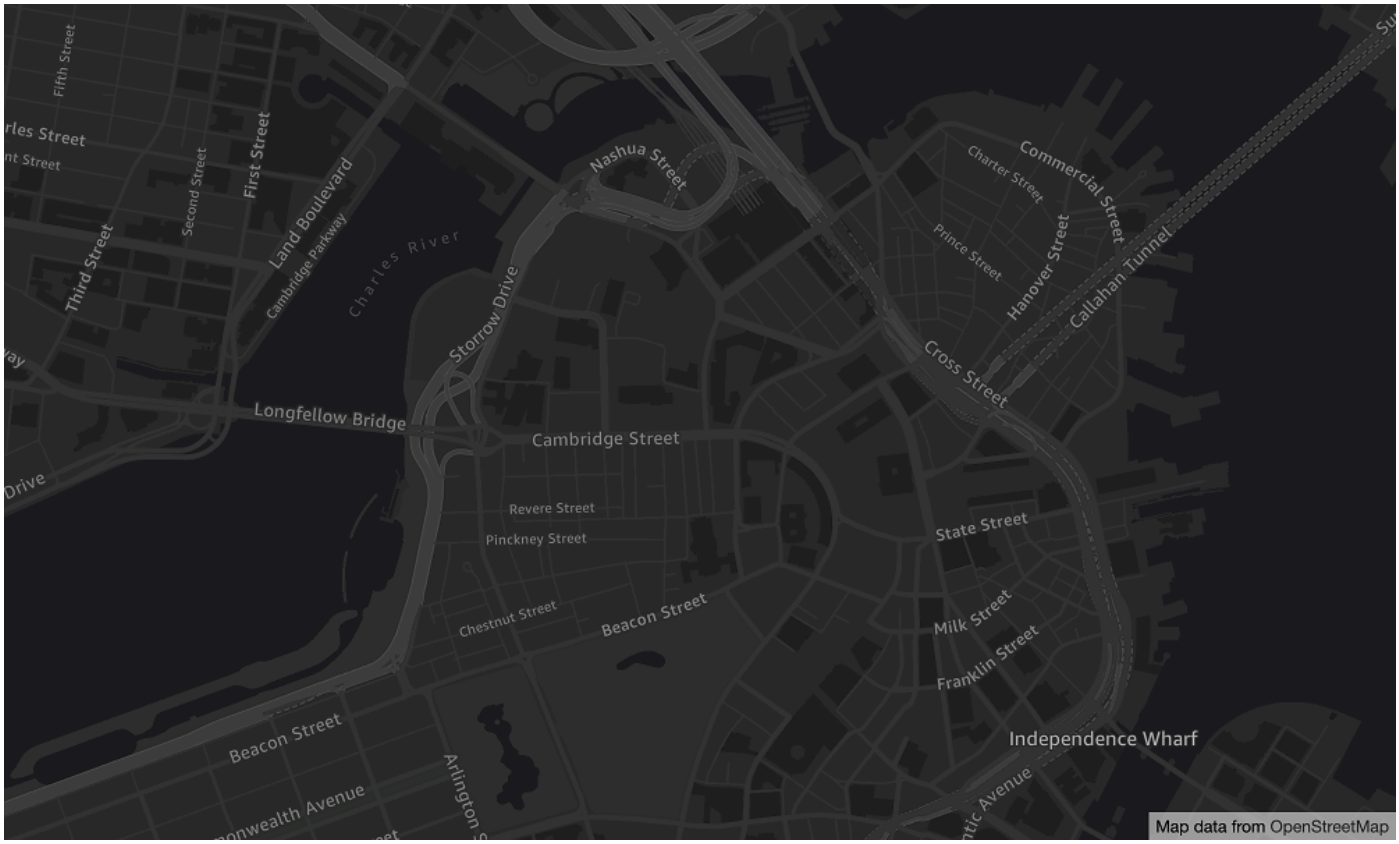
- Amazon Ember Bold、Noto Sans Bold
- Amazon Ember Condensed RC Bold、Noto Sans Bold
- Amazon Ember Condensed RC Regular、Noto Sans Regular
- Amazon Ember Medium、Noto Sans Medium

- Amazon Ember Regular Italic、Noto Sans Italic
- Amazon Ember Regular、Noto Sans Regular
- Amazon Ember Regular、Noto Sans Regular、Noto Sans Arabic Regular
- Amazon Ember Condensed RC Bold、Noto Sans Bold、Noto Sans Arabic Condensed Bold
- Amazon Ember Bold、Noto Sans Bold、Noto Sans Arabic Bold
- Amazon Ember Regular Italic、Noto Sans Italic、Noto Sans Arabic Regular
- Amazon Ember Condensed RC Regular、Noto Sans Regular、Noto Sans Arabic Condensed Regular
- Amazon Ember Medium、Noto Sans Medium、Noto Sans Arabic Medium

> **Note**
>
> `VectorOpenDataVisualizationLight` で使用されるフォントは、ほとんどのグリフには `Amazon Ember` を使用しますが、`Amazon Ember` でサポートされていないグリフには `Noto Sans` を使用する複合フォントです。

Open Data Visualization Dark

Open Data Visualization Dark



マップスタイル名: `VectorOpenDataVisualizationDark`

これは暗いテーマのスタイルで、色が落ち着いていて、オーバーレイされたデータを理解しやすい特徴が少なくなっています。

このベースマップは、 OpenStreetMap (OSM) 寄稿者からコンパイルされた OSM Daylight Map ディストリビューションに基づいています。OSM コミュニティには 180 万人以上のコントリビューターがいて、毎日 50 万以上の機能を更新しています。Amazon Location Service は、これらの編集を定期的に組み込みます。

[Fonts]

Amazon Location では、`GetMapGlyphs` を使用してフォントを提供しています。このマップで使用できるフォントスタックは次のとおりです。

- Amazon Ember Bold、Noto Sans Bold
- Amazon Ember Condensed RC Bold、Noto Sans Bold
- Amazon Ember Condensed RC Regular、Noto Sans Regular
- Amazon Ember Medium、Noto Sans Medium

- Amazon Ember Regular Italic、Noto Sans Italic
- Amazon Ember Regular、Noto Sans Regular
- Amazon Ember Regular、Noto Sans Regular、Noto Sans Arabic Regular
- Amazon Ember Condensed RC Bold、Noto Sans Bold、Noto Sans Arabic Condensed Bold
- Amazon Ember Bold、Noto Sans Bold、Noto Sans Arabic Bold
- Amazon Ember Regular Italic、Noto Sans Italic、Noto Sans Arabic Regular
- Amazon Ember Condensed RC Regular、Noto Sans Regular、Noto Sans Arabic Condensed Regular
- Amazon Ember Medium、Noto Sans Medium、Noto Sans Arabic Medium

> **Note**
>
> `VectorOpenDataVisualizationDark` で使用されるフォントは、ほとんどのグリフには `Amazon Ember` を使用しますが、`Amazon Ember` でサポートされていないグリフには `Noto Sans` を使用する複合フォントです。

カバレッジ: Open Data

オープンデータには、Amazon Location Service のマップリソース でレンダリングするための、世界中をカバーするマップが含まれています。

> **Note**
>
> オープンデータは Amazon Location Service マップリソースでのみ使用されます。ジオコーディング、リバースジオコーディング、検索のクエリをサポートしたり、ルートを計算するクエリをサポートしたりするデータプロバイダーとしてオープンデータを使用することはできません。

利用規約とデータアトリビューション: Open Data

オープンデータを使用する前に、オープンデータと に適用されるライセンス条項を含む、適用されるすべての法的要件に準拠できることを確認してください AWS。

AWS 要件の詳細については、「AWS サービス条件」を参照してください。

オープンデータ属性ガイドラインの詳細については、 OpenStreetMap「」の著作権とライセンス「」および OpenStreetMap「ライセンス/属性ガイドライン」を参照してください。

エラー報告とオープンデータへの寄稿

OpenStreetMap (OSM) と Natural Earth はコミュニティ主導のオープンデータプロジェクトです。データに問題が発生した場合は、エラーを報告したり、修正や提案を直接提供したりできます。

- OSM でエラーを報告したり、提案をしたりするには、地図上にメモを作成できます。これは投稿者が地図を修正する際に役立つ地図上のコメントです。OpenStreetMap ウェブサイト からメモを作成します。メモの詳細については、OpenStreetMap Wiki の「メモ」を参照してください。
- ロケーションの追加やエラーの修正など OpenStreetMap、 に直接寄与する方法の詳細については、OpenStreetMap Wiki の「マップデータの提供」を参照してください。
- Natural Earth 内のデータの修正リクエストを送信するには、Natural Earth のウェブサイト から問題を提出してください。

> **Note**
>
> でエラーを修正するとすぐに発生する OpenStreetMap 可能性がありますが、Open Data プロバイダーが使用する OSM データの夏マップ分布に修正が表示されるまでに時間がかかる場合があります。このプロセスに関する詳しい情報は、Daylight Map Distribution のウェブサイトに掲載されています。さらに、Amazon Location Service は、Amazon Location Service で使用される地図データを約毎月更新します。

データプロバイダー別の機能

このセクションでは、Amazon Location Service で利用できる機能を、データプロバイダーごとに分類して説明します。

次の表に、機能の大まかな概要を示します。

データプロバイダー	地理カバレッジエリア	機能のカバレッジ	AWS リージョン
Esri	グローバル	マップ、場所、ルート	Amazon Location が利用可能なすべてのリージョン。
Grab	東南アジア	マップ、場所、ルート	アジアパシフィック (シンガポール)、`ap-southeast-1` のみ。
HERE	グローバル	マップ、場所、ルート	Amazon Location が利用可能なすべてのリージョン。
オープンデータ	グローバル	マップ	Amazon Location が利用可能なすべてのリージョン。

以下のタブには、各機能エリア内の詳細が表示されます。

Map Features

次の表に、データプロバイダー別のマップ機能を示します。マップの概念の詳細については、「マップ」を参照してください。

データプロバイダー	対応しているマップタイプ	ベクトルズームレベル	ラスタズームレベル
Esri	ベクトル ラスター (画像) 詳細については、「Esri マップスタイル」を参照してください。	0-15	0-23

データプロバイダー	対応しているマップタイプ	ベクトルズームレベル	ラスタズームレベル
Grab	ベクトル (東南アジア のみ) 詳細については、「Grab マップスタイル」を参照してください。	0-14	なし
HERE	ベクトル ラスター (画像) ハイブリッド 詳細については、「HERE マップスタイル」を参照してください。	1-17	0-19
オープンデータ	ベクトル 詳細については、「Open Data マップスタイル」を参照してください。	0-15	なし

> **Note**
>
> ズームレベルは、各プロバイダーの API で定義されている最大設定と最小設定を表します。マップのエリアによって最大値が異なる場合があります。例えば、海タイルは主要都市のエリアよりも詳細なズームレベルが少ない場合があります。
> MapLibre (およびその他のマップレンダリングエンジン) では、最小ズームレベルと最大ズームレベルを設定でき、エリア内のデータプロバイダーのズームレベルも優先されるため、これらの不一致を処理するコードを記述する必要はありません。

Places and Search

次の表に、データプロバイダー別の場所と検索機能を示します。場所の概念の詳細については、「場所検索」を参照してください。

データプロバイダー	ジオコーディング	リバースジオコーディング	自動入力	GetPlace
Esri	以下を除くすべての機能: PlaceId	以下を除くすべての機能: TimeZone PlaceId	すべての機能	すべての機能
Grab	以下を除くすべての機能: unit type カテゴリはサポートされていません	すべての機能	すべての機能	以下を除くすべての機能: unit type SubMunicipality
HERE	以下を除くすべての機能: unit number unit type relevance フィルタリングに関する追加の制限事項	すべての機能	すべての機能	以下を除くすべての機能: unit number unit type SubMunicipality
オープンデータ	サポートされません	サポートされません	サポートされません	以下のみがサポートされています。

データプロバイダー	ジオコーディング	リバースジオコーディング	自動入力	GetPlace
				SubMunicipality

Route features

次の表に、データプロバイダー別のルート機能を示します。ルートの概念の詳細については、「ルート」を参照してください。ルートマトリックスの制限の詳細については、「出発地と目的地のポジションの制限」を参照してください。

データプロバイダー	移動モード	ルートを計算する	ルートマトリックス
Esri	車、トラック、ウォーキング	出発地と目的地は互いに 400 km 以内である必要があります。合計移動時間は 400 分を超えてはいけません。 ArrivalTime はサポートされていません。	出発地と目的地の位置は最大 10 です。 韓国ではサポートされていません。 出発地と目的地のペアはすべて 400 km 以内の距離でなければなりません。
Grab	車、オートバイ。 特定の都市 でのウォーキングとサイクリング。	距離制限なし。	出発地と目的地の位置は最大 350 です。
HERE	車、トラック、ウォーキング	距離制限なし。出発地と目的地の位置を囲む円の外側を 10 km 以上進むルートは計算されません。	出発地と目的地の位置は最大 350 です。 出発地と目的地の位置はすべて 180 km 圏内になければなりません。

データプロバイダー	移動モード	ルートを計算する	ルートマトリックス
			その他の制限 はありますが、より長いルートにも対応しています。
オープンデータ	サポートされません	サポートされません	サポートされません

データプロバイダー向けの利用規約とデータアトリビューション

データプロバイダーを使用する前に、プロバイダーの使用に適用されるライセンス条項など、適用される法的要件に準拠していることを確認してください。

AWS 要件の詳細については、「AWS サービス条件」を参照してください。

アプリケーションまたはドキュメントに Amazon Location リソースとともにデータプロバイダーを使用するときは、使用する各データプロバイダーの帰属情報を必ず提供してください。

各データプロバイダーのコンプライアンスとアトリビューションの詳細については、次のトピックを参照してください。

- Esri – 利用規約とデータアトリビューション: Esri
- Grab – 利用規約とデータアトリビューション: Grab
- HERE – 利用規約とデータアトリビューション HERE
- オープンデータ – 利用規約とデータアトリビューション: Open Data

Amazon Location のリージョンとエンドポイント

Amazon Location は、以下の AWS リージョンで利用できます。

リージョン

リージョン名	リージョン	エンドポイント	プロトコル	
米国東部 (オハイオ)	us-east-2	geo.us-east-2.amazonaws.com	HTTPS	
米国東部 (バージニア北部)	us-east-1	geo.us-east-1.amazonaws.com	HTTPS	
米国西部 (オレゴン)	us-west-2	geo.us-west-2.amazonaws.com	HTTPS	
アジアパシフィック (ムンバイ)	ap-south-1	geo.ap-south-1.amazonaws.com	HTTPS	
アジアパシフィック (シンガポール)	ap-southeast-1	geo.ap-southeast-1.amazonaws.com	HTTPS	
アジアパシフィック (シドニー)	ap-southeast-2	geo.ap-southeast-2.amazonaws.com	HTTPS	
アジアパシフィック (東京)	ap-northeast-1	geo.ap-northeast-1.amazonaws.com	HTTPS	
カナダ (中部)	ca-central-1	geo.ca-central-1.amazonaws.com	HTTPS	

リージョン名	リージョン	エンドポイント	プロトコル	
欧州 (フランクフルト)	eu-central-1	geo.eu-central-1.amazonaws.com	HTTPS	
欧州 (アイルランド)	eu-west-1	geo.eu-west-1.amazonaws.com	HTTPS	
欧州 (ロンドン)	eu-west-2	geo.eu-west-2.amazonaws.com	HTTPS	
欧州 (ストックホルム)	eu-north-1	geo.eu-north-1.amazonaws.com	HTTPS	
南米 (サンパウロ)	sa-east-1	geo.sa-east-1.amazonaws.com	HTTPS	
AWS GovCloud (米国西部)	us-gov-west-1	geo.us-gov-west-1.amazonaws.com geo-fips.us-gov-west-1.amazonaws.com	HTTPS HTTPS	

> **Note**
>
> この表のエンドポイントの使用方法の詳細については、次のセクションを参照してください。

エンドポイント

Amazon Location リージョンエンドポイントの一般的な構文は次のとおりです。

```
protocol://service-code.geo.region-code.amazonaws.com
```

この構文内で、Amazon Location は次のサービスコードを使用します。

サービス	サービスコード
Amazon Location Maps	マップ
Amazon Location の場所	場所
Amazon Location Geofences	ジオフェンシング
Amazon Location トラッカー	追跡
Amazon Location Routes	ルート

例えば、米国東部 (バージニア北部) の Amazon Location Maps の地域エンドポイントは、https://*maps*.geo.*us-east-1*.amazonaws.com になります。

API オペレーションエンドポイント

Amazon Location Service コントロールプレーンエンドポイントの構文は次のとおりです。

```
protocol://cp.service-code.geo.region-code.amazonaws.com
```

Amazon Location Service のコントロールプレーンアクションは次のとおりです。

サービス	エンドポイント	API オペレーション
Amazon Location Maps	https://cp.maps.geo.*region*.amazonaws.com	CreateMap DeleteMap DescribeMap ListMaps UpdateMap
Amazon Location の場所	https://cp.places.geo.*region*.amazonaws.com	CreatePlaceIndex DeletePlaceIndex

サービス	エンドポイント	API オペレーション
		DescribePlaceIndex ListPlaceIndexes UpdatePlaceIndex
Amazon Location Geofences	https://cp.geofencing.geo.*region*.amazonaws.com	CreateGeofenceCollection DeleteGeofenceCollection DescribeGeofenceCollection ListGeofenceCollections UpdateGeofenceCollection
Amazon Location トラッカー	https://cp.tracking.geo.*region*.amazonaws.com	CreateTracker DeleteTracker DescribeTracker UpdateTracker ListTrackers AssociateTrackerConsumer DisassociateTrackerConsumer ListTrackerConsumers
Amazon Location Routes	https://cp.routes.geo.*region*.amazonaws.com	CreateRouteCalculator DeleteRouteCalculator DescribeRouteCalculator ListRouteCalculators UpdateRouteCalculator

サービス	エンドポイント	API オペレーション
Amazon Location Metadata	https://cp.metadata.geo.*region*.amazonaws.com	CreateKey DeleteKey DescribeKey ListKeys UpdateKey

Amazon Location Service のサービスクォータ

このトピックでは、Amazon Location Service レート制限とクォータの概要を説明します。

> **Note**
>
> より高いクォータが必要な場合は、Service Quotas コンソールを使用して、調整可能なクォータのクォータ増加のリクエストを行うことができます。クォータの引き上げをリクエストする場合は、クォータの引き上げが必要なリージョンを選択します。ほとんどのクォータは AWS リージョンに固有であるためです。

Service Quotas は、 AWS アカウントおよび AWS リージョンごとに保持できるリソースの最大数です。Amazon Location Service、Service Quotas を超える追加のリクエストを拒否します。

レート制限 (Rate of... で始まるクォータ) は 1 秒あたりの最大リクエスト数で、各 API オペレーションに定義されている 1 秒間の上限の 80% をバーストレートとします。Amazon Location Service、オペレーションのレート制限を超えるリクエストを制限します。

名前	デフォルト	引き上げ可能	説明
アカウントごとの API キーリソース	サポートされている各リージョン：500	いいえ	各アカウントで持つことができる API キーリソース(アクティブまたは期限切れ) の最大数。
アカウントあたりの Geofence Collection リソース	サポートされている各リージョン：1,500	はい	アカウントあたりで作成できる Geofence Collection リソースの最大数。
Geofence Collection あたりのジオフェンス	サポートされている各リージョン：50,000	いいえ	Geofence Collection あたりで作成できるジオフェンスの最大数。
アカウントごとの Map リソース	サポートされている各リージョン：40	はい	アカウントごとに作成できる Map リソースの最大数。
アカウントごとのプレースインデックスリソース	サポートされている各リージョン：40	はい	アカウントあたりで作成できる Place Index リソースの最大数。
AssociateTrackerConsumer API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる AssociateTrackerConsumer リクエストの最大数。追加のリクエストは調整されます。
BatchDeleteDevicePositionHistory API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる BatchDeleteDevicePositionHistory リクエスト

名前	デフォルト	引き上げ可能	説明
			の最大数。追加のリクエストは調整されます。
BatchDeleteGeofence API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる BatchDeleteGeofence リクエストの最大数。追加のリクエストは調整されます。
BatchEvaluateGeofences API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる BatchEvaluateGeofences リクエストの最大数。追加のリクエストは調整されます。
BatchGetDevicePosition API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる BatchGetDevicePosition リクエストの最大数。追加のリクエストは調整されます。
BatchPutGeofence API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる BatchPutGeofence リクエストの最大数。追加のリクエストは調整されます。

名前	デフォルト	引き上げ可能	説明
BatchUpdateDevicePosition API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる BatchUpdateDevicePosition リクエストの最大数。追加のリクエストは調整されます。
CalculateRoute API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる CalculateRoute リクエストの最大数。追加のリクエストは調整されます。
CalculateRouteMatrix API リクエストのレート	サポートされている各リージョン: 5/秒	はい	1 秒あたりに実行できる CalculateRouteMatrix リクエストの最大数。追加のリクエストは調整されます。
CreateGeofenceCollection API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる CreateGeofenceCollection リクエストの最大数。追加のリクエストは調整されます。
CreateKey API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる CreateKey リクエストの最大数。追加のリクエストは調整されます。

名前	デフォルト	引き上げ可能	説明
CreateMap API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる CreateMap リクエストの最大数。追加のリクエストは調整されます。
CreatePlaceIndex API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる CreatePlaceIndex リクエストの最大数。追加のリクエストは調整されます。
CreateRouteCalculator API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる CreateRouteCalculator リクエストの最大数。追加のリクエストは調整されます。
CreateTracker API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる CreateTracker リクエストの最大数。追加のリクエストは調整されます。
DeleteGeofenceCollection API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる DeleteGeofenceCollection リクエストの最大数。追加のリクエストは調整されます。

名前	デフォルト	引き上げ可能	説明
DeleteKey API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる DeleteKey リクエストの最大数。追加のリクエストは調整されます。
DeleteMap API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる DeleteMap リクエストの最大数。追加のリクエストは調整されます。
DeletePlaceIndex API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる DeletePlaceIndex リクエストの最大数。追加のリクエストは調整されます。
DeleteRouteCalculator API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる DeleteRouteCalculator リクエストの最大数。追加のリクエストは調整されます。
DeleteTracker API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる DeleteTracker リクエストの最大数。追加のリクエストは調整されます。

名前	デフォルト	引き上げ可能	説明
DescribeGeofenceCollection API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる DescribeGeofenceCollection リクエストの最大数。追加のリクエストは調整されます。
DescribeKey API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる DescribeKey リクエストの最大数。追加のリクエストは調整されます。
DescribeMap API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる DescribeMap リクエストの最大数。追加のリクエストは調整されます。
DescribePlaceIndex API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる DescribePlaceIndex リクエストの最大数。追加のリクエストは調整されます。
DescribeRouteCalculator API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる DescribeRouteCalculator リクエストの最大数。追加のリクエストは調整されます。

名前	デフォルト	引き上げ可能	説明
DescribeTracker API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる DescribeTracker リクエストの最大数。追加のリクエストは調整されます。
DisassociateTrackerConsumer API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる DisassociateTrackerConsumer リクエストの最大数。追加のリクエストは調整されます。
ForecastGeofenceEvents API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる ForecastGeofenceEvents リクエストの最大数。追加のリクエストは調整されます。
GetDevicePosition API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる GetDevicePosition リクエストの最大数。追加のリクエストは調整されます。
GetDevicePositionHistory API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる GetDevicePositionHistory リクエストの最大数。追加のリクエストは調整されます。

名前	デフォルト	引き上げ可能	説明
GetGeofence API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる GetGeofence リクエストの最大数。追加のリクエストは調整されます。
GetMapGlyphs API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる GetMapGlyphs リクエストの最大数。追加のリクエストは調整されます。
GetMapSprites API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる GetMapSprites リクエストの最大数。追加のリクエストは調整されます。
GetMapStyleDescriptor API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる GetMapStyleDescriptor リクエストの最大数。追加のリクエストは調整されます。
GetMapTile API リクエストのレート	サポートされている各リージョン: 500/秒	はい	1 秒あたりに実行できる GetMapTile リクエストの最大数。追加のリクエストは調整されます。
GetPlace API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる GetPlace リクエストの最大数。追加のリクエストは調整されます。

名前	デフォルト	引き上げ可能	説明
ListDevicePositions API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる ListDevicePositions リクエストの最大数。追加のリクエストは調整されます。
ListGeofenceCollections API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる ListGeofenceCollections リクエストの最大数。追加のリクエストは調整されます。
ListGeofences API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる ListGeofences リクエストの最大数。追加のリクエストは調整されます。
ListKeys API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる ListKeys リクエストの最大数。追加のリクエストは調整されます。
ListMaps API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる ListMaps リクエストの最大数。追加のリクエストは調整されます。

名前	デフォルト	引き上げ可能	説明
ListPlaceIndexes API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる ListPlaceIndexes リクエストの最大数。追加のリクエストは調整されます。
ListRouteCalculators API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる ListRouteCalculators リクエストの最大数。追加のリクエストは調整されます。
ListTagsForResource API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる ListTagsForResource リクエストの最大数。追加のリクエストは調整されます。
ListTrackerConsumers API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる ListTrackerConsumers リクエストの最大数。追加のリクエストは調整されます。
ListTrackers API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる ListTrackers リクエストの最大数。追加のリクエストは調整されます。

名前	デフォルト	引き上げ可能	説明
PutGeofence API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる PutGeofence リクエストの最大数。追加のリクエストは調整されます。
SearchPlaceIndexForPosition API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる SearchPlaceIndexForPosition リクエストの最大数。追加のリクエストは調整されます。
SearchPlaceIndexForSuggestions API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる SearchPlaceIndexForSuggestions リクエストの最大数。追加のリクエストは調整されます。
SearchPlaceIndexForText API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる SearchPlaceIndexForText リクエストの最大数。追加のリクエストは調整されます。
TagResource API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる TagResource リクエストの最大数。追加のリクエストは調整されます。

名前	デフォルト	引き上げ可能	説明
UntagResource API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる UntagResource リクエストの最大数。追加のリクエストは調整されます。
UpdateGeofenceCollection API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる UpdateGeofenceCollection リクエストの最大数。追加のリクエストは調整されます。
UpdateKey API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる UpdateKey リクエストの最大数。追加のリクエストは調整されます。
UpdateMap API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる UpdateMap リクエストの最大数。追加のリクエストは調整されます。
UpdatePlaceIndex API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる UpdatePlaceIndex リクエストの最大数。追加のリクエストは調整されます。

名前	デフォルト	引き上げ可能	説明
UpdateRouteCalculator API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる UpdateRouteCalculator リクエストの最大数。追加のリクエストは調整されます。
UpdateTracker API リクエストのレート	サポートされている各リージョン：10/秒	はい	1 秒あたりに実行できる UpdateTracker リクエストの最大数。追加のリクエストは調整されます。
VerifyDevicePosition API リクエストのレート	サポートされている各リージョン：50/秒	はい	1 秒あたりに実行できる VerifyDevicePosition リクエストの最大数。追加のリクエストは調整されます。
アカウントあたりの計算ツールリソースのレート	サポートされている各リージョン：40	はい	アカウントごとに作成できるルート計算ツールリソースの最大数。
トラッカーあたりの Tracker コンシューマー	サポートされている各リージョン：5	いいえ	Tracker リソースを関連付けることができる Geofence Collection の最大数。
アカウントあたりの Tracker リソース	サポートされている各リージョン：500	はい	アカウントあたりで作成できる Tracker リソースの最大数。

> **Note**
>
> Cloudwatch を使用すると、クォータに対して使用状況を監視できます。詳細については、「CloudWatch を使用したクォータに対する使用状況のモニタリング」を参照してください。

Amazon Location Service のサービスクォータを管理する

Amazon Location Service は、Service Quotas と統合されています。Service Quotas は、クォータを一元的に表示および管理できる AWS サービスです。詳細については、「Service Quotas ユーザーガイド」の「Service Quotas とは」を参照してください。

Service Quotas を使用すると、Amazon Location service のサービスクォータの値を簡単に調べることができます。

AWS Management Console

コンソールを使用して Amazon Location Service のサービスクォータを表示するには

1. https://console.aws.amazon.com/servicequotas/ で Service Quotas コンソールを開きます。
2. ナビゲーションペインで、[AWS サービス] を選択します。
3. AWS サービス リストから「Amazon Location」を検索して選択します。

   [Service Quotas] の一覧には、サービスクォータ名、適用された値 (使用可能な場合)、 AWS デフォルトのクォータ、クォータ値が調整可能かどうかが表示されます。
4. 説明など、Service Quotas に関する追加情報を表示するには、クォータ名を選択します。
5. (オプション) クォータの引き上げをリクエストするには、[Request quota increase (クォータ引き上げリクエスト)] を選択、または必要な情報を入力または選択して、[Request (リクエスト)] を選択します。

コンソールを使用してさらにサービスクォータの操作を行うには、Service Quotas ユーザーガイドを参照してください。クォータの引き上げをリクエストするには、「Service Quotas ユーザーガイド」の「クォータ引き上げリクエスト」を参照してください。

AWS CLI

AWS CLIを使用して Amazon Location service のサービスクォータを表示するには

次のコマンドを実行して、デフォルトの Amazon Location のクォータを表示します。

```
aws service-quotas list-aws-default-service-quotas \
    --query 'Quotas[*].
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
    --service-code geo \
    --output table
```

を使用してサービスクォータをさらに操作するには AWS CLI、Service Quotas AWS CLI コマンドリファレンス」を参照してください。クォータの引き上げをリクエストするには、「AWS CLI コマンド リファレンス」で `request-service-quota-increase` コマンドを参照してください。

Amazon Location Service 開発者としての利用を開始する

Amazon Location Service を使用すると、バックエンドのウェブサービス、ウェブアプリケーション、モバイルアプリケーションなど、さまざまなフォームファクターやシステムのアプリケーションに地理関連の機能を提供できます。SDK、ライブラリ、コード例など、アプリケーションの構築に役立つツールが多数用意されています。

このセクションでは、Amazon Location の使用を開始する上で役立つ情報とリンクを提供します。特に、以下のトピックでは、有用な情報を提供しています。

- シナリオとユースケース — 開発シナリオのリストと、Amazon Location Service でそれらをどのように実現できるかを記載しています。
- Amazon Location SDK とツール — Amazon Locationを使ったプログラミングの際に役立つ Software Development Kit (SDK) とライブラリを紹介しています。
- Amazon Location Service API リファレンス — AWS SDK に付属するコア Amazon Location APIs へのリファレンス。
- コード例 — 使用を開始したり、既存のアプリケーションに機能を追加したりするのに役立つサンプルを提供しています。
- クイックスタートチュートリアル — 最初のアプリケーションの作成方法を解説しています。このチュートリアルには、Web アプリケーション作成用と Android ベースのモバイルアプリケーション作成用のバージョンがあります。
- Amazon Location Service の概念 — このセクションでは、マップ、場所検索、ルート、ジオフェンスとトラッカーを含む、Amazon Locationの基本概念について説明しています。
- Amplify — Amplifyは、 AWS クラウドを使用してWebおよびモバイルアプリケーションを作成するために必要な機能の多くをカプセル化した完全なソリューションです。すでに Amplify を使用している場合、またはこれから Amplify を使用する場合は、Amazon Location Service に組み込まれているジオライブラリが利利用可能です。Amplify Geo の使用を開始するには、こちらのドキュメントを参照してください。

シナリオとユースケース

Amazon Location Serviceは、 AWS クラウドで提供されているサービスです。クラウド内の独自の Amazon EC2 インスタンスから呼び出すこともできますが、多くのマッピングアプリケーションはデバイス、またはデバイスとクラウドを組み合わせた環境で実行されます。以下に、いくつかの一般的なシナリオと、その開発方法を示します。

- フリートのドライバーのルートを最適化するバックエンドアプリケーション。

  Amazon Location Service AWS クラウド を使用して、フリートのルートオプティマイザへの入力としてルートマトリックスを計算するアプリケーションを で Amazon Amazon EC2 で実行します。AWS SDK を使用して、Amazon Location を呼び出します。

- 顧客がビジネスの場所を検索できるようにするウェブアプリケーション。

  ロケーションベースのアプリケーションを含め、Amazon EC2 インスタンスで実行されるウェブサイトを作成できます。AWS SDK for JavaScript を使用してウェブアプリケーションを開発し、場所検索 を使用して場所を検索し、 を使用してマップに結果を表示します MapLibre。Amazon Location SDK を使用すると、位置情報を使ったプログラミングが容易になります。

- 既存の iOS または Android アプリケーションに位置情報機能を追加します。

  AWS SDK for Swift (iOS) または Kotlin (Android) を使用して Amazon Location を呼び出し、場所検索とマップ機能をアプリケーションに追加できます。 MapLibre を使用してマップをレンダリングします。他の言語でも使用できる AWS SDK は他にもあります。

- アセット (デバイスまたは車両) を追跡し、定義したエリアに出入りしたらアップデートを取得できます。

  デバイスを追跡するアプリケーションは複数の部分で構成されています。

  - 追跡する各デバイスには、追跡用のトラッカーリソースを作成する必要があります。MQTT などを使用して、位置の更新を Amazon Location Service に送信する必要があります。
  - ジオフェンスを作成して、アセットの出入りイベント情報を取得したいエリアを定義します。
  - Amazon EC2 やAWS Lambda を使用して、アセットがジオフェンスエリアに出入りした際のイベントに対応できます。
  - これを拡張させて、アセットの位置を追跡してマップ上に表示するウェブアプリケーションやデバイスアプリケーションを作成できます。

以下のセクションでは、Amazon Location Service 各機能で使用できるツールやライブラリについて詳しく説明します。

Amazon Location Service を使用するための SDK とツール

Amazon Location Service 使用に役立つツールはいくつかあります。

- AWS SDKs – Software AWS Development Kit (SDKs) は、多くの一般的なプログラミング言語で利用でき、任意の言語でアプリケーションを簡単に構築できる API、コード例、およびドキュメントを提供します。 AWS SDKs には、マップ、場所検索、ルート、ジオフェンス、トラッカーへのアクセスなど、Amazon Location のコア APIs と機能が含まれています。Amazon Location Service で使用できる SDK のさまざまなアプリケーションや言語の詳細については、「SDK (言語別)」を参照してください。
- MapLibre — Amazon Location Service では、レンダリングエンジンを使用してマップをMapLibreレンダリングすることを推奨しています。 MapLibre は、ウェブアプリケーションまたはモバイルアプリケーションでマップを表示するためのエンジンです。 MapLibre にはプラグインモデルも備わっており、一部の言語やプラットフォームで検索やルートを行うためのユーザーインターフェイスをサポートしています。の使用 MapLibre とそれが提供する機能の詳細については、「」を参照してくださいMapLibre。
- Amazon Location SDK — Amazon Location SDK は、Amazon Location Service を使用したアプリケーションの開発を容易にするオープンソースライブラリのセットです。ライブラリは、モバイルアプリケーションとウェブアプリケーションの認証、モバイルアプリケーションの場所追跡、Amazon Location データ型と GeoJSON、および AWS SDK v3 の Amazon Location クライアントのホストパッケージをサポートする機能を提供します。Amazon Location SDK の詳細については、「Amazon Location SDK」を参照してください。

SDK (言語別)

次の表は、 AWS SDKs と言語とフレームワーク MapLibre のバージョンに関する情報を、ウェブ、モバイル、バックエンドアプリケーションなどのアプリケーションタイプ別に示しています。

> **SDK のバージョン**
>
> AWS SDK の最新のビルド、およびプロジェクトで使用するその他の SDKsを使用して、SDKsを最新の状態に保つことをお勧めします。 AWS SDK は、最新の機能やセキュリティアップデートを提供します。例えば、 AWS SDK for の最新ビルドを見つけるには JavaScript、 SDK for ドキュメントの「ブラウザのインストールAWS JavaScript」トピックを参照してください。

Web frontend

ウェブフロントエンドアプリケーションの開発には、次の AWS SDKsと MapLibre バージョンを使用できます。

言語/フレームワーク	AWS SDK	レンダリングフレームワーク
完全サポートされています		
JavaScript	https://aws.amazon.com/sdk-for-javascript/	https://maplibre.org/projects/maplibre-gl-js/
ReactJS	https://aws.amazon.com/sdk-for-javascript/	https://github.com/maplibre/maplibre-react-native
TypeScript	https://aws.amazon.com/sdk-for-javascript/	https://maplibre.org/projects/maplibre-gl-js/
一部サポートされています		
Flutter	https://docs.amplify.aws/start/q/integration/flutter/ Flutter はまだ で完全にはサポートされていませんが AWS、Amplify 経由で提供されるサポートは限られています。	https://github.com/maplibre/flutter-maplibre-gl MapLibre Flutter ライブラリは実験的と見なされます。
Node.js	https://aws.amazon.com/sdk-for-javascript/	Node.js は MapLibre サポートされていません。
PHP	https://aws.amazon.com/sdk-for-php/	PHP は MapLibre サポートされていません。

Mobile frontend

モバイルフロントエンドアプリケーション開発では、次の AWS SDKsと MapLibre バージョンを使用できます。

言語/フレームワーク	AWS SDK	レンダリングフレームワーク
完全サポートされています		

言語/フレームワーク	AWS SDK	レンダリングフレームワーク
Java	https://aws.amazon.com/sdk-for-java/	https://maplibre.org/projects/maplibre-native/
Kotlin	https://aws.amazon.com/sdk-for-kotlin/ Android 用 Amazon Location Service Mobile Authentication SDK: https://github.com/aws-geospatial/amazon-location-mobile-auth-sdk-android Amazon Location Service Mobile Tracking SDK for Android: https://github.com/aws-geospatial/amazon-location-mobile-tracking-sdk-android	https://maplibre.org/projects/maplibre-native/ Java ベース MapLibre と同様に、カスタムバインディングが必要です。
ObjectiveC	https://github.com/aws-amplify/aws-sdk-ios	https://maplibre.org/projects/maplibre-native/
ReactNative	https://aws.amazon.com/sdk-for-javascript/	https://github.com/maplibre/maplibre-react-native

言語/フレームワーク	AWS SDK	レンダリングフレームワーク
Swift	https://aws.amazon.com/sdk-for-swift/ Amazon Location Service Mobile Authentication SDK for iOS: https://github.com/aws-geospatial/amazon-location-mobile-auth-sdk-ios Amazon Location Service Mobile Tracking SDK for iOS: https://github.com/aws-geospatial/amazon-location-mobile-tracking-sdk-ios	https://maplibre.org/projects/maplibre-native/
一部サポートされています		
Flutter	https://docs.amplify.aws/start/q/integration/flutter/ Flutter はまだ で完全にはサポートされていませんが AWS、Amplify 経由で提供されるサポートは限られています。	https://github.com/maplibre/flutter-maplibre-gl MapLibre Flutter ライブラリは実験的と見なされます。

Backend application

以下の AWS SDKs はバックエンドアプリケーション開発に使用できます。マップレンダリングは通常、バックエンドアプリケーションでは必要ないため、ここには記載 MapLibre されていません。

[言語]	AWS SDK
.NET	https://aws.amazon.com/sdk-for-net/

[言語]	AWS SDK
C++	https://aws.amazon.com/sdk-for-cpp/
Go	https://aws.amazon.com/sdk-for-go/
Java	https://aws.amazon.com/sdk-for-java/
JavaScript	https://aws.amazon.com/sdk-for-javascript/
Node.js	https://aws.amazon.com/sdk-for-javascript/
TypeScript	https://aws.amazon.com/sdk-for-javascript/
Kotlin	https://aws.amazon.com/sdk-for-kotlin/
PHP	https://aws.amazon.com/sdk-for-php/
Python	https://aws.amazon.com/sdk-for-python/
Ruby	https://aws.amazon.com/sdk-for-ruby/
Rust	https://aws.amazon.com/sdk-for-rust/ AWS SDK for Rust はデベロッパープレビュー中です。

Amazon Location での MapLibre ツールとライブラリの使用

Amazon Location でインタラクティブアプリケーションを作成するための重要なツールの 1 つは です MapLibre。MapLibre は主に、ウェブまたはモバイルアプリケーションでマップを表示するためのレンダリングエンジンです。ただし、プラグインのサポートも含まれており、Amazon Location の他の機能を操作する機能も備えています。以下では、作業したい領域に基づいて、使用できるツールについて説明します。

> **Note**
>
> Amazon Location のいずれかの機能を使用する場合には、使用する言語のAWS SDK をインストールします。

- マップ

  アプリケーションにマップを表示するには、Amazon Location から提供されたデータを使用して画面に描画するマップレンダリングエンジンが必要です。マップレンダリングエンジンには、マップを画面移動したりズームしたり、マーカーやプッシュピン、その他の注釈をマップに追加したりする機能もあります。

  Amazon Location Service では、レンダリングエンジンを使用してマップをMapLibreレンダリングすることを推奨しています。 MapLibre GL JS は でマップを表示するためのエンジンであり JavaScript、 MapLibre Native は iOS または Android のマップを提供します。

  MapLibre には、コア機能を拡張するプラグインエコシステムもあります。詳細については、https://maplibre.org/maplibre-gl-js-docs/plugins/ を参照してください。
- 場所検索

  検索ユーザーインターフェイスの作成を簡素化するには、ウェブ用のMapLibre ジオコーダーを使用できます (Android アプリケーションは Android Places プラグイン を使用できます)。

  Amazon Location for Maplibre ジオコーダーライブラリを使用して、 アプリケーションで で Amazon Location を使用するプロセスを簡素化`amazon-location-for-maplibre-gl-geocoder`します JavaScript。
- ルート

  マップにルートを表示するには、MapLibre方向 を使用します。
- ジオフェンスとトラッカー

  MapLibre にはジオフェンスと追跡用の特定のレンダリングやツールはありませんが、レンダリング機能とプラグインを使用して、ジオフェンスと追跡対象デバイスをマップに表示できます。

  追跡対象のデバイスは MQTT を使用するか、手動で Amazon Location Service へアップデートを送信できます。ジオフェンスイベントに応答するには、AWS Lambda を使用できます。

Amazon Location Service に追加機能を提供するオープンソースライブラリが多数用意されています。たとえば、空間分析機能を提供する Turf などです。

多くのライブラリは、オープンスタンダードの GeoJSON 形式のデータを使用しています。Amazon Location Service は、アプリケーションでの GeoJSON JavaScript の使用をサポートするライブラリを提供します。詳細については、次のセクション「Amazon Location SDK とライブラリ」を参照してください。

Amazon Location MapLibre Geocoder プラグイン

Amazon Location MapLibre ジオコーダープラグインは、 maplibre-gl-geocoderライブラリを使用してマップレンダリングとジオコーディングを操作するときに、Amazon Location 機能を JavaScript アプリケーションに簡単に組み込みできるように設計されています。

インストール

次のコマンドを使用して、モジュールで使用する Amazon Location MapLibre ジオコーダープラグインを NPM からインストールできます。

```
npm install @aws/amazon-location-for-maplibre-gl-geocoder
```

スクリプトを使用して、ブラウザで直接使用できるように HTML ファイルにインポートできます。

```
<script src="https://www.unpkg.com/@aws/amazon-location-for-maplibre-gl-
geocoder@1">/script<
```

モジュールの使用

このコードは、Amazon Location のジオコーディング機能を備えた Maplibre GL JavaScript マップを設定します。Amazon Cognito ID プールを介した認証を使用して、Amazon Location リソースにアクセスします。マップは指定されたスタイルと中央座標でレンダリングされ、 はマップ上の場所を検索できます。

```
// Import MapLibre GL JS
import maplibregl from "maplibre-gl";
// Import from the AWS JavaScript SDK V3
import { LocationClient } from "@aws-sdk/client-location";
// Import the utility functions
import { withIdentityPoolId } from "@aws/amazon-location-utilities-auth-helper";
// Import the AmazonLocationWithMaplibreGeocoder
import { buildAmazonLocationMaplibreGeocoder, AmazonLocationMaplibreGeocoder } from
 "@aws/amazon-location-for-maplibre-gl-geocoder"

const identityPoolId = "Identity Pool ID";
const mapName = "Map Name";
const region = "Region"; // region containing the Amazon Location resource
const placeIndex = "PlaceIndexName" // Name of your places resource in your AWS
 Account.
```

```
// Create an authentication helper instance using credentials from Amazon Cognito
const authHelper = await withIdentityPoolId("Identity Pool ID");

const client = new LocationClient({
  region: "Region", // Region containing Amazon Location resources
  ...authHelper.getLocationClientConfig(), // Configures the client to use
 credentials obtained via Amazon Cognito
});

// Render the map
const map = new maplibregl.Map({
    container: "map",
    center: [-123.115898, 49.295868],
    zoom: 10,
    style: `https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor`,
    ...authHelper.getMapAuthenticationOptions(),
});

// Gets an instance of the AmazonLocationMaplibreGeocoder Object.
const amazonLocationMaplibreGeocoder = buildAmazonLocationMaplibreGeocoder(client,
 placeIndex, {enableAll: true});

// Now we can add the Geocoder to the map.
map.addControl(amazonLocationMaplibreGeocoder.getPlacesGeocoder());
```

ブラウザでの使用

この例では、Amazon Location Client を使用して、Amazon Cognito を使用して認証するリクエストを行います。

> **Note**
>
> これらの例の一部では、Amazon Location Client を使用しています。Amazon Location Client は AWS SDK for JavaScript V3 に基づいており、HTML ファイルで参照されるスクリプトを使用して Amazon Location を呼び出すことができます。

HTML ファイルに以下を含めます。

```
< Import the Amazon Location With Maplibre Geocoder>
```

```
<script src="https://www.unpkg.com/@aws/amazon-location-with-maplibre-geocoder@1"></
script>
<Import the Amazon Location Client>
<script src="https://www.unpkg.com/@aws/amazon-location-client@1"></script>
<!Import the utility library>
<script src="https://www.unpkg.com/@aws/amazon-location-utilities-auth-helper@1"></
script>
```

ファイルに以下を含めます JavaScript 。

```
const identityPoolId = "Identity Pool ID";
const mapName = "Map Name";
const region = "Region"; // region containing Amazon Location resource

// Create an authentication helper instance using credentials from Amazon Cognito
const authHelper = await
 amazonLocationAuthHelper.withIdentityPoolId(identityPoolId);

// Render the map
const map = new maplibregl.Map({
  container: "map",
  center: [-123.115898, 49.295868],
  zoom: 10,
  style: `https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor`,
  ...authHelper.getMapAuthenticationOptions(),
});

// Initialize the AmazonLocationMaplibreGeocoder object
const amazonLocationMaplibreGeocoderObject =
 amazonLocationMaplibreGeocoder.buildAmazonLocationMaplibreGeocoder(client,
 placesName, {enableAll: true});

// Use the AmazonLocationWithMaplibreGeocoder object to add a geocoder to the map.
map.addControl(amazonLocationMaplibreGeocoderObject.getPlacesGeocoder());
```

Amazon Location MapLibre ジオコーダープラグインで使用される関数とコマンドを以下に示します。

- **`buildAmazonLocationMaplibreGeocoder`**

このクラスは、他のすべての呼び出しへのエントリポイントAmazonLocationMaplibreGeocderである のインスタンスを作成します。

```
const amazonLocationMaplibreGeocoder = buildAmazonLocationMaplibreGeocoder(client,
 placesIndex, {enableAll: true});
```

- **getPlacesGeocoder**

  マップに直接追加できる、すぐに使用できる IControl オブジェクトを返します。

```
const geocoder = getPlacesGeocoder();

// Initialize map
let map = await initializeMap();

// Add the geocoder to the map.
map.addControl(geocoder);
```

Amazon Location SDK とライブラリ

Amazon Location SDK は、Amazon Location アプリケーションの開発に役立つ機能を提供するオープンソースライブラリのセットです。以下の機能が含まれています。

- Amazon Location クライアント – AWS SDK v3 の Amazon Location オブジェクトは、ウェブ開発で簡単に使用できるようにバンドルおよびパッケージ化されています。
- 認証 — 認証ユーティリティは、Amazon Location Service 用のウェブページ、、iOS、または Android アプリケーションを構築する際の認証を簡素化します (Amazon Cognito または API キーを使用) 。 JavaScript https://docs.aws.amazon.com/location/latest/developerguide/dev-location-libraries.html#loc-sdk-auth-mobile-ios https://docs.aws.amazon.com/location/latest/developerguide/dev-location-libraries.html#loc-sdk-auth-mobile-Android
- 追跡 — モバイル追跡 SDKs は iOS および Android で使用できます。この SDK を使用すると、モバイルアプリケーションが Amazon Location Trackers とやり取りしやすくなります。
- Amazon Location GeoJSON 関数 – GeoJSON 変換ユーティリティを使用すると、業界標準の GeoJSON 形式のデータと Amazon Location API 形式を簡単に変換できます。

トピック

- Amazon Location SDK の使用を開始する方法
- Amazon Location クライアント
- JavaScript 認証ヘルパー
- GeoJSON 変換ヘルパー
- Android モバイル認証 SDK
- iOS モバイル認証 SDK
- Android モバイル追跡 SDK
- iOS モバイル追跡 SDK

Amazon Location SDK の使用を開始する方法

Amazon Location SDK は、アプリケーションで Amazon Location Service をより簡単に利用するための一連の関数です。これらの関数は、 JavaScript アプリケーションにインストールしてインポートできます。以下のセクションでは、Amazon Location クライアント、認証および GeoJSON ヘルパーライブラリについて説明します。

Amazon Location クライアント

AWS SDK v3 では、SDK はサービスごとに分離されます。必要なパーツだけをインストールできます。たとえば、Amazon Location クライアントと Amazon Cognito の認証情報プロバイダーをインストールするには、次のコマンドを使用します。

```
npm install @aws-sdk/client-location
npm install @aws-sdk/credential-providers
```

JavaScript ウェブフロントエンドアプリケーションで Amazon Location Service を簡単に使用するために、 AWS は Amazon Location ライブラリと認証情報プロバイダーのホストされたバンドルを提供します。バンドルされたクライアントを使用するには、以下のようにスクリプトタグで HTML に追加します。

```
<script src="https://unpkg.com/@aws/amazon-location-client@1.x/dist/
amazonLocationClient.js"></script>
```

> **Note**
>
> パッケージは最新の状態に保たれ、使いやすさのために下位互換性が保たれています。このスクリプトタグを使用するか NPM インストールを行うと、常に最新バージョンが取得されます。

JavaScript 認証ヘルパー

Amazon Location JavaScript 認証ヘルパーを使用すると、 JavaScript アプリケーションから Amazon Location API コールを行うときに認証が簡単になります。この認証ヘルパーは、Amazon Cognito または API キーを認証方法として使用する場合に特に役立ちます。これは、 で利用可能なオープンソースライブラリです GitHub。https://github.com/aws-geospatial/amazon-location-utilities-auth-helper-js 。

> **Note**
>
> 認証ヘルパーの Amazon Cognito サポートは、Amazon Cognito のフェデレーテッドアイデンティティ機能をサポートしていません。

インストール

Webpack などのビルドシステムを使用する場合、または HTML に<script>タグ付きのビルド済み JavaScript バンドルを含めることで、ローカルインストールでライブラリを使用できます。

- NPM を使用して次のコマンドを実行し、ライブラリをインストールします。

```
npm install @aws/amazon-location-utilities-auth-helper
```

- HTML ファイルで以下のコマンドを使用してスクリプトをロードします。

```
<script src="https://unpkg.com/@aws/amazon-location-utilities-auth-helper@1.x/dist/
amazonLocationAuthHelper.js"></script>
```

[Import] (インポート)

JavaScript アプリケーションで特定の関数を使用するには、その関数をインポートする必要があります。次のコードを使用して、withIdentityPoolId の関数をアプリケーションにインポートします。

```
import { withIdentityPoolId } from '@aws/amazon-location-utilities-auth-helper';
```

認証関数

Amazon Location 認証ヘルパーには、AuthHelper オブジェクトを返す以下の関数が含まれています。

- async withIdentityPoolId(identityPoolId: string): AuthHelper – この関数は AuthHelper 、Amazon Cognito で動作するように初期化された オブジェクトを返します。
- async withAPIKey(API_KEY: string): AuthHelper – この関数は AuthHelper 、API キーで動作するように初期化された オブジェクトを返します。

AuthHelper オブジェクトは、以下の関数を備えています。

- AuthHelper.getMapAuthenticationOptions() – AuthHelper オブジェクトのこの関数は、MapLibre JS のマップオプションtransformRequestで使用できる を持つ JavaScript オブジェクトを返します。アイデンティティプールで初期化された場合にのみ提供されます
- AuthHelper.getLocationClientConfig() – AuthHelper オブジェクトのこの関数は、を初期化するためにcredentials使用できる を持つ JavaScript オブジェクトを返します LocationClient。
- AuthHelper.getCredentials() – AuthHelper オブジェクトのこの関数は、Amazon Cognito から内部認証情報を返します。アイデンティティプールで初期化された場合にのみ提供されます

例: を使用して Amazon Cognito で MapLibre マップオブジェクトを初期化する AuthHelper

```
import { withIdentityPoolId } from '@aws/amazon-location-utilities-auth-helper';

const authHelper = await withIdentityPoolId("identity-pool-id"); // use Cognito pool id
 for credentials

const map = new maplibregl.Map({
    container: "map", // HTML element ID of map element
    center: [-123.1187, 49.2819], // initial map center point
    zoom: 16, // initial map zoom
```

```
    style: https://maps.geo.region.amazonaws.com/maps/v0/maps/mapName/style-
descriptor', // Defines the appearance of the map
    ...authHelper.getMapAuthenticationOptions(), // Provides credential options
 required for requests to Amazon Location
});
```

例: API キーを使用した MapLibre マップオブジェクトの初期化 (**AuthHelper**この場合は必須ではありません)

```
const map = new maplibregl.Map({
    container: "map", // HTML element ID of map element
    center: [-123.1187, 49.2819], // initial map center point
    zoom: 16, // initial map zoom
    style: https://maps.geo.region.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor?key=api-key-id',
});
```

例: Amazon Cognito と を使用して AWS SDK for JS から Location クライアントを初期化する Amazon Cognito AuthHelper

この例では、 AWS SDK for JavaScript v3 を使用しています。

```
import { withIdentityPoolId } from '@aws/amazon-location-utilities-auth-helper';

const authHelper = await withIdentityPoolId("identity-pool-id"); // use Cognito pool id
 for credentials

//initialize the Location client:
const client = new LocationClient({
    region: "region",
    ...authHelper.getLocationClientConfig() // sets up the Location client to use the
 Cognito pool defined above
});

//call a search function with the location client:
const result = await client.send(new SearchPlaceIndexForPositionCommand({
    IndexName: "place-index", // Place index resource to use
    Position: [-123.1187, 49.2819], // position to search near
    MaxResults: 10 // number of results to return
});
```

例: API キーと を使用して、 AWS SDK for JS から Location クライアントを初期化する AuthHelper

この例では、 AWS SDK for JavaScript v3 を使用しています。

```
import { withAPIKey } from '@aws/amazon-location-utilities-auth-helper';

const authHelper = await withAPIKey("api-key-id"); // use API Key id for credentials

//initialize the Location client:
const client = new LocationClient({
    region: "region",
    ...authHelper.getLocationClientConfig() // sets up the Location client to use the
 API Key defined above
});

//call a search function with the location client:
const result = await client.send(new SearchPlaceIndexForPositionCommand({
    IndexName: "place-index", // Place index resource to use
    Position: [-123.1187, 49.2819], // position to search near
    MaxResults: 10 // number of results to return
});
```

GeoJSON 変換ヘルパー

Amazon Location GeoJSON 変換ヘルパーは、Amazon Location Service データ型を業界標準の GeoJSON 形式に変換したり、業界標準の GeoJSON 形式から変換したりするためのツールを提供します。GeoJSON は、例えば、マップ上の地理的データをレンダリング MapLibre するために とともに使用されます。これは、 で利用可能なオープンソースライブラリです GitHub。https://github.com/aws-geospatial/amazon-location-utilities-datatypes-js 。

インストール

ライブラリは、webpack などのローカルインストールで使用するか、HTML に<script>タグ付きのビルド済み JavaScript バンドルを含めることで使用できます。

- NPM を使用して次のコマンドを実行し、ライブラリをインストールします。

```
npm install @aws/amazon-location-utilities-datatypes
```

- HTML ファイルで以下のコマンドを使用してスクリプトをロードします。

```
<script src="https://unpkg.com/@aws/amazon-location-utilities-datatypes@1.x/dist/
amazonLocationDataConverter.js"></script>
```

[Import] (インポート)

JavaScript アプリケーションで特定の関数を使用するには、その関数をインポートする必要があります。次のコードを使用して、placeToFeatureCollection の関数をアプリケーションにインポートします。

```
import { placeToFeatureCollection } from '@aws/amazon-location-utilities-datatypes';
```

GeoJSON 変換関数

Amazon Location GeoJSON 変換ヘルパーには、以下の関数が含まれています。

- placeToFeatureCollection(place: GetPlaceResponse | searchPlaceIndexForPositionResponse | searchPlaceIndexForTextResponse, keepNull: boolean): Feature – この関数は、場所検索関数からのレスポンスを、1 つ以上のポイント特徴を持つ GeoJSON FeatureCollection に変換します。
- devicePositionToFeatureCollection(devicePositions: GetDevicePositionResponse | BatchGetDevicePositionResponse | GetDevicePositionHistoryResponse | ListDevicePositionsResponse, keepNull: boolean) – この関数は、トラッカーデバイス位置関数からのレスポンスを、1 つ以上のポイント特徴を持つ GeoJSON FeatureCollection に変換します。
- routeToFeatureCollection(legs: CalculateRouteResponse): FeatureCollection – この関数は、ルート計算関数からのレスポンスを、単一の MultiStringLine 機能を持つ GeoJSON FeatureCollection に変換します。ルートの各レッグは、 の LineString エントリで表されます MultiStringLine。
- geofenceToFeatureCollection(geofences: GetGeofenceResponse | PutGeofenceRequest | BatchPutGeofenceRequest | ListGeofencesResponse): FeatureCollection — この関数は、ジオフェンス関数のリクエストまたはレスポンスをポリゴン機能を持つ GeoJSON FeatureCollection に変換します。レスポンスとリクエストの両方でジオフェンスを変換できるため、 PutGeofence または でアップロードする前にジオフェンスをマップに表示できます BatchPutGeofence。

  この関数は、円形ジオフェンスを近似ポリゴンフィーチャに変換しますが、必要に応じて、円形ジオフェンスを再作成するための「中心」プロパティと「半径」プロパティも備えています (次の関数を参照)。
- featureCollectionToGeofences(featureCollection: FeatureCollection): BatchPutGeofenceRequestEntry[] — この関数は、ポリゴン特徴を持つ GeoJSON

FeatureCollection を BatchPutGeofenceRequestEntry オブジェクトの配列に変換するため、結果を使用して へのリクエストを作成できます BatchPutGeofence。

の特徴量に「中心」プロパティと「半径」プロパティ FeatureCollection がある場合、ポリゴンのジオメトリを無視して、円ジオフェンスリクエストエントリに変換されます。

例: 検索結果を のポイントレイヤーに変換する MapLibre

この例では、 AWS SDK for JavaScript v3 を使用しています。

```
import { placeToFeatureCollection } from '@aws/amazon-location-utility-datatypes';

...

let map; // map here is an initialized MapLibre instance

const client = new LocationClient(config);
const input = { your_input };
const command = new searchPlaceIndexForTextCommand(input);
const response = await client.send(command);

// calling utility function to convert the response to GeoJSON
const featureCollection = placeToFeatureCollection(response);
map.addSource("search-result", featureCollection);
map.addLayer({
    id: "search-result",
    type: "circle",
    source: "search-result",
    paint: {
        "circle-radius": 6,
        "circle-color": "#B42222",
    },
});
```

Android モバイル認証 SDK

これらのユーティリティは、Android アプリケーションから Amazon Location Service API コールを行う際の認証に役立ちます。これは、Amazon Cognito または API キーを認証方法として使用する場合に特に役立ちます。

Android モバイル認証 SDK は、github: Amazon Location Service Mobile Authentication SDK for Android で利用できます。さらに、モバイル認証 SDK と AWS SDK の両方が AWS Maven リポジトリ で利用できます。

インストール

モバイル認証 SDK を使用するには、Android Studio のbuild.gradleファイルに次のインポートステートメントを追加します。

```
implementation("software.amazon.location:auth:0.0.1")
implementation("com.amazonaws:aws-android-sdk-location:2.72.0")
```

認証関数

認証ヘルパー SDK には次の機能があります。

- authHelper.authenticateWithApiKey("My-Amazon-Location-API-Key"): LocationCredentialsProvider: この関数は、 API キーを操作するためにLocationCredentialsProvider初期化された を返します。
- authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id"): LocationCredentialsProvider: この関数は、Amazon Cognito ID プールと連携するようにLocationCredentialsProvider初期化された を返します。

使用方法

コードで SDK を使用するには、次のクラスをインポートします。

```
import com.amazonaws.services.geo.AmazonLocationClient
import software.amazon.location.auth.AuthHelper
import software.amazon.location.auth.LocationCredentialsProvider
```

認証ヘルパーとロケーションクライアントプロバイダーインスタンスを作成するときは、2 つのオプションがあります。Amazon Location API キーまたは Amazon Cognitoを使用してインスタンスを作成できます。

- Amazon Location API キーを使用して認証ヘルパーインスタンスを作成するには、ヘルパークラスを次のように宣言します。

```
var authHelper = AuthHelper(applicationContext)
```

```
var locationCredentialsProvider : LocationCredentialsProvider =
 authHelper.authenticateWithApiKey("My-Amazon-Location-API-Key")
```

- Amazon Cognito を使用して認証ヘルパーインスタンスを作成するには、ヘルパークラスを次のように宣言します。

```
var authHelper = AuthHelper(applicationContext)
var locationCredentialsProvider : LocationCredentialsProvider =
 authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id")
```

ロケーション認証情報プロバイダーを使用して Amazon Location クライアントインスタンスを作成し、Amazon Location サービスを呼び出すことができます。次の例では、指定された緯度と経度に近い場所を検索します。

```
var locationClient =
 authHelper.getLocationClient(locationCredentialsProvider.getCredentialsProvider())
var searchPlaceIndexForPositionRequest =
 SearchPlaceIndexForPositionRequest().withIndexName("My-Place-Index-
Name").withPosition(arrayListOf(30.405423, -97.718833))
var nearbyPlaces =
 locationClient.searchPlaceIndexForPosition(searchPlaceIndexForPositionRequest)
```

iOS モバイル認証 SDK

これらのユーティリティは、iOS アプリケーションから Amazon Location Service API コールを行う際の認証に役立ちます。これは、Amazon Cognito または API キーを認証方法として使用する場合に特に役立ちます。

iOS モバイル認証 SDK は、github: Amazon Location Service Mobile Authentication SDK for iOS で利用できます。

インストール

Xcode プロジェクトに SDK をインストールします。

1. ファイル に移動し、XCode プロジェクトでパッケージ依存関係の追加を選択します。
2. 検索バーにパッケージ URL: https://github.com/aws-geospatial/amazon-location-mobile-auth-sdk-ios/ を入力し、Enter キーを押します。
3. `amazon-location-mobile-auth-sdk-ios` パッケージを選択し、パッケージの追加 を押します。

4. AmazonLocationiOSAuthSDK パッケージ製品を選択し、パッケージの追加 を押します。

認証関数

認証ヘルパー SDK には次の機能があります。

- authHelper.authenticateWithApiKey("My-Amazon-Location-API-Key"): LocationCredentialsProvider: この関数は、 API キーを操作するためにLocationCredentialsProvider初期化された を返します。
- authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id"): LocationCredentialsProvider: この関数は、Amazon Cognito ID プールと連携するようにLocationCredentialsProvider初期化された を返します。

使用方法

モバイル認証 SDK を使用するには、アクティビティに次のステートメントを追加します。

```
import AmazonLocationiOSAuthSDK
import AWSLocationXCF
```

認証ヘルパーとロケーションクライアントプロバイダーインスタンスを作成するときは、2 つのオプションがあります。Amazon Location API キーまたは Amazon Cognitoを使用してインスタンスを作成できます。

- Amazon Location API キーを使用して認証ヘルパーインスタンスを作成するには、ヘルパークラスを次のように宣言します。

```
let authHelper = AuthHelper()
let locationCredentialsProvider = authHelper.authenticateWithAPIKey(apiKey: "My-Amazon-Location-API-Key", region: "account-region")
```

- Amazon Cognito を使用して認証ヘルパーインスタンスを作成するには、ヘルパークラスを次のように宣言します。

```
let authHelper = AuthHelper()
let locationCredentialsProvider =
 authHelper.authenticateWithCognitoUserPool(identityPoolId: "My-Amazon-Location-API-Key", region: "account-region")
```

ロケーション認証情報プロバイダーを使用して Amazon Location クライアントインスタンスを作成し、Amazon Location サービスを呼び出すことができます。次の例では、指定された緯度と経度に近い場所を検索します。

```
let locationClient = AWSLocation.default()
let searchPlaceIndexForPositionRequest =
 AWSLocationSearchPlaceIndexForPositionRequest()!
searchPlaceIndexForPositionRequest.indexName = "My-Place-Index-Name"
searchPlaceIndexForPositionRequest.position = [30.405423, -97.718833]
let nearbyPlaces = locationClient.searchPlaceIndex(forPosition:
 searchPlaceIndexForPositionRequest)
```

Android モバイル追跡 SDK

Amazon Location モバイル追跡 SDK は、Amazon Location Trackers の認証、デバイス位置のキャプチャ、位置の更新の送信を容易にするユーティリティを提供します。SDK は、設定可能な更新間隔によるロケーション更新のローカルフィルタリングをサポートしています。これにより、データコストを削減し、Android アプリケーションの断続的な接続を最適化できます。

Android 追跡 SDK は、 GitHub: Amazon Location Mobile Tracking SDK for Android で利用できます。さらに、モバイル認証 SDK と AWS SDK の両方が AWS Maven リポジトリ で利用できます。Android 追跡 SDK は、一般的な AWS SDK で動作するように設計されています。

このセクションでは、Amazon Location モバイル追跡 Android SDK に関する以下のトピックについて説明します。

トピック

- インストール
- 使用方法
- フィルター
- Android Mobile SDK 追跡関数
- 例

インストール

SDK をインストールするには、Android Studio の build.gradle ファイルの依存関係セクションに次の行を追加します。

```
implementation("software.amazon.location:tracking:0.0.1")
```

```
implementation("software.amazon.location:auth:0.0.1")
implementation("com.amazonaws:aws-android-sdk-location:2.72.0")
```

使用方法

この手順では、 SDK を使用してLocationTrackerオブジェクトを認証および作成する方法について説明します。

> **Note**
>
> この手順では、 インストールセクションで説明されているライブラリをインポートしていることを前提としています。

1. コードに次のクラスをインポートします。

```
import software.amazon.location.tracking.LocationTracker
import software.amazon.location.tracking.config.LocationTrackerConfig
import software.amazon.location.tracking.util.TrackingSdkLogLevel
import com.amazonaws.services.geo.AmazonLocationClient
import software.amazon.location.auth.AuthHelper
import software.amazon.location.auth.LocationCredentialsProvider
```

2. 次にAuthHelper、 を作成します。LocationTrackerオブジェクトの作成にはLocationCredentialsProviderパラメータが必要なためです。

```
// Create an authentication helper using credentials from Cognito
val authHelper = AuthHelper(applicationContext)
val locationCredentialsProvider : LocationCredentialsProvider =
 authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id")
```

3. 次に、 LocationCredentialsProviderと を使用してLocationTrackerオブジェクトLocationTrackerConfigを作成します。

```
val config = LocationTrackerConfig(
    trackerName = "MY-TRACKER-NAME",
    logLevel = TrackingSdkLogLevel.DEBUG,
    accuracy = Priority.PRIORITY_HIGH_ACCURACY,
    latency = 1000,
    frequency = 5000,
    waitForAccurateLocation = false,
```

```
    minUpdateIntervalMillis = 5000,
)
locationTracker = LocationTracker(
    applicationContext,
    locationCredentialsProvider,
    config,
)
```

フィルター

Amazon Location モバイル追跡 Android SDK には、3 つの組み込みロケーションフィルターがあります。

- `TimeLocationFilter`: 定義された時間間隔に基づいて、アップロードする現在の場所をフィルタリングします。
- `DistanceLocationFilter`: 指定された距離しきい値に基づいてロケーションの更新をフィルタリングします。
- `AccuracyLocationFilter`: 前回の更新以降に移動した距離を現在の場所の精度と比較することで、場所の更新をフィルタリングします。

この例では、作成`LocationTracker`時に にフィルターを追加します。

```
val config = LocationTrackerConfig(
    trackerName = "MY-TRACKER-NAME",
    logLevel = TrackingSdkLogLevel.DEBUG,
    accuracy = Priority.PRIORITY_HIGH_ACCURACY,
    latency = 1000,
    frequency = 5000,
    waitForAccurateLocation = false,
    minUpdateIntervalMillis = 5000,
    locationFilters = mutableListOf(TimeLocationFilter(), DistanceLocationFilter(),
 AccuracyLocationFilter())
)
locationTracker = LocationTracker(
    applicationContext,
    locationCredentialsProvider,
    config,
)
```

この例では、 で実行時にフィルターを有効または無効にします`LocationTracker`。

```
// To enable the filter
locationTracker?.enableFilter(TimeLocationFilter())

// To disable the filter
locationTracker?.disableFilter(TimeLocationFilter())
```

Android Mobile SDK 追跡関数

Amazon Location モバイル追跡 SDK for Android には、次の機能が含まれています。

- クラス : `LocationTracker`

  `constructor(context: Context,locationCredentialsProvider: LocationCredentialsProvider,trackerName: String)`, または `constructor(context: Context,locationCredentialsProvider: LocationCredentialsProvider,clientConfig: LocationTrackerConfig)`

  これはオブジェクトを作成するためのイニシャライザー関数です`LocationTracker`。これには、`LocationCredentialsProvider` のインスタンス`trackerName`と、オプションで のインスタンスが必要です`LocationTrackingConfig`。設定が指定されていない場合、デフォルト値で初期化されます。
- クラス : `LocationTracker`

  `start(locationTrackingCallback: LocationTrackingCallback)`

  ユーザーの場所にアクセスして Amazon Location トラッカーに送信するプロセスを開始します。
- クラス : `LocationTracker`

  `isTrackingInForeground()`

  ロケーション追跡が現在進行中かどうかを確認します。
- クラス : `LocationTracker`

  `stop()`

  ユーザーの位置を追跡するプロセスを停止します。
- クラス : `LocationTracker`

  `startTracking()`

ユーザーの場所にアクセスして AWS トラッカーに送信するプロセスを開始します。

- クラス : LocationTracker

  ```
  startBackground(mode: BackgroundTrackingMode, serviceCallback:
  ServiceCallback)
  ```

  アプリケーションがバックグラウンドにある間、ユーザーのロケーションにアクセスして AWS トラッカーに送信するプロセスを開始します。 BackgroundTrackingMode には次のオプションがあります。

  - ACTIVE_TRACKING: このオプションは、ユーザーの場所の更新をアクティブに追跡します。
  - BATTERY_SAVER_TRACKING: このオプションは、15 分ごとにユーザーの位置の更新を追跡します。

- クラス : LocationTracker

  ```
  stopBackgroundService()
  ```

  アプリケーションがバックグラウンドにある間、ユーザーのロケーションにアクセスして AWS トラッカーに送信するプロセスを停止する。

- クラス : LocationTracker

  ```
  getTrackerDeviceLocation()
  ```

  Amazon Location サービスからデバイスロケーションを取得します。

- クラス : LocationTracker

  ```
  getDeviceLocation(locationTrackingCallback: LocationTrackingCallback?)
  ```

  フュージョンロケーションプロバイダークライアントから現在のデバイスロケーションを取得し、Amazon Location トラッカーにアップロードします。

- クラス : LocationTracker

  ```
  uploadLocationUpdates(locationTrackingCallback:
  LocationTrackingCallback?)
  ```

  設定されたロケーションフィルターに基づいてフィルタリングした後、Amazon Location サービスにデバイスロケーションをアップロードします。

- クラス : LocationTracker

```
enableFilter(filter: LocationFilter)
```

特定のロケーションフィルターを有効にします。

- クラス : LocationTracker

```
checkFilterIsExistsAndUpdateValue(filter: LocationFilter)
```

特定の場所フィルターを無効にします。

- クラス : LocationTrackerConfig

```
LocationTrackerConfig( // Required var trackerName:
String, // Optional var locationFilters: MutableList =
mutableListOf( TimeLocationFilter(), DistanceLocationFilter(), ), var
logLevel: TrackingSdkLogLevel = TrackingSdkLogLevel.DEBUG, var accuracy:
Int = Priority.PRIORITY_HIGH_ACCURACY, var latency: Long = 1000, var
frequency: Long = 1500, var waitForAccurateLocation: Boolean = false, var
minUpdateIntervalMillis: Long = 1000, var persistentNotificationConfig:
NotificationConfig = NotificationConfig())
```

これにより、ユーザー定義のパラメータ値LocationTrackerConfigで が初期化されます。パラメータ値を指定しない場合、デフォルト値に設定されます。

- クラス : LocationFilter

```
shouldUpload(currentLocation: LocationEntry, previousLocation:
LocationEntry?): Boolean
```

LocationFilter は、ユーザーがカスタムフィルター実装のために実装できるプロトコルです。以前の場所と現在の場所を比較shouldUploadし、現在の場所をアップロードする必要がある場合は 関数を実装する必要があります。

例

次のコードサンプルは、モバイル追跡 SDK 機能を示しています。

この例では、 を使用してバックグラウンドで追跡LocationTrackerを開始および停止します。

```
// For starting the location tracking
locationTracker?.startBackground(
```

```
BackgroundTrackingMode.ACTIVE_TRACKING,
object : ServiceCallback {
    override fun serviceStopped() {
        if (selectedTrackingMode == BackgroundTrackingMode.ACTIVE_TRACKING) {
            isLocationTrackingBackgroundActive = false
        } else {
            isLocationTrackingBatteryOptimizeActive = false
        }
    }
},
)

// For stopping the location tracking
locationTracker?.stopBackgroundService()
```

iOS モバイル追跡 SDK

Amazon Location モバイル追跡 SDK は、Amazon Location Trackers の認証、デバイス位置のキャプチャ、位置の更新の送信を容易にするユーティリティを提供します。SDK は、設定可能な更新間隔によるロケーション更新のローカルフィルタリングをサポートしています。これにより、データコストを削減し、iOS アプリケーションの断続的な接続を最適化できます。

iOS 追跡 SDK は、 GitHub: Amazon Location Mobile Tracking SDK for iOS で利用できます。

このセクションでは、Amazon Location モバイル追跡 iOS SDK に関する以下のトピックについて説明します。

トピック

- インストール
- 使用方法
- フィルター
- iOS Mobile SDK 追跡関数
- 例

インストール

iOS 用モバイル追跡 SDK をインストールするには、次の手順に従います。

1. Xcode プロジェクトで、ファイル に移動し、パッケージ依存関係の追加 を選択します。

2. https://github.com/aws-geospatial/amazon-location-mobile-tracking-sdk-ios/ という URL を検索バーに入力し、Enter キーを押します。
3. amazon-location-mobile-tracking-sdk-ios パッケージを選択し、パッケージの追加 をクリックします。
4. AmazonLocationiOSTrackingSDK パッケージ製品を選択し、パッケージの追加 をクリックします。

使用方法

次の手順は、Cognito の認証情報を使用して認証ヘルパーを作成する方法を示しています。

1. ライブラリをインストールしたら、説明の一方または両方を info.plist ファイルに追加する必要があります。

```
Privacy - Location When In Use Usage Description
Privacy - Location Always and When In Use Usage Description
```

2. 次に、 クラス AuthHelper に をインポートします。

```
import AmazonLocationiOSAuthSDKimport AmazonLocationiOSTrackingSDK
```

3. 次に、Amazon Cognito の認証情報を使用して認証ヘルパーを作成して、 AWS SDK でAuthHelperオブジェクトを作成します。

```
let authHelper = AuthHelper()
let locationCredentialsProvider =
 authHelper.authenticateWithCognitoUserPool(identityPoolId: "My-Cognito-Identity-
Pool-Id", region: "My-region") //example: us-east-1
let locationTracker = LocationTracker(provider: locationCredentialsProvider,
 trackerName: "My-tracker-name")

// Optionally you can set ClientConfig with your own values in either initialize or
 in a separate function
// let trackerConfig = LocationTrackerConfig(locationFilters:
 [TimeLocationFilter(), DistanceLocationFilter()],

trackingDistanceInterval: 30,
trackingTimeInterval: 30,
logLevel: .debug)
```

```
// locationTracker = LocationTracker(provider: credentialsProvider, trackerName:
 "My-tracker-name",config: trackerConfig)
// locationTracker.setConfig(config: trackerConfig)
```

フィルター

Amazon Location モバイル追跡 iOS SDK には、3 つの組み込み口ケーションフィルターがあります。

- TimeLocationFilter: 定義された時間間隔に基づいて、アップロードする現在の場所をフィルタリングします。
- DistanceLocationFilter: 指定された距離しきい値に基づいて口ケーションの更新をフィルタリングします。
- AccuracyLocationFilter: 前回の更新以降に移動した距離を現在の場所の精度と比較することで、場所の更新をフィルタリングします。

この例では、作成LocationTracker時に にフィルターを追加します。

```
val config = LocationTrackerConfig(
    trackerName = "MY-TRACKER-NAME",
    logLevel = TrackingSdkLogLevel.DEBUG,
    accuracy = Priority.PRIORITY_HIGH_ACCURACY,
    latency = 1000,
    frequency = 5000,
    waitForAccurateLocation = false,
    minUpdateIntervalMillis = 5000,
    locationFilters = mutableListOf(TimeLocationFilter(), DistanceLocationFilter(),
 AccuracyLocationFilter())
)

locationTracker = LocationTracker(
    applicationContext,
    locationCredentialsProvider,
    config,
)
```

この例では、 で実行時にフィルターを有効または無効にしますLocationTracker。

```
// To enable the filter
locationTracker?.enableFilter(TimeLocationFilter())
```

```
// To disable the filter
locationTracker?.disableFilter(TimeLocationFilter())
```

iOS Mobile SDK 追跡関数

Amazon Location モバイル追跡 SDK for iOS には、次の機能が含まれています。

- クラス : `LocationTracker`

  `init(provider: LocationCredentialsProvider, trackerName: String, config: LocationTrackerConfig? = nil)`

  これはオブジェクトを作成するためのイニシャライザー関数です`LocationTracker`。これには、`LocationCredentialsProvider` のインスタンス`trackerName`と、オプションで のインスタンスが必要です`LocationTrackingConfig`。設定が指定されていない場合、デフォルト値で初期化されます。
- クラス : `LocationTracker`

  `setTrackerConfig(config: LocationTrackerConfig)`

  これにより、位置トラッカーの初期化後の任意の時点でトラッカーの設定が有効になります。
- クラス : `LocationTracker`

  `getTrackerConfig()`

  これにより、アプリで使用または変更する位置追跡設定が取得されます。

  戻り値: `LocationTrackerConfig`
- クラス : `LocationTracker`

  `getDeviceId()`

  ロケーショントラッカーが生成したデバイス ID を取得します。

  戻り値: `String?`
- クラス : `LocationTracker`

  `startTracking()`

  ユーザーの場所にアクセスして AWS トラッカーに送信するプロセスを開始します。

- クラス : LocationTracker

  resumeTracking()

  ユーザーの場所にアクセスして AWS トラッカーに送信するプロセスを再開します。

- クラス : LocationTracker

  stopTracking()

  ユーザーの位置を追跡するプロセスを停止します。

- クラス : LocationTracker

  startBackgroundTracking(mode: BackgroundTrackingMode)

  アプリケーションがバックグラウンドにある間、ユーザーのロケーションにアクセスして AWS トラッカーに送信するプロセスを開始します。 BackgroundTrackingMode には次のオプションがあります。

  - Active: このオプションは、ロケーションの更新を自動的に一時停止しません。
  - BatterySaving: このオプションは、ロケーションの更新を自動的に一時停止します。
  - None: このオプションでは、全体的にバックグラウンドロケーションの更新が無効になります。

- クラス : LocationTracker

  resumeBackgroundTracking(mode: BackgroundTrackingMode)

  アプリケーションがバックグラウンドにある間、ユーザーの場所にアクセスして AWS トラッカーに送信するプロセスを再開します。

- クラス : LocationTracker

  stopBackgroundTracking()

  アプリケーションがバックグラウンドにある間、ユーザーのロケーションにアクセスして AWS トラッカーに送信するプロセスを停止する。

- クラス : LocationTracker

  getTrackerDeviceLocation(nextToken: String?, startTime: Date? = nil, endTime: Date? = nil, completion: @escaping (Result<GetLocationResponse, Error>)

開始日時と終了日時の間にアップロードされたユーザーのデバイスの追跡場所を取得します。

戻り値: Void

- クラス : LocationTrackerConfig

  init()

  これにより、デフォルト値 LocationTrackerConfig で が初期化されます。

- クラス : LocationTrackerConfig

  init(locationFilters: [LocationFilter]? = nil, trackingDistanceInterval: Double? = nil, trackingTimeInterval: Double? = nil, trackingAccuracyLevel: Double? = nil, uploadFrequency: Double? = nil, desiredAccuracy: CLLocationAccuracy? = nil, activityType: CLActivityType? = nil, logLevel: LogLevel? = nil)

  これにより、ユーザー定義のパラメータ値LocationTrackerConfigで が初期化されます。パラメータ値を指定しない場合、デフォルト値に設定されます。

- クラス : LocationFilter

  shouldUpload(currentLocation: LocationEntity, previousLocation: LocationEntity?, trackerConfig: LocationTrackerConfig)

  LocationFilter は、ユーザーがカスタムフィルター実装のために実装できるプロトコルです。ユーザーは、以前の場所と現在の場所を比較し、現在の場所をアップロードする必要があるかどうかを返す shouldUpload関数を実装する必要があります。

例

このセクションでは、Amazon Location Mobile Tracking SDK for iOS の使用例について詳しく説明します。

> **Note**
>
> ファイルに必要なアクセス許可が設定されていることを確認しますinfo.plist。これらは、 使用方法セクションにリストされているのと同じアクセス許可です。

次の例は、デバイスの位置を追跡し、追跡された位置を取得する機能を示しています。

```
Privacy - Location When In Use Usage Description
Privacy - Location Always and When In Use Usage Description
```

ロケーションの追跡を開始します。

```
do {
    try locationTracker.startTracking()
    }
catch TrackingLocationError.permissionDenied {
        // Handle permissionDenied by showing the alert message or opening the app
 settings
        }
```

ロケーションの追跡を再開します。

```
do {
    try locationTracker.resumeTracking()
    }
catch TrackingLocationError.permissionDenied {
    // Handle permissionDenied by showing the alert message or opening the app settings
    }
```

ロケーションの追跡を停止します。

```
locationTracker.stopTracking()
```

バックグラウンド追跡を開始する：

```
do {

 locationTracker.startBackgroundTracking(mode: .Active) // .Active, .BatterySaving, .None
    }
catch TrackingLocationError.permissionDenied {
   // Handle permissionDenied by showing the alert message or opening the app settings
   }
```

バックグラウンド追跡を再開します。

```
do {
```

```
    locationTracker.resumeBackgroundTracking(mode: .Active)
    }
catch TrackingLocationError.permissionDenied {
    // Handle permissionDenied by showing the alert message or opening the app settings
    }
```

バックグラウンド追跡を停止するには：

```
locationTracker.stopBackgroundTracking()
```

トラッカーからデバイスの追跡対象ロケーションを取得します。

```
func getTrackingPoints(nextToken: String? = nil) {
let startTime: Date = Date().addingTimeInterval(-86400) // Yesterday's day date and
 time
let endTime: Date = Date()
locationTracker.getTrackerDeviceLocation(nextToken: nextToken, startTime: startTime,
 endTime: endTime, completion: { [weak self] result in
    switch result {
    case .success(let response):

        let positions = response.devicePositions
        // You can draw positions on map or use it further as per your requirement

        // If nextToken is available, recursively call to get more data
        if let nextToken = response.nextToken {
            self?.getTrackingPoints(nextToken: nextToken)
        }
    case .failure(let error):
        print(error)
    }
})
}
```

Amazon Location API

Amazon Location Service は、位置情報機能にプログラムでアクセスするための API オペレーションを提供します。これには、マップ、場所、ルート、トラッカー、ジオフェンス、リソースのタグ付け用の API が含まれます。使用可能な API アクションについては、「Amazon Location Service API リファレンス」をご覧ください。

このガイドの「コードの例」章にサンプルがあります。

AWS SDK での Amazon Location の使用

AWS Software Development Kit (SDKsは、多くの一般的なプログラミング言語で使用できます。各 SDK には、開発者が好みの言語で AWS アプリケーションを簡単に構築できるようにする API、コード例、およびドキュメントが用意されています。

Amazon Location Service で使用できる SDK の言語別の詳細については、このガイドの「SDK (言語別) 」を参照してください。

> **SDK のバージョン**
>
> プロジェクトで使用する AWS SDK および他の SDKsの最新のビルドを使用し、SDKsを最新の状態に保つことをお勧めします。 AWS SDK は、最新の機能やセキュリティアップデートを提供します。例えば、 SDK for の最新ビルドを確認するには JavaScript、 AWS SDK for ドキュメントの「ブラウザのインストールAWS JavaScript」トピックを参照してください。

Amazon Location API のエラーメッセージの更新

2023 年 8 月 1 日から、Amazon Location チームは次の表に示すように API エラーメッセージを変更しています。エラーコードの変更はありません。アプリケーションが正確なエラーメッセージ文字列に依存している場合は、新しい文字列でアプリケーションを更新する必要があります。質問や問題については、 にお問い合わせください AWS Support。

トピック

- 場所
- マップ
- トラッカー
- ルート
- メタデータ
- ジオフェンス

場所

場所

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
500	InternalServerException	Internal Server Exception	Internal server error. Try again later.
404	ResourceNotFoundException	resource <PlaceIndexName> not found, reason: <Reason> Resource '<PlaceIndexName>' not found placeIdx<PlaceIndexName> not found, reason: <Reason> no place index with name '%s' found	Place index not found: <PlaceIndexName>.
404	ResourceNotFoundException	place not found	Place not found: <PlaceId>.
400	ValidationException	PlaceIndex <PlaceIndexName> cannot be used for SearchPlaceIndexForSuggestions because it has IntendedUse <IntendedUse>	A place index with 'IntendedUse' set to Storage does not support 'SearchPlaceIndexForSuggestion' operation.
400	ValidationException	only one of 'BiasPosition' or 'FilterBBox' may be set	Only one of 'BiasPosition' or 'FilterBBox' may be set.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	BiasPosition must have exactly 2 entries	'BiasPosition' must have exactly 2 entries.
400	ValidationException	BiasPosition[0] must be between -180 and 180	'BiasPosition[0]' must be between -180 and 180.
400	ValidationException	BiasPosition[1] must be between -90 and 90	'BiasPosition[1]' must be between -90 and 90.
400	ValidationException	FilterBBox must have exactly 4 entries	'FilterBBox' must have exactly 4 entries.
400	ValidationException	FilterBBox[0] must be between -180 and 180	'FilterBBox[0]' must be between -180 and 180.
400	ValidationException	FilterBBox[1] must be between -90 and 90	'FilterBBox[1]' must be between -90 and 90.
400	ValidationException	FilterBBox[2] must be between -180 and 180	'FilterBBox[2]' must be between -180 and 180.
400	ValidationException	FilterBBox[3] must be between -90 and 90	'FilterBBox[3]' must be between -90 and 90.
400	ValidationException	FilterBBox must have more southwesterly point before more northeasterly point	'FilterBBox' must have more southwesterly position before more northeasterly position.
400	ValidationException	Position must have exactly 2 entries	'Position' must have exactly 2 entries.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	Position[0] must be between -180 and 180	'Position[0]' must be between -180 and 180.
400	ValidationException	Position[1] must be between -90 and 90	'Position[1]' must be between -90 and 90.
400	ValidationException	Language is not a valid BCP 47 language tag	'Language' must comply with the BCP 47 Language Tag standard, but was set to <GivenValue>. For more information, see https://wikipedia.org/wiki/IETF_language_tag.
400	ValidationException	'placeID' is invalid	'PlaceId' must be a valid ID.
400	ValidationException	no customer account ID parameter found	'RequesterAccountID' is a required field.
400	ValidationException	Invalid token	'NextToken' must be a valid token.
400	ValidationException	Expired token	'NextToken' must not be expired.
400	ValidationException	unsupported price plan '<PricingPlan>'	'PricingPlan' must be set to RequestBasedUsage.
400	ValidationException	'DataSource' must be one of: Here, Esri	'DataSource' must be one of Esri, Grab, Here.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	Grab is only supported in the ap-southeast-1 region	'DataSource' Grab must only be used in following regions: ap-southeast-1.
400	ValidationException	'IntendedUse' and 'PricingPlan' must both be provided to update either property	'IntendedUse' and 'PricingPlan' must both be provided to update either attribute .
402	ServiceQuotaExceededException	Place resources per account exceeded quota limits. For more info, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/	Place index resources have exceeded the quota per account per region. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/.
409	ConflictException	Resource already exists	Place index already exists: <PlaceIndexName>.

マップ

マップ

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
500	InternalServerException	Internal Server Exception	Internal server error. Try again later.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
		unable to find style template Error fetching style was not able to serialize the map style file	
404	ResourceNotFoundException	Map not found	Map not found: <MapName>.
404	ResourceNotFoundException	Sprites are not supported for this resource	Sprite not found: <SpriteName>.
400	ValidationException	Resource name should be set	'MapName' is a required field.
400	ValidationException	Must provide a valid number for start and end of Range	Font Unicode range start and end numbers must both be provided.
400	ValidationException	Start of range is an invalid number: <StartValue>	Start of font Unicode range must be a valid number.
400	ValidationException	End of range is an invalid number: <StartValue>	End of font Unicode range must be a valid number.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	End of range must be exactly 255 higher from start of range, difference found: <Difference>	The difference between the start and end of the font Unicode range must be exactly 255. Difference found: <Difference>.
400	ValidationException	Start of range must be a multiple of 256, found <StartValue>	Start of font Unicode range must be a multiple of 256, but was set to: <StartValue>.
400	ValidationException	Request font is empty	'FontStack' is a required field.
400	ValidationException	Request font is not valid for the datasource <DataSource>	<FontStack> is not a supported font stack for data source <DataSource>. For more information about the list of supported font stacks, see https://docs.aws.amazon.com/location/latest/APIReference/API_GetMapGlyphs.html.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	Request font is not valid	<FontStack> is not a supported font stack for data source <DataSource>. For more information about the list of supported font stacks, see https://docs.aws.amazon.com/location/latest/APIReference/API_GetMapGlyphs.html.
400	ValidationException	DataSource is invalid: <DataSource>	'DataSource' must be one of Esri, Grab, Here.
400	ValidationException	Request filename is empty	'FileName' is a required field.
400	ValidationException	Request filename is not valid	<SpriteFile> is not a supported sprite file name. For more information about the list of supported sprite file names, see https://docs.aws.amazon.com/location/latest/APIReference/API_GetMapSprites.html.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	Filename is invalid: <FileName>	<SpriteFile> is not a supported sprite file name. For more information about the list of supported sprite file names, see https://docs.aws.amazon.com/location/latest/APIReference/API_GetMapSprites.html.
400	ValidationException	Filename is an invalid content type: <FileName>	<SpriteFile> is not a supported sprite file name. For more information about the list of supported sprite file names, see https://docs.aws.amazon.com/location/latest/APIReference/API_GetMapSprites.html.
400	ValidationException	Filename is invalid: <FileName>	'Filename' must not be empty.
400	ValidationException	y-coordinate part of 'Y' must be a valid integer	y- coordinate part of 'Y' must be an integer.
400	ValidationException	tile resolution part of 'Y' must be a valid integer followed by 'x'	Tile resolution part of 'Y' must be an integer followed by 'X'.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	file type extension part of 'Y' must not be empty if a '.' is present	File type extension part of 'Y' must not be empty if a '.' is present.
400	ValidationException	'Z' must be a valid integer	'Z' must be an integer.
400	ValidationException	'X' must be a valid integer	'X' must be an integer.
400	ValidationException	'Z' must not be less than minimum zoom of style '<Style>' (<Minimum Value>)	'Z' must not be less than minimum zoom of style <Style> (<MinimumValue>).
400	ValidationException	'Z' must not be greater than maximum zoom of style '<Style' (<Maximum Value>)	'Z' must not be greater than maximum zoom of style Style (<MaximumValue>).
400	ValidationException	'Z' value not supported	'Z' must be between 0 and 63.
400	ValidationException	tile resolution part of 'Y' must be omitted because '<Style>' is a vector style	Tile resolution part of 'Y' must be omitted for style <Style>.
400	ValidationException	tile resolution part of 'Y' must be at least 1	Tile resolution part of 'Y' must be at least 1.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	tile resolution part of 'Y' must not be greater than max resolution of style '<Style>' (<Maximum Resolution>)	Tile resolution part of 'Y' must not be greater than maximum resolution of style <Style> (max <MaxResolution>).
400	ValidationException	file type extension part of 'Y' must be one of <SupportedFileFormats> (or may be omitted) for style '<Style>'	File type extension part of 'Y' must be one of <SupportedFileFormats> (or may be omitted) for style <Style>.
400	ValidationException	file type extension part of 'Y' must be omitted for style '<Style>'	File type extension part of 'Y' must be omitted for style <Style>.
400	ValidationException	y-coordinate part of 'Y' must be an integer in the range 0..2^Zoom -1 (0..<MaxTileCoordinate>)	y-coordinate part of 'Y' must be an integer in the range 0..2^Zoom -1 (0..<MaxTileCoordinate>).
400	ValidationException	'DataSource' must be one of: Here, Esri	'DataSource' must be one of Esri, Grab, Here.
400	ValidationException	unsupported price plan '<PricingPlan>'	'PricingPlan' must be set to RequestBasedUsage.
400	ValidationException	Invalid token	'NextToken' must be a valid token.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	Expired token	'NextToken' must not be expired.
400	ValidationException	Unsupported Map Style: <Style>	<Style> is not a supported map style. For more information about list of supported map styles, see https://docs.aws.amazon.com/location/latest/APIReference/API_MapConfiguration.html.
402	ServiceQuotaExceededException	Map resources per account exceeded quota limits. For more info, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/	Map resources have exceeded the quota per account per region. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/.
409	ConflictException	Resource already exists	Map already exists: <MapName>.

トラッカー

トラッカー

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
500	InternalServerException	Internal Server Exception internal server error unable to retrieve point from the storage unable to verify tracker Error processing List request	Internal server error. Try again later.
404	ResourceNotFoundException	tracker not found: <TrackerName> Tracker with name <TrackerName> was not found	Tracker not found: <TrackerName>.
404	ResourceNotFoundException	association not found: TrackerName <TrackerName>; and ConsumerArn < ConsumerArn >	Association between tracker <TrackerName> and consumer <ConsumerArn> is not found.
400	ValidationException	'ConsumerArn' must refer to a geofence collection resource	'ConsumerArn' must refer to a geofence collection resource.
400	ValidationException	'ConsumerArn' must refer to a resource	'ConsumerArn' must refer to a resource

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
		in the same region as the tracker it is associated to	in the same region as the tracker it is associated with.
400	ValidationException	'ConsumerArn' must refer to a resource in the same AWS account as the tracker is it associated to	'ConsumerArn' must refer to a resource in the same AWS account as the tracker it is associated with.
400	ValidationException	'DataSource' must be one of: Here, Esri	'DataSource' must be one of Esri, Grab, Here.
400	ValidationException	Nothing to update.	At least one of the following fields must be set: 'Description', 'PositionFiltering'
400	ValidationException	Invalid token	'NextToken' must be a valid token.
400	ValidationException	Expired token	'NextToken' must not be expired.
400	ValidationException	request.TrackerName not found on request	'TrackerName ' is a required field.
400	ValidationException	no deviceId parameter found	'DeviceId' is a required field.
400	ValidationException	unsupported price plan '<PricingPlan>'	'PricingPlan' must be set to RequestBasedUsage

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	Invalid token	'NextToken' must be a valid token.
400	ValidationException	Expired token	'NextToken' must not be expired.
400	ValidationException	provided start time is incorrect, should follow the format YYYY-MM-DDThh:mm:ss.sssZ“	'StartTimeInclusive' must follow the format YYYY-MM-DDThh:mm:ss.sssZ.
400	ValidationException	provided end time is incorrect, should follow the format YYYY-MM-DDThh:mm:ss.sssZ	'EndTimeExclusive' must follow the format YYYY-MM-DDThh:mm:ss.sssZ.
400	ValidationException	end time must be after start time	'EndTimeExclusive' must be after 'StartTimeInclusive'.
400	ValidationException	invalid key state	KMS key must be a symmetric Customer Master Key (CMK). Invalid state found. For more information about how key state affects the use of a KMS key, see https://docs.aws.amazon.com/kms/latest/developerguide/key-state.html.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	key not found	Invalid KMS key. '<KmsKeyId>' <KmsKeyIdValue> not found.
400	ValidationException	key is disabled	Symmetric Customer Master Key (CMK) must be enabled.
400	ValidationException	access denied	Symmetric Customer Master Key (CMK) must allow Amazon Location to create grants to its KMS key.
402	ServiceQuotaExceededException	Tracker <TrackerName> may not have more than <Max> consumer associations	Tracker resource may not have more than <Max> consumer associations. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
402	ServiceQuotaExceededException	Trackers per account exceeded quota limits. For more info, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/	Tracking resources have exceeded the quota per account per region. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/.
409	ConflictException	association already exists: TrackerName <TrackerName>; and ConsumerArn <ConsumerArn>	An association already exists between tracker <TrackerName> and consumer <ConsumerArn>.
409	ConflictException	Tracker already exists: <TrackerName>	Tracker already exists: <TrackerName>.

ルート

ルート

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
500	InternalServerException	Internal Server Exception	Internal server error. Try again later.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
404	ResourceNotFoundException	Resource not found	Route calculator not found: <RouteCalculatorName>.
400	ValidationException	Invalid token	'NextToken' must be a valid token.
400	ValidationException	Expired token	'NextToken' must not be expired.
400	ValidationException	'DataSource' must be one of: Here, Esri, Grab	'DataSource' must be one of Esri, Grab, Here.
400	ValidationException	<PricingPlan> pricing plan is not supported	'PricingPlan' must be set to RequestBasedUsage
400	ValidationException	unsupported price plan '<PricingPlan>'	'PricingPlan' must be set to RequestBasedUsage
400	ValidationException	Grab is only supported in the ap-southeast-1 region	'DataSource' <DataSourceName> must only be used in following regions: ap-southeast-1.
400	ValidationException	PricingPlan must be 'RequestBasedUsage'	'PricingPlan' must be set to RequestBasedUsage.
400	ValidationException	'DeparturePositions[0][0]' must be between -180 and 180	'DeparturePositions[0][0]' must be between -180 and 180.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	'DeparturePositions[0][1]' must be between -90 and 90	'DeparturePositions[0][1]' must be between -90 and 90.
400	ValidationException	'DestinationPositions[0][0]' must be between -180 and 180	'DestinationPositions[0][0]' must be between -180 and 180.
400	ValidationException	'DestinationPositions[0][1]' must be between -90 and 90.	'DestinationPositions[0][1]' must be between -90 and 90
400	ValidationException	'DepartNow' may not be true if 'DepartureTime' is set	Only one of 'DepartNow' or 'DepartureTime' may be set.
400	ValidationException	'<TravelModeOption>'' may not be set when 'TravelMode' has value <TravelModeOption>	'<TravelModeOption>' must not be set when 'TravelMode' has value <TravelModeOption>.
400	ValidationException	'CarModeOptions' may not be set when 'TravelMode' has value Walking	'CarModeOptions' must not be set when 'TravelMode' has value Walking.
400	ValidationException	'TruckModeOptions' may not be set when 'TravelMode' has value Walking	'TruckModeOptions' must not be set when 'TravelMode' has value Walking.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	'TruckModeOptions' may not be set when 'TravelMode' has value Car	'TruckModeOptions' must not be set when 'TravelMode' has value Car.
400	ValidationException	'CarModeOptions' may not be set when 'TravelMode' has value Truck	'CarModeOptions' must not be set when 'TravelMode' has value Truck.
400	ValidationException	At least one of [Height, Length, Width] must be set in 'TruckModeOptions.Dimensions'	At least one of the following attribute must be set in TruckModeOptions.Dimensions: Height, Length, Width.
400	ValidationException	At least one of [Total] must be set in 'TruckModeOptions.Weight'	At least one of the following attribute must be set in TruckModeOptions.Weight: Total.
400	ValidationException	'DeparturePositions' count must be 10 or less with DataSource set to Esri	'DeparturePositions' must have length at most 10 for 'DataSource' Esri.
400	ValidationException	'DestinationPositions' count must be 10 or less with DataSource set to Esri	'DestinationPositions' must have length at most 10 for 'DataSource' Esri.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	'DeparturePositions[0]' is more than 40km away from 'DestinationPositions[0]'	'DeparturePositions[0]' must not be more than 40 km away from 'DestinationPositions[0]'.
400	ValidationException	'DeparturePositions[0]' is more than 400km away from 'DestinationPositions[0]'	'DeparturePositions[0]' must not be more than 400 km away from 'DestinationPositions[0]'.
400	ValidationException	DeparturePositions[0] is contained within an unsupported region. Korea is not supported for CalculateRouteMatrix with the provider Esri.	DeparturePositions[0] is located in Korea, which is not supported when using CalculateRouteMatrix with data provider Esri.
400	ValidationException	'<HereTruckDimension>' must be between <Min> and <Max> <Unit>	'HereTruckDimension' must be between <Min> and <Max> <Unit>.
400	ValidationException	'WaypointPositions[0][0]' must be between -180 and 180	'WaypointPositions[0][0]' must be between -180 and 180.
400	ValidationException	'WaypointPositions[0][1]' must be between -90 and 90	'WaypointPositions[0][1]' must be between -90 and 90.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	'WaypointPositions[1][0]' must be between -180 and 180	'WaypointPositions[1][0]' must be between -180 and 180.
400	ValidationException	'WaypointPositions[1][1]' must be between -90 and 90	'WaypointPositions[1][1]' must be between -90 and 90.
400	ValidationException	No road segment could be matched for one or more coordinates within a radius (1km)	One or more provided positions are more than 1 km from the nearest road segment.
400	ValidationException	Some positions in the request are unreachable	Some positions in the request are unreachable.
400	ValidationException	Total distance between all waypoints must be not be greater than 40km for DataSource Esri when using TravelMode Walking	Total distance between all route positions must not be greater than 40 km for 'DataSource' Esri and 'TravelMode' Walking.
400	ValidationException	Total distance between all waypoints must be not be greater than 400km for DataSource Esri	Total distance between all route positions must not be greater than 400 km for 'DataSource' Esri.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	Following positions in the request are unreachable: <UnreachablePositions>	The following positions are unreachable: <UnreachablePositions>.
400	ValidationException	'DepartureTime' contains a badly-formatted timestamp	'DepartureTime' must follow the format YYYY-MM-DDThh:mm:ss.sssZ.
400	ValidationException	'TravelMode' <TravelMode> is not supported by <DataProvider>	'TravelMode' <TravelMode> not supported by data provider <DataProvider>.
400	ValidationException	'DeparturePositions' must be set	'DeparturePositions' must not be empty.
400	ValidationException	'DestinationPositions' must be set	'DestinationPositions' must not be empty.
400	ValidationException	Some inputs in the request are invalid	Some inputs in the request are invalid.
400	ValidationException	No route found between position <FirstPosition> and position <SecondPosition>	No route found between position <FirstPosition> and position <SecondPosition>.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	No route found	No route found. For more information, see https://developer.here.com/documentation/routing-api/dev_guide/topics/notice.html.
400	ValidationException	No route found	No route found.
402	ServiceQuotaExceededException	Route calculators per account exceeded quota limits. For more info, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/	Route calculator resources have exceeded the quota per account per region. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/.
409	ConflictException	Resource already exists	Route calculator already exists: <RouteCalculatorName>.

メタデータ

メタデータ

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
500	InternalServerException	Internal Server Error Error processing List request	Internal server error. Try again later.
404	ResourceNotFoundException	APIKey not found	Api key not found: <APIKeyName>.
404	ResourceNotFoundException	APIKeyID not found	ApiKeyId not found: <APIKeyID>.
400	ValidationException	Either ExpireTime or NoExpiry must be provided	At least one of the following fields must be set: 'ExpireTime', 'NoExpiry'.
400	ValidationException	NoExpiry cannot be set to false if no ExpireTime is provided	'ExpireTime' must be set when 'NoExpiry' has value false.
400	ValidationException	ExpireTime cannot be set if NoExpiry is true	'ExpireTime' must not be set when 'NoExpiry' has value true.
400	ValidationException	Expire time '<ExpireTimeValue>' is not a valid time format	'ExpireTime' must follow the format YYYY-MM-DDThh:mm:ss.sssZ.
400	ValidationException	Expire time '<ExpireTimeValue>' cannot	'ExpireTime' must not be in the past.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
		be in the past when creating a key	
400	ValidationException	Invalid token	'NextToken' must be a valid token.
400	ValidationException	Expired token	'NextToken' must not be expired.
400	ValidationException	The API Key %s has been recently used and the requested update may impact current usage. Specify ForceUpdate=true to update the API Key configuration.	This update may cause some users to lose API access. Because this API Key has been used in the last 7 days, you must set 'ForceUpdate' to true to confirm this change.
400	ValidationException	Expire time '<ExpireTimeValue>' must not be more than 1 minute in the past	'ExpireTime' must not be more than 1 minute in the past.
400	ValidationException	Description, ExpireTime, NoExpiry and Restrictions can't all be empty	At least one of the following fields must be set: 'Description', 'ExpireTime', 'NoExpiry', 'Restrictions'.
400	ValidationException	API Key expired	'ApiKeyId' must not be expired.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
409	ConflictException	API key named <APIKeyName> already exists	Api key already exists: <APIKeyName>.

ジオフェンス

ジオフェンス

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
500	InternalServerException	internal server error Internal server error Unsupported geofence geometry encountered geometry marshal error geometry load error unable to get geofence collection unable to delete geofences unable to retrieve geofence Error processing List request	Internal server error. Try again later.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
404	ResourceNotFoundException	collection not found: <GeofenceCollectionName> <GeofenceCollectionName> geofence collection not found Resource not found error no geofence with given name found	Geofence Collection not found: <GeofenceCollectionName>.
400	ValidationException	unsupported price plan '<PricingPlan>'	'PricingPlan' must be set to RequestBasedUsage.
400	ValidationException	KMS key must be a symmetric CMK. Invalid usage type: <UsageType>	KMS key must be a symmetric Customer Master Key (CMK). Invalid usage type <UsageType>. For how to create a symmetric CMK, refer to https://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html#create-symmetric-cmk.
400	ValidationException	Invalid token	'NextToken' must be a valid token.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	Expired token	'NextToken' must not be expired.
400	ValidationException	PricingPlanDataSource cannot be updated without updating PricingPlan	'PricingPlan' must be provided to update 'PricingPlanDataSource'.
400	ValidationException	nothing to update	At least one of the following fields must be set: 'Description'
400	ValidationException	invalid key state	KMS key must be a symmetric Customer Master Key (CMK). Invalid state <InvalidState>. For more information about how key state affects the use of a KMS key, see https://docs.aws.amazon.com/kms/latest/developerguide/key-state.html.
400	ValidationException	key not found	Invalid KMS key. '<KmsKeyId>' <KmsKeyIdValue> not found.
400	ValidationException	key is disabled	Symmetric Customer Master Key (CMK) must be enabled.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	access denied	Symmetric Customer Master Key (CMK) must allow Amazon Location to create grants to its KMS key.
400	ValidationException	duplicate geofence ID in batch	'GeofenceId' <DuplicatedGeofenceID> is duplicated in batch.
400	ValidationException	missing GeofenceId	'GeofenceId' must not be empty.
400	ValidationException	Invalid token	'NextToken' must be a valid token.
400	ValidationException	Expired token	'NextToken' must not be expired.
400	ValidationException	Position[0] must be between -180 and 180	'Position[0]' must be between -180 and 180.
400	ValidationException	Position[1] must be between -90 and 90	'Position[1]' must be between -90 and 90.
400	ValidationException	radius must be less than or equal to 1000km	'Geometry.Circle.Radius' must be less than or equal to 1000km.
400	ValidationException	no geofence with given name found	Geofence not found: <CollectionName>.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	Geometry must contain either a Circle or Polygon, not both	Only one of 'Circle' or 'Polygon' may be set within 'Geometry'.
400	ValidationException	Geometry must contain a Polygon or a Circle	One of 'Polygon' or 'Circle' must be set within 'Geometry'.
400	ValidationException	radius must be greater than 0m	'Geometry.Circle.Radius' must be greater than 0m.
400	ValidationException	empty polygon	'Geometry.Polygon' must not be empty.
400	ValidationException	empty polygon ring	'Geometry.Polygon' must not be empty.
400	ValidationException	circle can not cross antimeridian	'Geometry.Circle' must not cross antimeridian. Cut it in two such that neither part's representation crosses the antimeridian.
400	ValidationException	polygon can not cross antimeridian	'Geometry.Polygon' must not cross antimeridian. Cut it in two such that neither part's representation crosses the antimeridian.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	polygon can not have interior rings (holes), remove holes	'Geometry.Polygon' must not have interior rings (holes). For more information about interior rings see https://www.rfc-editor.org/rfc/rfc7946.html#appendix-A.3.
400	ValidationException	polygon ring is not closed	'Geometry.Polygon' contains an open ring. Close the ring by ensuring the first and last positions are equal.
400	ValidationException	polygon ring has more than 1000 vertices	'Geometry.Polygon' must not have more than 1000 vertices.
400	ValidationException	polygon ring has fewer than 4 positions	Number of vertices in 'Geometry.Polygon' must be greater or equal to 4.
400	ValidationException	invalid center	'Geometry.Circle.Center' must be a valid position (longitude/latitude pair).
400	ValidationException	radius must be greater than 0m	'Geometry.Circle.Radius' must be greater than 0 m.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	longitude range should be between -180 and 180 degrees	Longitude must be between -180 and 180 degrees, but was set to <Provided Longitude>.
400	ValidationException	latitude range should be between -90 and 90 degrees	Latitude must be between -90 and 90 degrees, but was set to <Provided Longitude>.
400	ValidationException	polygon exterior ring is expected to be counter clockwise	'Geometry.Polygon' must be oriented counter-clockwise.
400	ValidationException	polygon interior ring should be clockwise oriented	'Geometry.Polygon' must be oriented clockwise.
400	ValidationException	radius must be less than or equal to 1000km	'Geometry.Circle.Radius' must be less than or equal to 1000 km.
400	ValidationException	timestamp.Parse() error	'SampleTime' must follow the format YYYY-MM-DDThh:mm:ss.sssZ.
400	ValidationException	invalid input	'SourceArn' must refer to a tracker resource.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
400	ValidationException	arn: invalid prefix	'SourceArn' must be a valid ARN. For more information, see https://docs.aws.amazon.com/general/latest/gr/AWS-arns-and-namespaces.html.
400	ValidationException	arn: not enough sections	'SourceArn' must be a valid ARN. For more information, see https://docs.aws.amazon.com/general/latest/gr/AWS-arns-and-namespaces.html.
400	ValidationException	invalid resource part	'SourceArn' must refer to a tracker resource.
402	ServiceQuotaExceededException	Geofence collections per account exceeded quota limits. For more info, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/	Geofence collection resources have exceeded the quota per account per region. For more information, see https://aws.amazon.com/premiumsupport/knowledge-center/manage-service-limits/.

エラーコード	Exception	以前のエラーメッセージ	新しいエラーメッセージ
409	ConflictException	collection already exists: <GeofenceCollectionName>	Geofence Collection already exists: <GeofenceCollectionName>.
409	ConflictException	Resource conflict error	Geofence already exists: <GeofenceName>.

Amazon Location Service を使用するためのコードサンプルとチュートリアル

このトピックでは、Amazon Location Service を学習するのに役立つコードサンプル、チュートリアル、ブログ記事のリストを紹介します。各コードサンプルには、その仕組みの説明が含まれています。

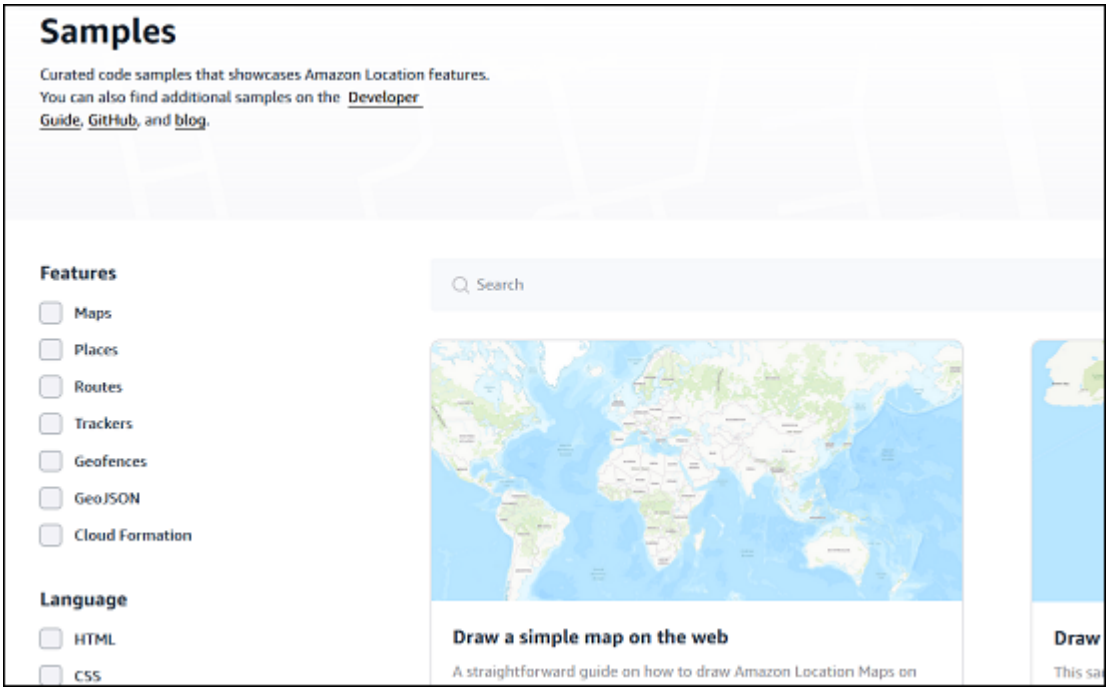
その他のサンプルは、AWS 地理 GitHub空間ページ 、AWS Amazon Location のサンプル GitHub ページ、ブログAWS サイト にあります。

> **Note**
>
> AWS 地理空間 GitHub ページと AWS サンプル GitHub ページの違いを理解しておくとよいでしょう。
>
> - 地理空間 GitHub – AWS 地理空間 GitHub ページには、Amazon Location Service チームが作成および管理するサンプルが含まれています。
> - サンプル GitHub – AWS Amazon Location のサンプル GitHub ページには、Amazon Location 用に作成されたサンプルが含まれていますが、アクティブにメンテナンスされているかどうかはわかりません。

クイックスタートのチュートリアルから始めることをおすすめします。他のサンプルを参照する際に基礎となる知識を網羅しています。

トピック

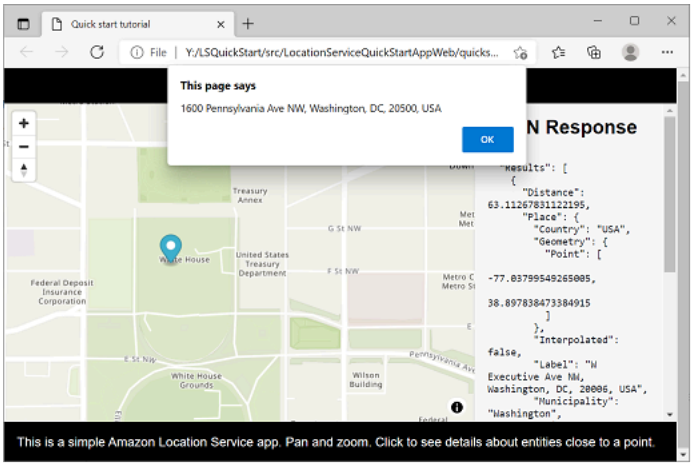
Amazon Location デモサイト

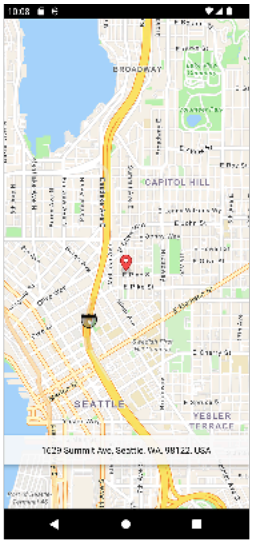


Amazon Location デモサイトでは、Amazon Location Service のソースコードを使用したデモを見ることができます。このサイトには、ホスト型ウェブデモと Android 用のデモアプリが含まれています。

また、サイトのサンプルページには、機能、言語、プラットフォームでフィルタリングできるさまざまなサンプルがあります。

チュートリアル：クイックスタート

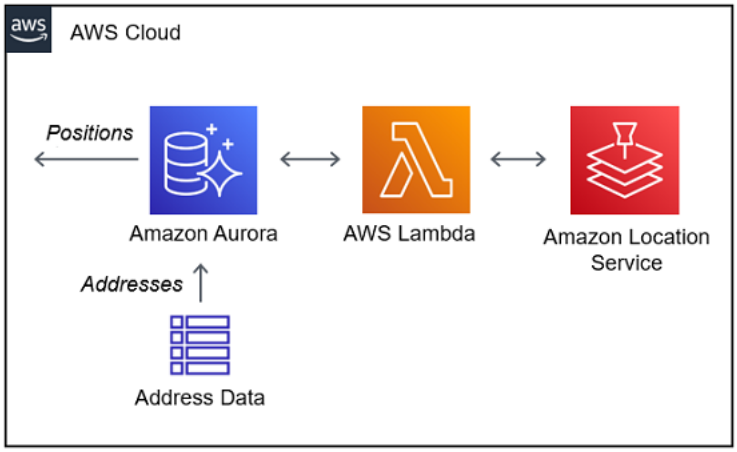




ウェブ、iOS、Android デバイスで利用できるクイックスタートチュートリアルがあります。このチュートリアルでは、プラットフォームごとに、アプリケーションにインタラクティブマップを追加する方法と、アプリケーションから Amazon Location Service APIs を呼び出す方法について説明します。このチュートリアルは、静的ウェブページ JavaScript の 、Android 電話アプリケーション用の Kotlin、または iOS アプリケーション用の Swift で利用できます。

- JavaScript 静的ウェブページのドキュメントリンク: ウェブ ACL の作成
- Android アプリケーションの Kotlin ドキュメントリンク: Amazon Location Service のクイックスタート
- iOS アプリの Swift ドキュメントリンク: iOS アプリケーションの作成

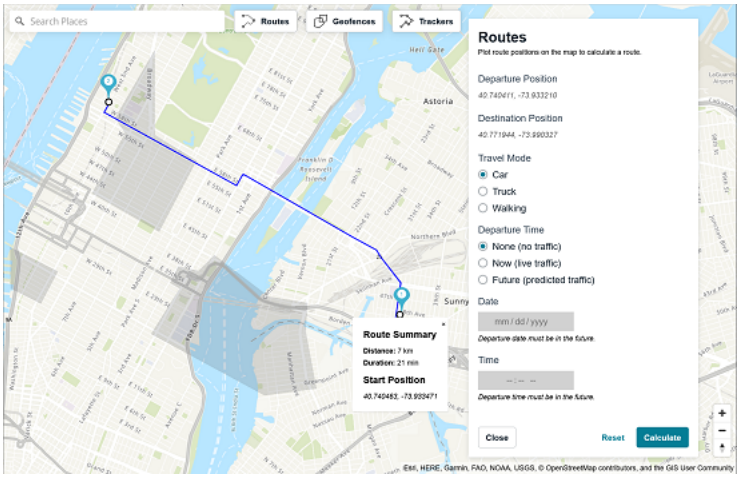
チュートリアル：データベースのエンリッチメント



このチュートリアルでは、 から呼び出される Amazon Location Service を使用してアドレス AWS Lambda を正規化し、Amazon Aurora データベースのレコードに緯度と経度を追加する方法を示します。Amazon Aurora と を使用します AWS Lambda。

ドキュメントのリンク：Amazon Aurora PostgreSQL ユーザー-Amazon Location Service 義関数

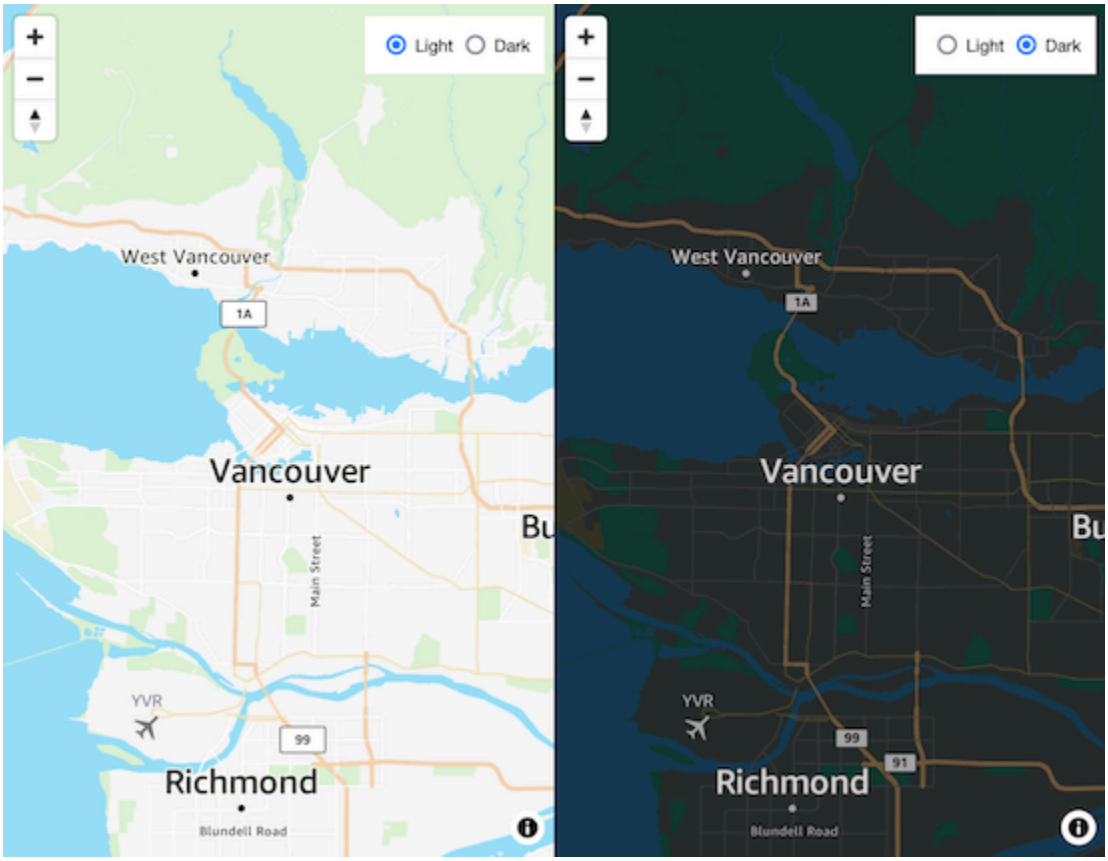
例：Explore アプリケーション



Amazon Location Service の機能について学ぶ最良の方法の 1 つは、Amazon Location コンソールの Explore 機能を使用することです。この完全なウェブアプリケーションの例は、コンソールのマップ、場所、ルート、ジオフェンス、トラッカーの機能を模倣して、これらの機能を独自のアプリで再現する方法を示しています。Amplify、React、および を使用します JavaScript。

サンプル GitHub リンク: サンプルアプリケーションを調べる

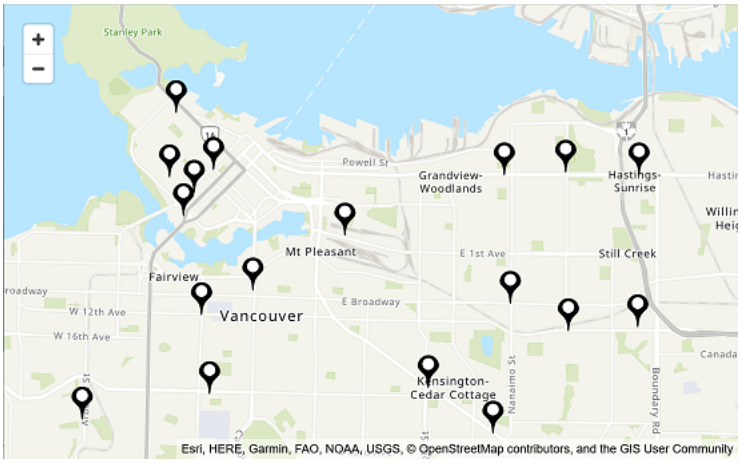
例：マップのスタイル設定



このコード例は、 MapLibre で を使用して衛星マップとベクトルロードマップを切り替える方法を示しています JavaScript。 MapLibre、Amazon Location 認証ヘルパー、および を使用します JavaScript。

地理空間 GitHub リンク: スタイル切り替えによるインタラクティブマップ

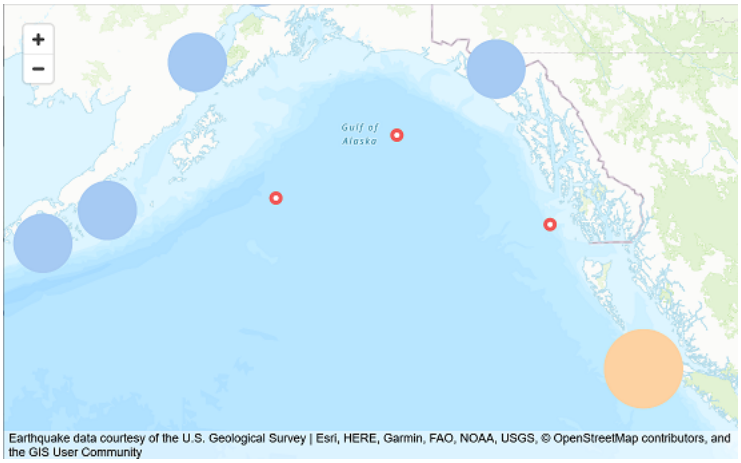
例：マーカーの描画



このコード例は、カナダのブリティッシュコロンビア州バンクーバーにある Amazon Locker の場所を示しています。ポイントロケーションにマーカーを描画する方法を示しています。MapLibre、Node.js、React、Amazon Location 認証ヘルパー、および を使用します JavaScript。

地理空間 GitHub リンク: ポイントにマーカーがあるインタラクティブマップ

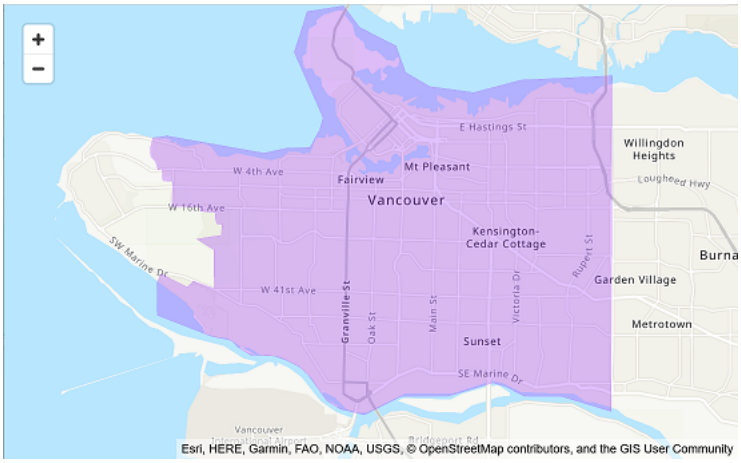
例：クラスター化されたポイントを描画



USGS の地震データを使用したこのコード例では、マップ上でポイントが近接している場合にまとめて描画する方法を示しています。 MapLibre、Node.js、React、Amplify、および を使用します JavaScript。

サンプル GitHub リンク: ポイントのクラスターを含むインタラクティブマップ

例：ポリゴンを描画



このコード例は、マップ上にポリゴンを描画する方法を示しています。MapLibre、Node.js、React、Amazon Location 認証ヘルパー、および を使用します JavaScript。

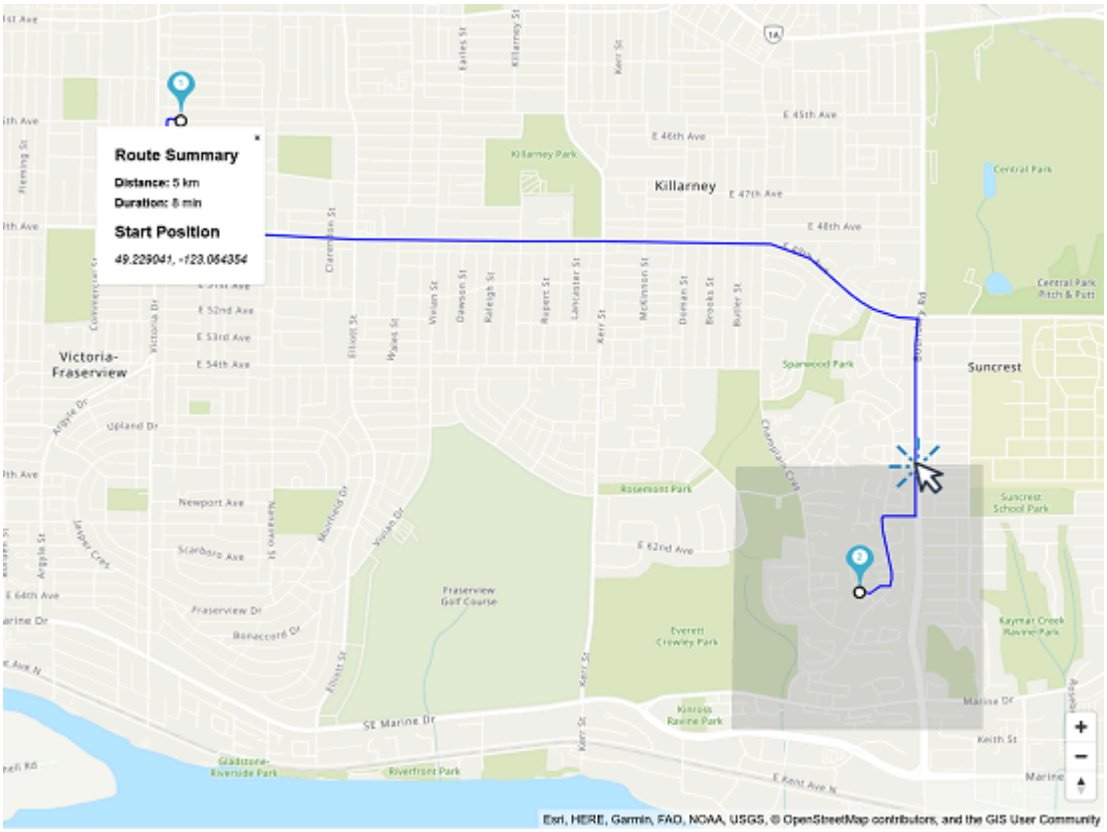
地理空間 GitHub リンク: ポリゴンを使用したインタラクティブマップ

例：マップ言語を変更

このコード例は、Amazon Location のマップの表示言語を変更する方法を示しています。Amplify、React、および を使用します MapLibre。

サンプル GitHub リンク: マップ言語の変更サンプル

ブログ：配達予定時刻を通知



このブログ記事では、配達予定時刻を顧客に通知するさまざまな方法を紹介しています。ルートを使用して推定運転時間を表示し、次にトラッカーとジオフェンスを使用してドライバーが顧客に近づいたときに通知する方法を説明しています。Amplify、React、Amazon EventBridge、Amazon Simple Notification Service (Amazon SNS) を使用します。

ブログリンク：到着予定時刻と近接を通知

例: ストリーム位置の更新

Kinesis Stream To Tracker App: このサンプルは、Kinesis Data Stream を使用して Amazon Location Service でトラッカーの更新を投稿する方法を示しています。このサンプルは、Python で記述されたデプロイ可能な Lambda アプリケーションで、Kinesis Data Stream と統合して Kinesis イベントとバッチ更新デバイスの位置を消費できます。

リポジトリリンク: Amazon Location Amazon Kinesis Data Streams Stream to Tracker App

追跡とジオフェンスの詳細については、ジオフェンスとトラッカーのドキュメントを参照してください。デベロッパーは、AWS の Serverless Application Repository のドキュメントに従って、または Lambda コンソール からディクレットでアプリケーションをデプロイできます。

Device Position Streaming サンプルアプリケーション : このコード例は、デバイスの位置データを Kinesis Data Stream にストリーミングする方法と、ジオフェンス通知の仕組みを示しています。このアプリは、上記の Kinesis Stream to Tracker サンプルアプリが Amazon Location Service で更新されるストリーミングトラッカーの位置に対して実行されていることに依存します。

リポジトリリンク: Amazon Location Device Position Streaming サンプルアプリケーション

例: モバイルアプリケーションのジオフェンシングと追跡

このサンプルアプリケーションは、トラッカーとジオフェンスが Lambda AWS IoT と Amazon Location の機能を組み合わせてどのように相互作用するかを示しています。iOS と Android で利用できるチュートリアルがあります。

チュートリアルリンク: サンプルジオフェンスとトラッカーモバイルアプリケーション

Amazon Location Service の使い方

Amazon Location Service 機能を使って、地理的タスクや位置情報関連のタスクを実行することができます。これらのタスクを組み合わせて、ジオマーケティング、配送、資産追跡など、より複雑なユースケースに対応することができます。

アプリケーションに位置情報機能を組み込む準備ができたら、目標と好みに応じて、以下の方法を使って Amazon Location Service 機能を利用してください。

- 探索ツール — Amazon Location リソースを試してみたい場合は、以下のツールが API にアクセスして試す最も速い方法です。
  - Amazon Location コンソールには、さまざまなクイックアクセスツールが用意されています。Explore ページを利用して、リソースの作成、管理、API を試行することができます。コンソールは、下記の他の方法を使用するための準備として、リソース (通常は1回限りのタスク) を作成するのにも便利であります。
  - AWSコマンドラインインターフェイス (CLI) では、リソースを作成し、ターミナルを利用して Amazon Location API にアクセスすることができます。認証情報を利用して設定すると、AWS CLIが認証を処理します。
  - Amazon Location Service APIを使用したタスクの実行方法を示すコード例やチュートリアルを見ることができます。これには、コンソールのExploreページの機能の多くを模倣した例も含まれています。
- プラットフォーム SDKs – マップ上でのデータを可視化していない場合、AWS上で構築するためのAWS標準ツールのどれでも使うことができます。
  - C++、Go、Java、.NET JavaScript、Node.js、PHP、Python、Ruby の SDKs を使用できます。
- フロントエンド SDK とライブラリ — Amazon Location 使って、モバイル・プラットフォーム上でアプリケーションを構築したり、あらゆるプラットフォーム上でデータを地図上に可視化したい場合、次のような選択肢があります :
  - AWS Amplify ライブラリは、iOS 、Android https://docs.amplify.aws/guides/location-service/setting-up-your-app/q/platform/android、および JavaScriptウェブアプリケーション内で Amazon Location を統合します。
  - MapLibre ライブラリを使用すると、Amazon Location を使用して iOS 、Android 、およびJavaScriptウェブアプリケーションにクライアント側マップをレンダリングできます。
  - Tangram ES ライブラリを利用すれば、iOS と Android のウェブアプリケーション内で OpenGL ES を利用してベクターデータから 2D および 3D マップをレンダリングすることができます。ウェブアプリケーションにも JavaScriptTAKgram があります。

- HTTPS リクエストの直接送信 — SDK が利用ないプログラミング言語を使っている場合、またはAWSへのリクエストの送信方法をより細かくコントロールしたい場合は、署名バージョン 4 署名プロセスで認証された直接 HTTPS リクエストを送信することで Amazon Location にアクセスことができます。詳細については、AWS 全般のリファレンスの「Signature Version 4 の署名プロセス」を参照してください。

この章では、位置データを使うアプリケーションに共通する多くのタスクについて説明します。一般的なユースケースのセクションでは、これらを他のAWSサービスと組み合わせて、より複雑なユースケースを実現する方法について説明します。

トピック

Amazon Location Service を使用するための前提条件

このセクションでは、Amazon Location Service を使用するために必要な操作を説明します。利用するには、 AWS アカウント を持ち、Amazon Location へのアクセスが設定されている必要があります。

にサインアップする AWS アカウント

がない場合は AWS アカウント、次のステップを実行して作成します。

にサインアップするには AWS アカウント

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。セキュリティのベストプラクティスとして、ユーザーに管理アクセスを割り当て、ルートユーザーのみを使用してルートユーザーアクセスが必要なタスクを実行してください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。https://aws.amazon.com/ の [マイアカウント] を選んで、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理できます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、 を保護し AWS アカウントのルートユーザー、 を有効にして AWS IAM Identity Center、日常的なタスクにルートユーザーを使用しないように管理ユーザーを作成します。

のセキュリティ保護 AWS アカウントのルートユーザー

1. ルートユーザーを選択し、 AWS アカウント E メールアドレスを入力して、アカウント所有者AWS Management Consoleとして にサインインします。次のページでパスワードを入力します。

   ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドの「ルートユーザーとしてサインインする」を参照してください。
2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、「IAM ユーザーガイド」の AWS アカウント 「ルートユーザーの仮想 MFA デバイスを有効にする (コンソール)」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。
2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、「 ユーザーガイド」の「デフォルト でユーザーアクセス IAM アイデンティティセンターディレクトリを設定するAWS IAM Identity Center 」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

  IAM Identity Center ユーザーを使用してサインインする方法については、「 AWS サインインユーザーガイド」の AWS 「 アクセスポータルにサインインする」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。
2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「グループの参加」を参照してください。

Amazon Location Service へのアクセスを許可する

デフォルトでは、管理者権限のないユーザーには一切のアクセス許可がありません。Amazon Location にアクセスするには、特定のアクセス権限を含む IAM ポリシーをアタッチしてアクセス権限を付与する必要があります。リソースへのアクセスを許可する時は、最小特権の原則に従うようにしてください。

> **Note**
>
> 認証されていないユーザーに Amazon Location Service 機能 (ウェブベースのアプリケーションなど) へのアクセスを許可する方法については、「Amazon Location Service へのアクセスを許可する」を参照してください。

以下のポリシー例は、すべての Amazon Location オペレーションにアクセスする権限をユーザーに付与します。その他の例については、「Amazon Location Service のアイデンティティベースのポリシーの例」を参照してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "geo:*"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

アクセス権限を付与するには、ユーザー、グループ、またはロールにアクセス許可を追加します。

- のユーザーとグループ AWS IAM Identity Center :

  アクセス許可セットを作成します。「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」の手順に従ってください。

- IAM 内で、ID プロバイダーによって管理されているユーザー:

  ID フェデレーションのロールを作成します。詳細については、「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) 用のロールの作成」を参照してください。

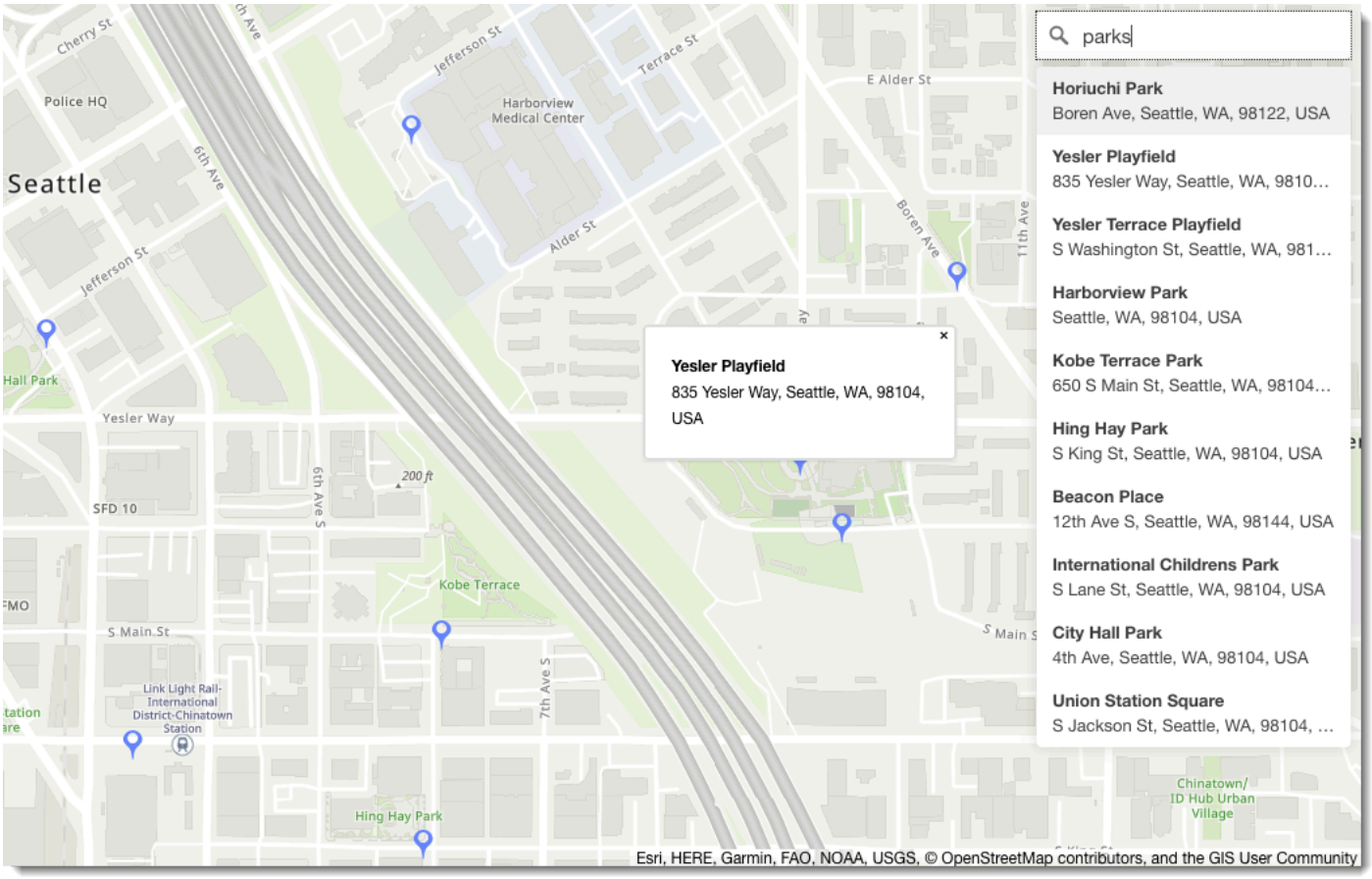
- IAM ユーザー:
  - ユーザーが担当できるロールを作成します。手順については、「IAM ユーザーガイド」の「IAM ユーザー用ロールの作成」を参照してください。

- (お奨めできない方法) ポリシーをユーザーに直接アタッチするか、ユーザーをユーザーグループに追加する。詳細については、「IAM ユーザーガイド」の「ユーザー (コンソール) へのアクセス権限の追加」を参照してください。

Amazon Location Service を使用するアプリケーションを作成する場合、認証されていないアクセスを一部のユーザーに許可する必要がある場合があります。これらのユースケースについては、「Amazon Cognito を使用した非認証アクセスの有効化」を参照してください。

アプリケーションで Amazon Location マップを使用する

Amazon Location マップは費用対効果が高く、インタラクティブです。アプリケーション内の既存のマップを置き換えてコストを節約したり、新しいマップを追加して店舗の場所などのロケーションベースのデータを視覚的に表示したりできます。



Amazon Location Service では、マップリソースを作成して設定することで、マップ操作のデータプロバイダーを選択できます。マップリソースに、データプロバイダーと、マップのレンダリングに使用されるスタイルを設定できます。

リソースを作成したら、AWS SDK から直接リクエストを送信するか、環境内のマップのレンダリング専用に作成されたライブラリを使用して送信できます。

> **Note**
>
> マップの概念の概要については、「マップ」を参照してください。

トピック

- 前提条件
- アプリケーションにマップを表示する
- マップ上にデータフィーチャを描画する
- を使用してマップの拡張を設定する MapLibre
- マップリソースの管理

前提条件

アプリケーションにマップを表示する前に、前提条件となる手順に従ってください。

トピック

- マップリソースを作成する
- リクエストを認証する

マップリソースを作成する

アプリケーションでマップを使用するには、マップで使用するマップスタイルとデータプロバイダーを指定するマップリソースが必要です。

> **Note**
>
> 配送車両や従業員など、ビジネスで使用する資産をアプリケーションで追跡またはルーティングする場合、位置情報プロバイダーに Esri を使用してはいけません。詳細については、「AWS サービス規約」のセクション 82 を参照してください。

Amazon Location Service コンソール、AWS CLI、または Amazon Location API を使用して、マップリソースを作成できます。

Console

Amazon Location Service コンソールを使用してマップリソースを作成するには

1. Amazon Location コンソールの マップ ページで、マップを作成 を選択してマップスタイルをプレビューします。
2. 新しいマップリソースの名前と説明を追加します。
3. マップスタイルを選択します。

> **Note**
>
> 配送車両や従業員など、ビジネスで使用する資産をアプリケーションで追跡またはルーティングする場合、位置情報プロバイダーに Esri を使用してはいけません。詳細については、「AWS サービス規約」のセクション 82 を参照してください。

4. 政治的見解から使用するものを選択します。
5. Amazon Location の利用規約に同意し、[マップを作成] を選択します。選択したマップを、拡大、縮小、または任意の方向への画面移動をすることができます。
6. ユーザーがスタイルを切り替えられるようにする (たとえば、衛星画像とベクタースタイルを切り替えられるようにする) には、スタイルごとにマップリソースを作成する必要があります。

   使用したくないマップスタイルを含むリソースは、コンソールのマップホームページから削除できます。

API

Amazon Location API を使用してマップリソースを作成するには

Amazon Location API `CreateMap` のオペレーションを使用してください。

次の例は、 マップスタイル*`ExampleMap`*を使用して というリソース*`VectorEsriStreets`*マップリソースを作成する API リクエストです。

```
POST /maps/v0/maps HTTP/1.1
```

```
Content-type: application/json

{
   "Configuration": {
      "Style": "VectorEsriStreets"
   },
   "MapName": "ExampleMap"
   }
}
```

> **Note**
>
> 配送車両や従業員など、ビジネスで使用する資産をアプリケーションで追跡またはルーティングする場合、位置情報プロバイダーに Esri を使用してはいけません。詳細については、「AWS サービス規約」のセクション 82 を参照してください。

AWS CLI

AWS CLI コマンドを使用してマップリソースを作成するには

create-mapコマンドを実行します。

次の例では、マップスタイル*VectorEsriStreets*として *ExampleMap*を使用して というマップリソースを作成します。

```
aws location \
  create-map \
  --configuration Style="VectorEsriStreets" \
  --map-name "ExampleMap"
```

> **Note**
>
> 配送車両や従業員など、ビジネスで使用する資産をアプリケーションで追跡またはルーティングする場合、位置情報プロバイダーに Esri を使用してはいけません。詳細については、「AWS サービス規約」のセクション 82 を参照してください。

リクエストを認証する

マップリソースを作成し、位置情報をアプリケーションに組み込む準備ができたら、リクエストの認証方法を選択する必要があります。

> **Note**
>
> ほとんどのマップフロントエンドアプリケーションでは、Amazon Location Service のマップやその他の機能への認証されていないアクセスが必要です。アプリケーションによっては、AWS Signature v4 を使用してリクエストを認証したり、Amazon Cognito または Amazon Location API キーを使用して認証なしでリクエストしたりできます。これらのオプションの詳細については、「Amazon Location Service へのアクセスを許可する」を参照してください。

アプリケーションにマップを表示する

このセクションでは、Amazon Location API を使用する場合に、マップレンダリングツールを使用してモバイルアプリケーションまたはウェブアプリケーションにマップを表示する方法についてのチュートリアルを提供します。Amazon Location Service の使い方 トピックで説明したように、Amplify、、 MapLibreTangram など、Amazon Location でマップをレンダリングするときに使用するライブラリを選択できます。

アプリケーションにマップを表示するには、次のいずれかを実行します。

- ウェブおよびモバイルフロントエンドアプリケーションにマップを表示する最も直接的な方法は、を使用することです MapLibre。MapLibre チュートリアルまたはクイックスタートチュートリアルに従って、 の使用方法を学ぶことができます MapLibre。
- すでにAWS Amplify をお使いの場合は、Amplify Geo SDK を使用することをお勧めします。詳細については、「 Amplify チュートリアル」をご覧ください。
- すでに Tangram のユーザーで、Amazon Location Service への移行中も Tangram を引き続き使用してマップのレンダリングを行いたい場合は、「 Tangram チュートリアル」に従ってください。

トピック

- Amazon Location Service での MapLibre ライブラリの使用
- Amazon Location Service で Amplify ライブラリを使用する
- Amazon Location Service で Tagram を使用する

Amazon Location Service での MapLibre ライブラリの使用

以下のチュートリアルでは、Amazon Location で MapLibre ライブラリを使用する方法について説明します。

トピック

- Amazon Location Service での MapLibre GL JS の使用
- Amazon Location Service での Android 用 MapLibre ネイティブ SDK の使用
- Amazon Location Service での iOS 用 MapLibre ネイティブ SDK の使用

Amazon Location Service での MapLibre GL JS の使用

MapLibre GL JS を使用して、クライアント側のマップをウェブアプリケーションに埋め込みます。

MapLibre GL JS は、Amazon Location Service Maps API が提供するスタイルやタイルと互換性のあるオープンソース JavaScript ライブラリです。 MapLibre GL JS を基本的な HTML または JavaScript アプリケーションに統合して、カスタマイズ可能で応答性の高いクライアント側マップを埋め込むことができます。

このチュートリアルでは、基本的な HTML および JavaScript アプリケーション内で MapLibre GL JS を Amazon Location と統合する方法について説明します。このチュートリアルで紹介するライブラリやテクニックが、React や Angular などのフレームワークにも適用可能です。

このチュートリアルのサンプルアプリケーションは、 の Amazon Location Service サンプルリポジトリの一部として利用できますGitHub。

アプリケーションの構築： Scaffolding

このチュートリアルでは、 を使用して JavaScript HTML ページにマップを構築するウェブアプリケーションを作成します。

まず、マップのコンテナを含む HTML ページ (`index.html`) を作成します。

- `map`の `id` が付いた `div` 要素を入力し、マップの寸法をマップビューに適用します。寸法はビューポートから継承されます。

```
<html>
  <head>
    <style>
      body {
```

```
        margin: 0;
      }

      #map {
        height: 100vh; /* 100% of viewport height */
      }
    </style>
  </head>
  <body>
    <!-- map container -->
    <div id="map" />
  </body>
</html>
```

アプリケーションの構築：依存関係の追加

次の依存関係をアプリケーションに追加します。

- MapLibre GL JS (v3.x)、および関連する CSS。
- Amazon Location JavaScript 認証ヘルパー。

```
<!-- CSS dependencies -->
<link
  href="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.css"
  rel="stylesheet"
/>
<!-- JavaScript dependencies -->
<script src="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.js"></script>
<script src="https://unpkg.com/@aws/amazon-location-authentication-helper.js"></script>
<script>
  // application-specific code
</script>
```

これで、マップのコンテナを含む空のページが作成されます。

アプリケーションの構築：設定

を使用してアプリケーションを設定するには JavaScript：

1. リソースの名前と ID を入力します。

```
// Cognito Identity Pool ID
```

```
const identityPoolId = "us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd";
// Amazon Location Service Map name
const mapName = "ExampleMap";
```

2. 「マップを使用する - ステップ 2. 認証を設定する」で作成した認証されていない アイデンティティプールを使用して、認証情報プロバイダーをインスタンス化します。これをinitializeMap という関数に入れます。この関数には、次のステップで追加する他のマップ初期化コードも含まれます。

```
// extract the Region from the Identity Pool ID; this will be used for both Amazon
 Cognito and Amazon Location
AWS.config.region = identityPoolId.split(":")[0];

async function initializeMap() {
  // Create an authentication helper instance using credentials from Cognito
  const authHelper = await
 amazonLocationAuthHelper.withIdentityPoolId(identityPoolId);

  // ... more here, later
}
```

アプリケーションの構築：マップの初期化

ページが読み込まれた後にマップを表示するには、マップを初期化する必要があります。マップの初期位置を調整したり、コントロールを追加したり、データをオーバーレイしたりできます。

```
async function initializeMap() {
  // Create an authentication helper instance using credentials from Cognito
  const authHelper = await amazonLocationAuthHelper.withIdentityPoolId(identityPoolId);

  // Initialize the map
  const map = new maplibregl.Map({
    container: "map",
    center: [-123.1187, 49.2819], // initial map centerpoint
    zoom: 10, // initial map zoom
    style: 'https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/style-
descriptor',
    ...authHelper.getMapAuthenticationOptions(), // authentication, using cognito
  });

  map.addControl(new maplibregl.NavigationControl(), "top-left");
}
```

```
initializeMap();
```

> **Note**
>
> 使用する各データプロバイダーのワードマークまたはテキストアトリビューションを、アプリケーションまたはドキュメントに記載する必要があります。属性文字列は、`sources.esri.attribution`、、`sources.here.attribution`および`sources.grabmaptiles.attribution`キーのスタイル記述子レスポンスに含まれます。MapLibre GL JS は自動的に属性を提供します。Amazon Location リソースをデータプロバイダーと併用する場合は、サービスの利用規約を必ずお読みください。

Java アプリケーションを実行する

このサンプルアプリケーションは、ローカル Web サーバーで使用するか、ブラウザで開いて実行できます。

ローカル Web サーバーで使用するには、npx を使用できます。Node.js の一部としてインストールされるためです。`npx serve` は、`index.html`と同じディレクトリ内で使用できます。これによってアプリケーションに `localhost:5000` が提供されます。

> **Note**
>
> 認証されていない Amazon Cognito ロール用に作成したポリシーに `referer` 条件が含まれている場合、`localhost:` のURL によるテストがブロックされる可能性があります。この場合、ポリシーに含まれる URL を提供する Web サーバーでテストできます。

チュートリアルを完了すると、最終的なアプリケーションは次の例のようになります。

```
<!-- index.html -->
<html>
  <head>
    <link href="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.css"
 rel="stylesheet" />
    <style>
      body {
        margin: 0;
      }
```

```
      #map {
        height: 100vh;
      }
    </style>
  </head>

  <body>
    <!-- map container -->
    <div id="map" />
    <!-- JavaScript dependencies -->
    <script src="https://unpkg.com/maplibre-gl@3.x/dist/maplibre-gl.js"></script>
    <script src="https://unpkg.com/@aws/amazon-location-authentication-helper.js"></
script>
    <script>
      // configuration
      const identityPoolId = "us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd"; //
 Cognito Identity Pool ID
      const mapName = "ExampleMap"; // Amazon Location Service Map Name

      // extract the region from the Identity Pool ID
      const region = identityPoolId.split(":")[0];

      async function initializeMap() {
        // Create an authentication helper instance using credentials from Cognito
        const authHelper = await
 amazonLocationAuthHelper.withIdentityPoolId(identityPoolId);

        // Initialize the map
        const map = new maplibregl.Map({
          container: "map",
          center: [-123.115898, 49.295868],
          zoom: 10,
          style: `https://maps.geo.${region}.amazonaws.com/maps/v0/maps/${mapName}/
style-descriptor`,
          ...authHelper.getMapAuthenticationOptions(),
        });
        map.addControl(new maplibregl.NavigationControl(), "top-left");
      }

      initializeMap();
    </script>
  </body>
</html>
```

このアプリケーションを実行すると、選択したマップスタイルを使用して全画面マップが表示されます。このサンプルは、 の Amazon Location Service サンプルリポジトリにありますGitHub。

Amazon Location Service での Android 用 MapLibre ネイティブ SDK の使用

MapLibre ネイティブ SDK を使用して、Android アプリケーションにインタラクティブマップを埋め込みます。

Android 用 MapLibre ネイティブ SDK は Mapbox Native に基づくライブラリであり、Amazon Location Service Maps API が提供するスタイルやタイルと互換性があります。 MapLibre Native SDK for Android を統合して、インタラクティブマップビューをスケーラブルでカスタマイズ可能なベクトルマップに Android アプリケーションに埋め込むことができます。

このチュートリアルでは、Android 用 MapLibre ネイティブ SDK を Amazon Location と統合する方法について説明します。このチュートリアルのサンプルアプリケーションは、 の Amazon Location Service サンプルリポジトリの一部として利用できますGitHub。

アプリケーションの構築：初期化

アプリケーションを初期化するには：

1. [空のアクティビティ] テンプレートから新しい Android Studio プロジェクトを作成します。
2. プロジェクト言語に [Kotlin] が選択されていることを確認します。
3. [Minimum SDK of API 14: Android 4.0 (Ice Cream Sandwich)] 以降を選択してください。
4. [プロジェクト構造]を開き、[ファイル] > [プロジェクト構造...] に移動します。[依存関係] セクションを選択します。
5. [<All Modules>] を選択したら、[+] ボタンを選択して[新しいライブラリ依存関係]を追加します。
6. [AWS Android SDK] バージョンは、2.20.0 以降を追加します。例：`com.amazonaws:aws-android-sdk-core:2.20.0`
7. MapLibre Android バージョン 9.4.0 以降のネイティブ SDK を追加します。
   例：`org.maplibre.gl:android-sdk:9.4.0`
8. build.gradle ファイルのプロジェクトレベルで、次の Maven リポジトリを追加して Android のパッケージにアクセスします MapLibre。

```
allprojects {
    repositories {
        // Retain your existing repositories
        google()
```

```
        jcenter()

        //  Declare the repositories for MapLibre
        mavenCentral()
    }
}
```

アプリケーションの構築：設定

アプリケーションにリソースと AWS リージョンをを設定するには：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="identityPoolId">us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd</
string>
    <string name="mapName">ExampleMap</string>
    <string name="awsRegion">us-east-1</string>
</resources>
```

アプリケーションの構築：アクティビティレイアウト

app/src/main/res/layout/activity_main.xml を編集します。

- マップをレンダリングする MapView を追加します。これにより、マップの初期中心点も設定されます。
- アトリビューションを表示する TextView を追加してください。

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.mapbox.mapboxsdk.maps.MapView
        android:id="@+id/mapView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:mapbox_cameraTargetLat="49.2819"
```

```
        app:mapbox_cameraTargetLng="-123.1187"
        app:mapbox_cameraZoom="12"
        app:mapbox_uiAttribution="false"
        app:mapbox_uiLogo="false" />

    <TextView
        android:id="@+id/attributionView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#80808080"
        android:padding="5sp"
        android:textColor="@android:color/black"
        android:textSize="10sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        tools:ignore="SmallSp" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

> **Note**
>
> 使用する各データプロバイダーのワードマークまたはテキストアトリビューションを、アプリケーションまたはドキュメントに記載する必要があります。アトリビューション文字列は、スタイル記述子のレスポンスのsources.esri.attribution、sources.here.attribution、source.grabmaptiles.attキーに含まれています。Amazon Location リソースをデータプロバイダーと併用する場合は、サービスの利用規約を必ずお読みください。

アプリケーションの構築：リクエスト変換

AWS リクエストをインターセプトし、署名バージョン 4 を使用して署名する SigV4Interceptor という名前のクラスを作成します。この情報は、メインアクティビティの作成時にマップリソースの取得に使用される HTTP クライアントに登録されます。

```
package aws.location.demo.okhttp

import com.amazonaws.DefaultRequest
import com.amazonaws.auth.AWS4Signer
import com.amazonaws.auth.AWSCredentialsProvider
import com.amazonaws.http.HttpMethodName
import com.amazonaws.util.IOUtils
```

```
import okhttp3.HttpUrl
import okhttp3.Interceptor
import okhttp3.Request
import okhttp3.Response
import okio.Buffer
import java.io.ByteArrayInputStream
import java.net.URI

class SigV4Interceptor(
    private val credentialsProvider: AWSCredentialsProvider,
    private val serviceName: String
) : Interceptor {
    override fun intercept(chain: Interceptor.Chain): Response {
        val originalRequest = chain.request()

        if (originalRequest.url().host().contains("amazonaws.com")) {
            val signer = if (originalRequest.url().encodedPath().contains("@")) {
                // the presence of "@" indicates that it doesn't need to be double URL-
encoded
                AWS4Signer(false)
            } else {
                AWS4Signer()
            }

            val awsRequest = toAWSRequest(originalRequest, serviceName)
            signer.setServiceName(serviceName)
            signer.sign(awsRequest, credentialsProvider.credentials)

            return chain.proceed(toSignedOkHttpRequest(awsRequest, originalRequest))
        }

        return chain.proceed(originalRequest)
    }

    companion object {
        fun toAWSRequest(request: Request, serviceName: String): DefaultRequest<Any> {
            // clone the request (AWS-style) so that it can be populated with
 credentials
            val dr = DefaultRequest<Any>(serviceName)

            // copy request info
            dr.httpMethod = HttpMethodName.valueOf(request.method())
            with(request.url()) {
                dr.resourcePath = uri().path
```

```
                dr.endpoint = URI.create("${scheme()}://${host()}")

                // copy parameters
                for (p in queryParameterNames()) {
                    if (p != "") {
                        dr.addParameter(p, queryParameter(p))
                    }
                }
            }

            // copy headers
            for (h in request.headers().names()) {
                dr.addHeader(h, request.header(h))
            }

            // copy the request body
            val bodyBytes = request.body()?.let { body ->
                val buffer = Buffer()
                body.writeTo(buffer)
                IOUtils.toByteArray(buffer.inputStream())
            }

            dr.content = ByteArrayInputStream(bodyBytes ?: ByteArray(0))

            return dr
        }

        fun toSignedOkHttpRequest(
            awsRequest: DefaultRequest<Any>,
            originalRequest: Request
        ): Request {
            // copy signed request back into an OkHttp Request
            val builder = Request.Builder()

            // copy headers from the signed request
            for ((k, v) in awsRequest.headers) {
                builder.addHeader(k, v)
            }

            // start building an HttpUrl
            val urlBuilder = HttpUrl.Builder()
                .host(awsRequest.endpoint.host)
                .scheme(awsRequest.endpoint.scheme)
                .encodedPath(awsRequest.resourcePath)
```

```
            // copy parameters from the signed request
            for ((k, v) in awsRequest.parameters) {
                urlBuilder.addQueryParameter(k, v)
            }

            return builder.url(urlBuilder.build())
                .method(originalRequest.method(), originalRequest.body())
                .build()
        }
    }
}
```

アプリケーションの構築：メインアクティビティ

メインアクティビティは、ユーザーに表示されるビューを初期化します。これには以下が含まれます。

- Amazon Cognito `CredentialsProvider` をインスタンス化します。
- 署名バージョン 4をインターセプターとして登録します。
- マップスタイル記述子を指定してマップを設定し、適切なアトリビューションを表示します。

また、`MainActivity` には、ライフサイクルイベントをマップビューに転送して、呼び出してもアクティブなビューポートを維持できるようにする役割もあります。

```
package aws.location.demo.maplibre

import android.os.Bundle
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import aws.location.demo.okhttp.SigV4Interceptor
import com.amazonaws.auth.CognitoCachingCredentialsProvider
import com.amazonaws.regions.Regions
import com.mapbox.mapboxsdk.Mapbox
import com.mapbox.mapboxsdk.maps.MapView
import com.mapbox.mapboxsdk.maps.Style
import com.mapbox.mapboxsdk.module.http.HttpRequestUtil
import okhttp3.OkHttpClient

private const val SERVICE_NAME = "geo"
```

```
class MainActivity : AppCompatActivity() {
    private var mapView: MapView? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // configuration
        val identityPoolId = getString(R.string.identityPoolId)
        val region = getString(R.string.awsRegion)
        val mapName = getString(R.string.mapName)

        // Credential initialization
        val credentialProvider = CognitoCachingCredentialsProvider(
            applicationContext,
            identityPoolId,
            Regions.fromName(identityPoolId.split(":").first())
        )

        // initialize MapLibre
        Mapbox.getInstance(this, null)
        HttpRequestUtil.setOkHttpClient(
            OkHttpClient.Builder()
                .addInterceptor(SigV4Interceptor(credentialProvider, SERVICE_NAME))
                .build()
        )

        // initialize the view
        setContentView(R.layout.activity_main)

        // initialize the map view
        mapView = findViewById(R.id.mapView)
        mapView?.onCreate(savedInstanceState)
        mapView?.getMapAsync { map ->
            map.setStyle(
                Style.Builder()
                    .fromUri("https://maps.geo.${region}.amazonaws.com/maps/v0/maps/
${mapName}/style-descriptor")
            ) { style ->
                findViewById<TextView>(R.id.attributionView).text =
 style.sources.first()?.attribution
            }
        }
    }
```

```
    override fun onStart() {
        super.onStart()
        mapView?.onStart()
    }

    override fun onResume() {
        super.onResume()
        mapView?.onResume()
    }

    override fun onPause() {
        super.onPause()
        mapView?.onPause()
    }

    override fun onStop() {
        super.onStop()
        mapView?.onStop()
    }

    override fun onSaveInstanceState(outState: Bundle) {
        super.onSaveInstanceState(outState)
        mapView?.onSaveInstanceState(outState)
    }

    override fun onLowMemory() {
        super.onLowMemory()
        mapView?.onLowMemory()
    }

    override fun onDestroy() {
        super.onDestroy()
        mapView?.onDestroy()
    }
}
```

このアプリケーションを実行すると、選択したスタイルで全画面マップが表示されます。このサンプルは、 の Amazon Location Service サンプルリポジトリの一部として入手できますGitHub。

Amazon Location Service での iOS 用 MapLibre ネイティブ SDK の使用

MapLibre ネイティブ SDK for iOS を使用して、クライアント側マップを iOS アプリケーションに埋め込みます。

iOS 用 MapLibre ネイティブ SDK は Mapbox GL Native に基づくライブラリであり、Amazon Location Service Maps API が提供するスタイルとタイルと互換性があります。 MapLibre Native SDK for iOS を統合して、スケーラブルでカスタマイズ可能なベクトルマップを含むインタラクティブマップビューを iOS アプリケーションに埋め込むことができます。

このチュートリアルでは、 MapLibre ネイティブ SDK for iOS を Amazon Location と統合する方法について説明します。このチュートリアルのサンプルアプリケーションは、 の Amazon Location Service サンプルリポジトリの一部として利用できますGitHub。

アプリケーションの構築：初期化

アプリケーションを初期化するには：

1. [App] テンプレートから新しい Xcode プロジェクトを作成します。
2. インターフェースには [SwiftUI] を選択します。
3. ライフサイクルには [SwiftUI] アプリケーションを選択します。
4. 言語には [Swift] を選択します。

Swift パッケージを使用した依存関係の追加 MapLibre

Xcode プロジェクトにパッケージ依存関係を追加するには：

1. [ファイル] > [Swift パッケージ] > [パッケージ依存関係の追加] に移動します。
2. リポジトリ URL に **`https://github.com/maplibre/maplibre-gl-native-distribution`** を入力します。

   > **Note**
   >
   > Swift パッケージの詳細については、Apple.com の「アプリへのパッケージ依存関係の追加」を参照してください。

3. ターミナルに をインストールします CocoaPods。

   ```
   sudo gem install cocoapods
   ```

4. アプリケーションのプロジェクトディレクトリに移動し、 CocoaPods パッケージマネージャーを使用して Podfile を初期化します。

   ```
   pod init
   ```

5. Podfile を開いて、AWSCore を依存関係として追加します。

```
platform :ios, '12.0'

target 'Amazon Location Service Demo' do
  use_frameworks!

  pod 'AWSCore'
end
```

6. 依存関係をダウンロードしインストールします。

```
pod install --repo-update
```

7. が CocoaPods 作成した Xcode ワークスペースを開きます。

```
xed .
```

アプリケーションの構築：設定

Info.plist に以下のキーと値を追加してアプリケーションを設定します。

キー	値
AWSRegion	us-east-1
IdentityPoolId	us-east-1:54 f2ba88-9390-498d-aaa5-0d97f b7ca3bd
MapName	ExampleMap

アプリケーションの構築： ContentView レイアウト

マップをレンダリングするには、ContentView.swift を編集します。

- マップをレンダリングする MapView を追加します。
- アトリビューションを表示する TextField を追加します。

これにより、マップの初期中心点も設定されます。

```
import SwiftUI

struct ContentView: View {
    @State private var attribution = ""

    var body: some View {
        MapView(attribution: $attribution)
            .centerCoordinate(.init(latitude: 49.2819, longitude: -123.1187))
            .zoomLevel(12)
            .edgesIgnoringSafeArea(.all)
            .overlay(
                TextField("", text: $attribution)
                    .disabled(true)
                    .font(.system(size: 12, weight: .light, design: .default))
                    .foregroundColor(.black)
                    .background(Color.init(Color.RGBColorSpace.sRGB, white: 0.5,
 opacity: 0.5))
                    .cornerRadius(1),
                alignment: .bottomTrailing)
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

Note

使用する各データプロバイダーのワードマークまたはテキストアトリビューションを、アプリケーションまたはドキュメントに記載する必要があります。アトリビューション文字列は、スタイル記述子のレスポンスのsources.esri.attribution、sources.here.attribution、source.grabmaptiles.attキーに含まれています。Amazon Location リソースをデータプロバイダーと併用する場合は、サービスの利用規約を必ずお読みください。

アプリケーションの構築：リクエスト変換

AWSSignatureV4Delegate.swift という名前の Swift ファイルを作成し、AWS リクエストをインターセプトし、署名バージョン 4 を使用して署名するクラスを定義します。マップビューでは、このクラスのインスタンスが URL の書き換えも担当するオフラインストレージデリゲートとして割り当てられます。

```
import AWSCore
import Mapbox

class AWSSignatureV4Delegate : NSObject, MGLOfflineStorageDelegate {
    private let region: AWSRegionType
    private let identityPoolId: String
    private let credentialsProvider: AWSCredentialsProvider

    init(region: AWSRegionType, identityPoolId: String) {
        self.region = region
        self.identityPoolId = identityPoolId
        self.credentialsProvider = AWSCognitoCredentialsProvider(regionType: region,
 identityPoolId: identityPoolId)
        super.init()
    }

    class func doubleEncode(path: String) -> String? {
        return path.addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)?
            .addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)
    }

    func offlineStorage(_ storage: MGLOfflineStorage, urlForResourceOf kind:
 MGLResourceKind, with url: URL) -> URL {
        if url.host?.contains("amazonaws.com") != true {
            // not an AWS URL
            return url
        }

        // URL-encode spaces, etc.
        let keyPath = String(url.path.dropFirst())
        guard let percentEncodedKeyPath =
 keyPath.addingPercentEncoding(withAllowedCharacters: .urlPathAllowed) else {
            print("Invalid characters in path '\(keyPath)'; unsafe to sign")
            return url
        }
```

```
        let endpoint = AWSEndpoint(region: region, serviceName: "geo", url: url)
        let requestHeaders: [String: String] = ["host": endpoint!.hostName]

        // sign the URL
        let task = AWSSignatureV4Signer
            .generateQueryStringForSignatureV4(
                withCredentialProvider: credentialsProvider,
                httpMethod: .GET,
                expireDuration: 60,
                endpoint: endpoint!,
                // workaround for https://github.com/aws-amplify/aws-sdk-ios/
issues/3215
                keyPath: AWSSignatureV4Delegate.doubleEncode(path:
 percentEncodedKeyPath),
                requestHeaders: requestHeaders,
                requestParameters: .none,
                signBody: true)
        task.waitUntilFinished()

        if let error = task.error as NSError? {
            print("Error occurred: \(error)")
        }

        if let result = task.result {
            var urlComponents = URLComponents(url: (result as URL),
 resolvingAgainstBaseURL: false)!
            // re-use the original path; workaround for https://github.com/aws-amplify/
aws-sdk-ios/issues/3215
            urlComponents.path = url.path

            // have Mapbox GL fetch the signed URL
            return (urlComponents.url)!
        }

        // fall back to an unsigned URL
        return url
    }
}
```

アプリケーションの構築：マップビュー

マップビューは、AWSSignatureV4Delegate インスタンスを初期化し、リソースを取得してマップをレンダリングする基盤となる MGLMapView の構成を行います。また、スタイル記述子のソースからにアトリビューション文字列を ContentView に伝達する処理も行います。

以下の struct の定義を含む、MapView.swift という名前の新しい Swift ファイルを作成します。

```
import SwiftUI
import AWSCore
import Mapbox

struct MapView: UIViewRepresentable {
    @Binding var attribution: String

    private var mapView: MGLMapView
    private var signingDelegate: MGLOfflineStorageDelegate

    init(attribution: Binding<String>) {
        let regionName = Bundle.main.object(forInfoDictionaryKey: "AWSRegion") as!
 String
        let identityPoolId = Bundle.main.object(forInfoDictionaryKey: "IdentityPoolId")
 as! String
        let mapName = Bundle.main.object(forInfoDictionaryKey: "MapName") as! String

        let region = (regionName as NSString).aws_regionTypeValue()

        // MGLOfflineStorage doesn't take ownership, so this needs to be a member here
        signingDelegate = AWSSignatureV4Delegate(region: region, identityPoolId:
 identityPoolId)

        // register a delegate that will handle SigV4 signing
        MGLOfflineStorage.shared.delegate = signingDelegate

        mapView = MGLMapView(
            frame: .zero,
            styleURL: URL(string: "https://maps.geo.\(regionName).amazonaws.com/maps/
v0/maps/\(mapName)/style-descriptor"))

        _attribution = attribution
    }

    func makeCoordinator() -> Coordinator {
```

```
        Coordinator($attribution)
    }

    class Coordinator: NSObject, MGLMapViewDelegate {
        var attribution: Binding<String>

        init(_ attribution: Binding<String>) {
            self.attribution = attribution
        }

        func mapView(_ mapView: MGLMapView, didFinishLoading style: MGLStyle) {
            let source = style.sources.first as? MGLVectorTileSource
            let attribution = source?.attributionInfos.first
            self.attribution.wrappedValue = attribution?.title.string ?? ""
        }
    }

    // MARK: - UIViewRepresentable protocol

    func makeUIView(context: UIViewRepresentableContext<MapView>) -> MGLMapView {
        mapView.delegate = context.coordinator

        mapView.logoView.isHidden = true
        mapView.attributionButton.isHidden = true
        return mapView
    }

    func updateUIView(_ uiView: MGLMapView, context:
 UIViewRepresentableContext<MapView>) {
    }

    // MARK: - MGLMapView proxy

    func centerCoordinate(_ centerCoordinate: CLLocationCoordinate2D) -> MapView {
        mapView.centerCoordinate = centerCoordinate
        return self
    }

    func zoomLevel(_ zoomLevel: Double) -> MapView {
        mapView.zoomLevel = zoomLevel
        return self
    }
}
```

このアプリケーションを実行すると、選択したスタイルで全画面マップが表示されます。このサンプルは、 の Amazon Location Service サンプルリポジトリの一部として入手できますGitHub。

Amazon Location Service で Amplify ライブラリを使用する

次のチュートリアルは、Amazon Location で AWS Amplify を使用する方法を説明します。Amplify は MapLibre GL JS を使用して、 JavaScriptベースのアプリケーションでマップをレンダリングします。

Amplify は、Amazon Location Service を利用する Amplify Geo など、さまざまなカテゴリのサービスへのインターフェイスを提供する一連のオープンソースのクライアントライブラリです。 AWS Amplify Geo JavaScript ライブラリ の詳細をご覧ください。

> **Note**
>
> このチュートリアルは、「マップを使用する - アプリケーションにマップを追加するには」の手順をすでに実行していることを前提としています。

アプリケーションの構築： Scaffolding

このチュートリアルでは、 を使用して JavaScript HTML ページにマップを構築するウェブアプリケーションを作成します。

まず、マップのコンテナを含む HTML ページ (`index.html`) を作成します。

- `map`の `id` が付いた `div` 要素を入力し、マップの寸法をマップビューに適用します。寸法はビューポートから継承されます。

```
<html>
  <head>
    <style>
      body { margin: 0; }
      #map { height: 100vh; } /* 100% of viewport height */
    </style>
  </head>

  <body>
    <!-- map container -->
    <div id="map" />
```

```
  </body>
</html>
```

アプリケーションの構築：依存関係の追加

次の依存関係をアプリケーションに追加します。

- AWS Amplify マップライブラリとジオライブラリ。
- AWS Amplify コアライブラリ。
- AWS Amplify 認証ライブラリ。
- AWS Amplify スタイルシート。

```
<!-- CSS dependencies -->
    <link href="https://cdn.amplify.aws/packages/maplibre-
gl/1.15.2/maplibre-gl.css" rel="stylesheet" integrity="sha384-
DrPVD9GufrxGb7kWwRv0CywpXTmfvbKOZ5i5pN7urmIThew0zXKTME+gutUgtpeD"
 crossorigin="anonymous" referrerpolicy="no-referrer"></link>

<!-- JavaScript dependencies -->
    <script src="https://cdn.amplify.aws/packages/maplibre-gl/1.15.2/maplibre-gl.js"
 integrity="sha384-rwYfkmAOpciZS2bDuwZ/Xa/Gog6jXem8D/whm3wnsZSVFemDDlprcUXHnDDUcrNU"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/core/4.3.0/aws-amplify-core.min.js"
 integrity="sha384-7Oh+5w0l7XGyYvSqbKi2Q7SA5K640V5nyW2/LEbevDQEV1HMJqJLA1A00z2hu8fJ"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/auth/4.3.8/aws-amplify-auth.min.js"
 integrity="sha384-jfkXCEfYyVmDXYKlgWNwv54xRaZgk14m7sjeb2jLVBtUXCD2p+WU8YZ2mPZ9Xbdw"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/geo/1.1.0/aws-amplify-geo.min.js"
 integrity="sha384-TFMTyWuCbiptXTzvOgzJbV8TPUupG1rA1AVrznAhCSpXTIdGw82bGd8RTk5rr3nP"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/maplibre-gl-js-
amplify/1.1.0/maplibre-gl-js-amplify.umd.min.js" integrity="sha384-7/
RxWonKW1nM9zCKiwU9x6bkQTjldosg0D1vZYm0Zj+K/vUSnA3sOMhlRRWAtHPi" crossorigin="anonymous"
 referrerpolicy="no-referrer"></script>
<script>
  // application-specific code
</script>
```

これで、マップのコンテナを含む空のページが作成されます。

アプリケーションの構築：設定

を使用してアプリケーションを設定するには JavaScript：

1. 「マップを使用する - ステップ 2、認証を設定する」で作成した認証されていないアイデンティティプールの識別子を入力します。

```
// Cognito Identity Pool ID
const identityPoolId = "region:identityPoolID"; // for example: us-
east-1:123example-1234-5678
// extract the Region from the Identity Pool ID
const region = identityPoolId.split(":")[0];
```

2. アイデンティティプールやマップリソース (ここではデフォルトの `explore.map` という名前で表示) など、作成したリソースを使用するように AWS Amplify を設定します。

```
// Configure Amplify
const { Amplify } = aws_amplify_core;
const { createMap } = AmplifyMapLibre;

Amplify.configure({
  Auth: {
    identityPoolId,
    region,
  },
  geo: {
    AmazonLocationService: {
      maps: {
        items: {
          "explore.map": {
            style: "Default style"
          },
        },
        default: "explore.map",
      },
      region,
    },
  }
});
```

アプリケーションの構築：マップの初期化

ページが読み込まれた後にマップを表示するには、マップを初期化する必要があります。マップの初期位置を調整したり、コントロールを追加したり、データをオーバーレイしたりできます。

```
async function initializeMap() {
  const map = await createMap(
    {
      container: "map",
      center: [-123.1187, 49.2819],
      zoom: 10,
      hash: true,
    }
  );

  map.addControl(new maplibregl.NavigationControl(), "top-left");
}

initializeMap();
```

> **Note**
>
> 使用する各データプロバイダーのワードマークまたはテキストアトリビューションを、アプリケーションまたはドキュメントに記載する必要があります。アトリビューション文字列は、スタイル記述子のレスポンスの`sources.esri.attribution`、`sources.here.attribution`、`sources.grabmaptiles.at`キーに含まれています。Amplify は自動的にアトリビューションを提供します。Amazon Location リソースをデータプロバイダーと併用する場合は、サービスの利用規約を必ずお読みください。

Java アプリケーションを実行する

このサンプルアプリケーションは、ローカル Web サーバーで使用するか、ブラウザで開いて実行できます。

ローカル Web サーバーを使用するには、Node.js の一部としてインストールされた npx を使用するか、任意の他の Web サーバーを使用できます。npx を使用するには、`index.html` と同じディレクトリ内から `npx serve` を入力してください。これによってアプリケーションに `localhost:5000` が提供されます。

> **Note**
>
> 認証されていない Amazon Cognito ロール用に作成したポリシーに `referer` 条件が含まれている場合、`localhost:` のURL によるテストがブロックされる可能性があります。この場合、ポリシーに含まれる URL を提供する Web サーバーでテストできます。

チュートリアルを完了すると、最終的なアプリケーションは次の例のようになります。

```
<html>
  <head>
    <!-- CSS dependencies -->
    <link href="https://cdn.amplify.aws/packages/maplibre-
gl/1.15.2/maplibre-gl.css" rel="stylesheet" integrity="sha384-
DrPVD9GufrxGb7kWwRv0CywpXTmfvbKOZ5i5pN7urmIThew0zXKTME+gutUgtpeD"
 crossorigin="anonymous" referrerpolicy="no-referrer"></link>

    <!-- JavaScript dependencies -->
    <script src="https://cdn.amplify.aws/packages/maplibre-gl/1.15.2/maplibre-gl.js"
 integrity="sha384-rwYfkmAOpciZS2bDuwZ/Xa/Gog6jXem8D/whm3wnsZSVFemDDlprcUXHnDDUcrNU"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/core/4.3.0/aws-amplify-core.min.js"
 integrity="sha384-7Oh+5w0l7XGyYvSqbKi2Q7SA5K640V5nyW2/LEbevDQEV1HMJqJLA1A00z2hu8fJ"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/auth/4.3.8/aws-amplify-auth.min.js"
 integrity="sha384-jfkXCEfYyVmDXYKlgWNwv54xRaZgk14m7sjeb2jLVBtUXCD2p+WU8YZ2mPZ9Xbdw"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/geo/1.1.0/aws-amplify-geo.min.js"
 integrity="sha384-TFMTyWuCbiptXTzvOgzJbV8TPUupG1rA1AVrznAhCSpXTIdGw82bGd8RTk5rr3nP"
 crossorigin="anonymous" referrerpolicy="no-referrer"></script>
    <script src="https://cdn.amplify.aws/packages/maplibre-gl-js-
amplify/1.1.0/maplibre-gl-js-amplify.umd.min.js" integrity="sha384-7/
RxWonKW1nM9zCKiwU9x6bkQTjldosg0D1vZYm0Zj+K/vUSnA3sOMhlRRWAtHPi" crossorigin="anonymous"
 referrerpolicy="no-referrer"></script>

    <style>
      body { margin: 0; }
      #map { height: 100vh; }
    </style>
  </head>

  <body>
```

```
    <div id="map" />
    <script type="module">
      // Cognito Identity Pool ID
      const identityPoolId = "region:identityPoolId"; // for example: us-
east-1:123example-1234-5678
      // extract the Region from the Identity Pool ID
      const region = identityPoolId.split(":")[0];

      // Configure Amplify
      const { Amplify } = aws_amplify_core;
      const { createMap } = AmplifyMapLibre;

      Amplify.configure({
        Auth: {
          identityPoolId,
          region,
        },
        geo: {
          AmazonLocationService: {
            maps: {
              items: {
                "explore.map": {
                  style: "Default style"
                },
              },
              default: "explore.map",
            },
            region,
          },
        }
      });

      async function initializeMap() {
        const map = await createMap(
          {
            container: "map",
            center: [-123.1187, 49.2819],
            zoom: 10,
            hash: true,
          }
        );

        map.addControl(new maplibregl.NavigationControl(), "top-left");
      }
```

```
      initializeMap();
    </script>
  </body>
</html>
```

このアプリケーションを実行すると、選択したマップスタイルを使用して全画面マップが表示されます。このサンプルは、Amazon Location Service コンソールの任意のマップリソースページの [マップを埋め込む] タブにも記載されています。

このチュートリアルを完了したら、「AWS Amplify」ドキュメントの「マップを表示する」トピックに移動して、マップにマーカーを表示する方法などの詳細を確認してください。

Amazon Location Service で Tagram を使用する

このセクションでは、以下のチュートリアルで、Tangram を Amazon Location と統合する方法について説明します。

> **Important**
>
> 以下のチュートリアルの Tangram スタイルは、`VectorHereContrast` のスタイルで設定された Amazon Location マップリソースとのみ互換性があります。

以下は、 *`VectorHereContrast`*スタイル*`TangramExampleMap`*を使用して AWS という新しいマップリソースを作成する CLI コマンドの例です。

```
aws --region us-east-1 \
  location \
  create-map \
  --map-name "TangramExampleMap" \
  --configuration "Style=VectorHereContrast"
```

> **Note**
>
> 請求は使用量によって決まります。他の AWS サービスの利用料金が請求される場合があります。詳細については、「Amazon Location Service 料金」を参照してください。

トピック

- Amazon Location Service で Tagram を使用する
- Amazon Location Service で Android 用 Tangram ES を使用する
- Amazon Location Service で iOS 用 Tangram ES を使用する

Amazon Location Service で Tagram を使用する

Tangram は柔軟性の高いマッピングエンジンで、ベクタータイルから 2D および 3D マップをリアルタイムでレンダリングできるように設計されています。MapZen が設計したスタイルと Amazon Location Service マップ API が提供する HERE タイルで使用できます。このガイドでは、基本的な HTML/JavaScript アプリケーション内の Amazon Location とTAKgram を統合する方法について説明しますが、React や Angular などのフレームワークを使用する場合にも同じライブラリとテクニックが適用されます。

TAKgram は、モバイルに対応したインタラクティブマップ用のオープンソース JavaScript ライブラリである Leaflet 上に構築されています。つまり、Leaflet と互換性のあるのプラグインやコントロールの多くは Tangram でも動作するということです。

Tilezen スキーマで動作するように構築されたTangram スタイルは、HERE のマップを使用する場合には Amazon Location とほぼ互換性があります。具体的には次のとおりです。

- Bubble Wrap — 興味のある地点をわかりやすく表示する便利なアイコンが付いた、フル機能の道案内スタイルです。
- Cinnabar — クラシックな外観で、一般的なマップアプリに適しています。
- Refill — Stamen Designの独創的な Toner スタイルにインスパイアされた、データ視覚化オーバーレイ用にデザインされたミニマルなマップスタイルです。
- Tron — TRONのビジュアル言語によるスケール返還を探求したスタイルです。
- Walkabout — ハイキングや外出に最適なアウトドア重視のスタイルです。

このガイドでは、Bubble Wrap と呼ばれるTAKgram スタイルを使用して、基本的な HTML/JavaScript アプリケーション内でTAKgram を Amazon Location と統合する方法について説明します。このサンプルは、 の Amazon Location Service サンプルリポジトリの一部として入手できますGitHub。

他の Tangram スタイルには地形情報をエンコードするラスタータイルが最適ですが、この機能はまだ Amazon Location ではサポートされていません。

> **Important**
>
> 以下のチュートリアルの Tangram スタイルは、`VectorHereContrast` スタイルで設定された Amazon Location マップリソースとのみ互換性があります。

アプリケーションの構築： Scaffolding

アプリケーションは、ウェブアプリケーション上にマップを構築 JavaScript するための を含む HTML ページです。マップのコンテナを含む HTML ページ (`index.html`) を作成します。

- マップの `id` が付いた `div` 要素を入力し、マップの寸法をマップビューに適用します。
- 寸法はビューポートから継承されます。

```
<html>
  <head>
    <style>
      body {
        margin: 0;
      }

      #map {
        height: 100vh; /* 100% of viewport height */
      }
    </style>
  </head>
  <body>
    <!-- map container -->
    <div id="map" />
  </body>
</html>
```

アプリケーションの構築：依存関係の追加

以下の依存関係を追加します。

- Leaflet とそれに関連する CSS。
- Tangram。
- AWS SDK for JavaScript。

```
<!-- CSS dependencies -->
<link
  rel="stylesheet"
  href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
  integrity="sha512-xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/
sMZMZ19scR4PsZChSR7A=="
  crossorigin=""
/>
<!-- JavaScript dependencies -->
<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
<script src="https://unpkg.com/tangram"></script>
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.784.0.min.js"></script>
<script>
  // application-specific code
</script>
```

これにより、必要な前提条件を含む空のページが作成されます。次のステップでは、アプリケーションの JavaScript コードを記述する手順を説明します。

アプリケーションの構築：設定

リソースと認証情報を使用してアプリケーションを設定します。

1. リソースの名前と ID を入力します。

   ```
   // Cognito Identity Pool ID
   const identityPoolId = "us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd";
   // Amazon Location Service map name; must be HERE-backed
   const mapName = "TangramExampleMap";
   ```

2. 「マップを使用する - ステップ 2. 認証を設定する」で作成した認証されていない アイデンティティプールを使用して、認証情報プロバイダーをインスタンス化します。通常の AWS SDK ワークフローではない認証情報を使用するため、セッションの有効期限は 1時間となっています。

   ```
   // extract the region from the Identity Pool ID; this will be used for both Amazon
    Cognito and Amazon Location
   AWS.config.region = identityPoolId.split(":", 1)[0];

   // instantiate a Cognito-backed credential provider
   const credentials = new AWS.CognitoIdentityCredentials({
     IdentityPoolId: identityPoolId,
   ```

```
});
```

3. Tangram ではタイルの取得に使用する URL を上書きできますが、リクエストをインターセプトして署名する機能は含まれていません。

   その解決策として、合成ホスト名 `amazon.location` を使用して Amazon Location を指すように `sources.mapzen.url` をオーバーライドします。このホスト名はサービスワーカーが処理します。以下は、Bubble Wrap を使用したシーン設定の例です。

```
const scene = {
  import: [
    // Bubble Wrap style
    "https://www.nextzen.org/carto/bubble-wrap-style/10/bubble-wrap-style.zip",
    "https://www.nextzen.org/carto/bubble-wrap-style/10/themes/label-7.zip",
    "https://www.nextzen.org/carto/bubble-wrap-style/10/themes/bubble-wrap-road-
shields-usa.zip",
    "https://www.nextzen.org/carto/bubble-wrap-style/10/themes/bubble-wrap-road-
shields-international.zip",
  ],
  // override values beneath the `sources` key in the style above
  sources: {
    mapzen: {
      // point at Amazon Location using a synthetic URL, which will be handled by
 the service
      // worker
      url: `https://amazon.location/${mapName}/{z}/{x}/{y}`,
    },
    // effectively disable raster tiles containing encoded normals
    normals: {
      max_zoom: 0,
    },
    "normals-elevation": {
      max_zoom: 0,
    },
  },
};
```

アプリケーションの構築：リクエスト変換

サービスワーカーを登録して初期化するには、マップが初期化される前に呼び出される `registerServiceWorker` 関数を作成します。これにより、 を制御するサービスワー

カーsw.jsとして と呼ばれる別のファイルに提供された JavaScript コードが登録されますindex.html。

認証情報は Amazon Cognito から読み込まれ、リージョンと一緒にサービスワーカーに渡されます。この情報は、署名バージョン 4 でタイルリクエストに署名するために使われます。

```
/**
 * Register a service worker that will rewrite and sign requests using Signature
 Version 4.
 */
async function registerServiceWorker() {
  if ("serviceWorker" in navigator) {
    try {
      const reg = await navigator.serviceWorker.register("./sw.js");

      // refresh credentials from Amazon Cognito
      await credentials.refreshPromise();

      await reg.active.ready;

      if (navigator.serviceWorker.controller == null) {
        // trigger a navigate event to active the controller for this page
        window.location.reload();
      }

      // pass credentials to the service worker
      reg.active.postMessage({
        credentials: {
          accessKeyId: credentials.accessKeyId,
          secretAccessKey: credentials.secretAccessKey,
          sessionToken: credentials.sessionToken,
        },
        region: AWS.config.region,
      });
    } catch (error) {
      console.error("Service worker registration failed:", error);
    }
  } else {
    console.warn("Service worker support is required for this example");
  }
}
```

sw.js の Service Worker は、認証情報やリージョンの設定変更を拾うために message のイベントを待ち受けるよう実装されています。また、fetch のイベントを監視することでプロキシサーバーとしても機能します。amazon.location 合成ホスト名をターゲットとする fetch イベントは、適切な Amazon Location API を対象とするように書き替えられ、Amplify Core のSigner を使用して署名されます。

```
// sw.js
self.importScripts(
  "https://unpkg.com/@aws-amplify/core@3.7.0/dist/aws-amplify-core.min.js"
);

const { Signer } = aws_amplify_core;

let credentials;
let region;

self.addEventListener("install", (event) => {
  // install immediately
  event.waitUntil(self.skipWaiting());
});

self.addEventListener("activate", (event) => {
  // control clients ASAP
  event.waitUntil(self.clients.claim());
});

self.addEventListener("message", (event) => {
  const {
    data: { credentials: newCredentials, region: newRegion },
  } = event;

  if (newCredentials != null) {
    credentials = newCredentials;
  }

  if (newRegion != null) {
    region = newRegion;
  }
});

async function signedFetch(request) {
  const url = new URL(request.url);
  const path = url.pathname.slice(1).split("/");
```

```
  // update URL to point to Amazon Location
  url.pathname = `/maps/v0/maps/${path[0]}/tiles/${path.slice(1).join("/")}`;
  url.host = `maps.geo.${region}.amazonaws.com`;
  // strip params (Tangram generates an empty api_key param)
  url.search = "";

  const signed = Signer.signUrl(url.toString(), {
    access_key: credentials.accessKeyId,
    secret_key: credentials.secretAccessKey,
    session_token: credentials.sessionToken,
  });

  return fetch(signed);
}

self.addEventListener("fetch", (event) => {
  const { request } = event;

  // match the synthetic hostname we're telling Tangram to use
  if (request.url.includes("amazon.location")) {
    return event.respondWith(signedFetch(request));
  }

  // fetch normally
  return event.respondWith(fetch(request));
});
```

認証情報を自動的に更新し、有効期限が切れる前にサービスワーカーに送信するには、index.html で以下の関数を使用します。

```
async function refreshCredentials() {
  await credentials.refreshPromise();

  if ("serviceWorker" in navigator) {
    const controller = navigator.serviceWorker.controller;

    controller.postMessage({
      credentials: {
        accessKeyId: credentials.accessKeyId,
        secretAccessKey: credentials.secretAccessKey,
        sessionToken: credentials.sessionToken,
      },
```

```
    });
  } else {
    console.warn("Service worker support is required for this example.");
  }

  // schedule the next credential refresh when they're about to expire
  setTimeout(refreshCredentials, credentials.expireTime - new Date());
}
```

アプリケーションの構築：マップの初期化

ページが読み込まれた後にマップを表示するには、マップを初期化する必要があります。マップの初期位置を調整したり、コントロールを追加したり、データをオーバーレイしたりできます。

> **Note**
>
> 使用する各データプロバイダーのワードマークまたはテキストアトリビューションを、アプリケーションまたはドキュメントに記載する必要があります。アトリビューション文字列は、スタイル記述子のレスポンスのsources.esri.attribution、sources.here.attribution、source.grabmaptiles.attキーに含まれています。
>
> Tangram はこれらのリソースをリクエストせず、HERE のマップとのみ互換性があるため、「© 2020 HERE」を使用してください。Amazon Location リソースをデータプロバイダーと併用する場合は、サービスの利用規約を必ずお読みください。

```
/**
 * Initialize a map.
 */
async function initializeMap() {
  // register the service worker to handle requests to https://amazon.location
  await registerServiceWorker();

  // Initialize the map
  const map = L.map("map").setView([49.2819, -123.1187], 10);
  Tangram.leafletLayer({
    scene,
  }).addTo(map);
  map.attributionControl.setPrefix("");
  map.attributionControl.addAttribution("© 2020 HERE");
}
```

```
initializeMap();
```

Java アプリケーションを実行する

このサンプルを実行するには、次の方法があります。

- HTTPS をサポートするホストを使用する。
- ローカル Web サーバーを使用して、サービスワーカーのセキュリティ制限に準拠する。

ローカル Web サーバーで使用するには、npx を使用できます。Node.js の一部としてインストールされるためです。`npx serve` は、`index.html` および `sw.js` と同じディレクトリ内から使用できます。これによってアプリケーションに localhost:5000 が提供されます。

以下は、`index.html` ファイルです。

```
<!-- index.html -->
<html>
  <head>
    <link
      rel="stylesheet"
      href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
      integrity="sha512-xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/
keqq/sMZMZ19scR4PsZChSR7A=="
      crossorigin=""
    />
    <style>
      body {
        margin: 0;
      }

      #map {
        height: 100vh;
      }
    </style>
  </head>

  <body>
    <div id="map" />
    <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
    <script src="https://unpkg.com/tangram"></script>
    <script src="https://sdk.amazonaws.com/js/aws-sdk-2.784.0.min.js"></script>
```

```
    <script>
      // configuration
      // Cognito Identity Pool ID
      const identityPoolId = "<Identity Pool ID>";
      // Amazon Location Service Map name; must be HERE-backed
      const mapName = "<Map name>";

      AWS.config.region = identityPoolId.split(":")[0];

      // instantiate a credential provider
      credentials = new AWS.CognitoIdentityCredentials({
        IdentityPoolId: identityPoolId,
      });

      const scene = {
        import: [
          // Bubble Wrap style
          "https://www.nextzen.org/carto/bubble-wrap-style/10/bubble-wrap-style.zip",
          "https://www.nextzen.org/carto/bubble-wrap-style/10/themes/label-7.zip",
          "https://www.nextzen.org/carto/bubble-wrap-style/10/themes/bubble-wrap-road-
shields-usa.zip",
          "https://www.nextzen.org/carto/bubble-wrap-style/10/themes/bubble-wrap-road-
shields-international.zip",
        ],
        // override values beneath the `sources` key in the style above
        sources: {
          mapzen: {
            // point at Amazon Location using a synthetic URL, which will be handled by
 the service
            // worker
            url: `https://amazon.location/${mapName}/{z}/{x}/{y}`,
          },
          // effectively disable raster tiles containing encoded normals
          normals: {
            max_zoom: 0,
          },
          "normals-elevation": {
            max_zoom: 0,
          },
        },
      };

      /**
```

```
       * Register a service worker that will rewrite and sign requests using Signature
Version 4.
       */
      async function registerServiceWorker() {
        if ("serviceWorker" in navigator) {
          try {
            const reg = await navigator.serviceWorker.register("./sw.js");

            // refresh credentials from Amazon Cognito
            await credentials.refreshPromise();

            await reg.active.ready;

            if (navigator.serviceWorker.controller == null) {
              // trigger a navigate event to active the controller for this page
              window.location.reload();
            }

            // pass credentials to the service worker
            reg.active.postMessage({
              credentials: {
                accessKeyId: credentials.accessKeyId,
                secretAccessKey: credentials.secretAccessKey,
                sessionToken: credentials.sessionToken,
              },
              region: AWS.config.region,
            });
          } catch (error) {
            console.error("Service worker registration failed:", error);
          }
        } else {
          console.warn("Service Worker support is required for this example");
        }
      }

      /**
       * Initialize a map.
       */
      async function initializeMap() {
        // register the service worker to handle requests to https://amazon.location
        await registerServiceWorker();

        // Initialize the map
        const map = L.map("map").setView([49.2819, -123.1187], 10);
```

```
        Tangram.leafletLayer({
          scene,
        }).addTo(map);
        map.attributionControl.setPrefix("");
        map.attributionControl.addAttribution("© 2020 HERE");
      }

      initializeMap();
    </script>
  </body>
</html>
```

以下は、sw.js ファイルです。

```
// sw.js
self.importScripts(
  "https://unpkg.com/@aws-amplify/core@3.7.0/dist/aws-amplify-core.min.js"
);

const { Signer } = aws_amplify_core;

let credentials;
let region;

self.addEventListener("install", (event) => {
  // install immediately
  event.waitUntil(self.skipWaiting());
});

self.addEventListener("activate", (event) => {
  // control clients ASAP
  event.waitUntil(self.clients.claim());
});

self.addEventListener("message", (event) => {
  const {
    data: { credentials: newCredentials, region: newRegion },
  } = event;

  if (newCredentials != null) {
    credentials = newCredentials;
  }
```

```
  if (newRegion != null) {
    region = newRegion;
  }
});

async function signedFetch(request) {
  const url = new URL(request.url);
  const path = url.pathname.slice(1).split("/");

  // update URL to point to Amazon Location
  url.pathname = `/maps/v0/maps/${path[0]}/tiles/${path.slice(1).join("/")}`;
  url.host = `maps.geo.${region}.amazonaws.com`;
  // strip params (Tangram generates an empty api_key param)
  url.search = "";

  const signed = Signer.signUrl(url.toString(), {
    access_key: credentials.accessKeyId,
    secret_key: credentials.secretAccessKey,
    session_token: credentials.sessionToken,
  });

  return fetch(signed);
}

self.addEventListener("fetch", (event) => {
  const { request } = event;

  // match the synthetic hostname we're telling Tangram to use
  if (request.url.includes("amazon.location")) {
    return event.respondWith(signedFetch(request));
  }

  // fetch normally
  return event.respondWith(fetch(request));
});
```

このサンプルは、 の Amazon Location Service サンプルリポジトリの一部として入手できますGitHub。

Amazon Location Service で Android 用 Tangram ES を使用する

Tangram ES は、OpenGL ES を使用してベクトルデータから 2D および 3D マップをレンダリングする C++ ライブラリです。Tangram のネイティブ版です。

Tilezen スキーマで動作するように構築されたTangram スタイルは、HERE のマップを使用する場合には Amazon Location とほぼ互換性があります。具体的には次のとおりです。

- Bubble Wrap — 興味のある地点をわかりやすく表示する便利なアイコンが付いた、フル機能の道案内スタイルです。
- Cinnabar — クラシックな外観で、一般的なマップアプリに適しています。
- Refill — Stamen Designの独創的な Toner スタイルにインスパイアされた、データ視覚化オーバーレイ用にデザインされたミニマルなマップスタイルです。
- Tron — TRONのビジュアル言語によるスケール返還を探求したスタイルです。
- Walkabout — ハイキングや外出に最適なアウトドア重視のスタイルです。

このガイドでは、Cinnabar と呼ばれる Tangram スタイルを使用して、Android 用 Tangram ES を Amazon Location と統合する方法について説明します。このサンプルは、 の Amazon Location Service サンプルリポジトリの一部として入手できますGitHub。

他の Tangram スタイルには地形情報をエンコードするラスタータイルが最適ですが、この機能はまだ Amazon Location ではサポートされていません。

> **Important**
>
> 以下のチュートリアルの Tangram スタイルは、`VectorHereContrast` スタイルで設定された Amazon Location マップリソースとのみ互換性があります。

アプリケーションの構築：初期化

アプリケーションを初期化するには：

1. [空のアクティビティ] テンプレートから新しい Android Studio プロジェクトを作成します。
2. プロジェクト言語に [Kotlin] が選択されていることを確認します。
3. [Minimum SDK of API 16: Android 4.1 (Jelly Bean)] 以降を選択してください。
4. [プロジェクト構造]を開き、[ファイル]、[プロジェクト構造...] に移動します。[依存関係] セクションを選択します。
5. [<All Modules>] を選択したら、[+] ボタンを選択して[新しいライブラリ依存関係]を追加します。
6. [AWS Android SDK] バージョンは、2.19.1 以降を追加します。例：`com.amazonaws:aws-android-sdk-core:2.19.1`

7. [Tangram] バージョンは、0.13.0 以降を追加します。例: `com.mapzen.tangram:tangram:0.13.0`。

> **Note**
>
> [Tangram] の検索：`com.mapzen.tangram:tangram:0.13.0` は、「見つかりません」というメッセージを表示しますが、[OK] を選択すると追加できます。

アプリケーションの構築：設定

アプリケーションにリソースと AWS リージョンをを設定するには：

1. `app/src/main/res/values/configuration.xml` を作成します。
2. リソースの名前と ID、および AWS リージョン を入力します。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="identityPoolId">us-east-1:54f2ba88-9390-498d-aaa5-0d97fb7ca3bd</
string>
    <string name="mapName">TangramExampleMap</string>
    <string name="awsRegion">us-east-1</string>
    <string name="sceneUrl">https://www.nextzen.org/carto/cinnabar-style/9/cinnabar-
style.zip</string>
    <string name="attribution">© 2020 HERE</string>
</resources>
```

アプリケーションの構築：アクティビティレイアウト

`app/src/main/res/layout/activity_main.xml` を編集します。

- マップをレンダリングする `MapView` を追加します。これにより、マップの初期中心点も設定されます。
- アトリビューションを表示する `TextView` を追加してください。

これにより、マップの初期中心点も設定されます。

> **Note**
>
> 使用する各データプロバイダーのワードマークまたはテキストアトリビューションを、アプリケーションまたはドキュメントに記載する必要があります。アトリビューション文字列は、スタイル記述子のレスポンスの`sources.esri.attribution`、`sources.here.attribution`、`source.grabmaptiles.att`キーに含まれています。
>
> Tangram はこれらのリソースをリクエストせず、HERE のマップとのみ互換性があるため、「© 2020 HERE」を使用してください。Amazon Location リソースをデータプロバイダーと併用する場合は、サービスの利用規約を必ずお読みください。

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.mapzen.tangram.MapView
        android:id="@+id/map"
        android:layout_height="match_parent"
        android:layout_width="match_parent" />

    <TextView
        android:id="@+id/attributionView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#80808080"
        android:padding="5sp"
        android:textColor="@android:color/black"
        android:textSize="10sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        tools:ignore="SmallSp" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

アプリケーションの構築：リクエスト変換

SigV4Interceptor リクエストをインターセプトし、署名バージョン 4 を使用して署名する AWS という名前のクラスを作成します。この情報は、メインアクティビティの作成時にマップリソースの取得に使用される HTTP クライアントに登録されます。

```
package aws.location.demo.okhttp

import com.amazonaws.DefaultRequest
import com.amazonaws.auth.AWS4Signer
import com.amazonaws.auth.AWSCredentialsProvider
import com.amazonaws.http.HttpMethodName
import com.amazonaws.util.IOUtils
import okhttp3.HttpUrl
import okhttp3.Interceptor
import okhttp3.Request
import okhttp3.Response
import okio.Buffer
import java.io.ByteArrayInputStream
import java.net.URI

class SigV4Interceptor(
    private val credentialsProvider: AWSCredentialsProvider,
    private val serviceName: String
) : Interceptor {
    override fun intercept(chain: Interceptor.Chain): Response {
        val originalRequest = chain.request()

        if (originalRequest.url().host().contains("amazonaws.com")) {
            val signer = if (originalRequest.url().encodedPath().contains("@")) {
                // the presence of "@" indicates that it doesn't need to be double URL-
encoded
                AWS4Signer(false)
            } else {
                AWS4Signer()
            }

            val awsRequest = toAWSRequest(originalRequest, serviceName)
            signer.setServiceName(serviceName)
            signer.sign(awsRequest, credentialsProvider.credentials)

            return chain.proceed(toSignedOkHttpRequest(awsRequest, originalRequest))
        }
```

```
        return chain.proceed(originalRequest)
    }

    companion object {
        fun toAWSRequest(request: Request, serviceName: String): DefaultRequest<Any> {
            // clone the request (AWS-style) so that it can be populated with
credentials
            val dr = DefaultRequest<Any>(serviceName)

            // copy request info
            dr.httpMethod = HttpMethodName.valueOf(request.method())
            with(request.url()) {
                dr.resourcePath = uri().path
                dr.endpoint = URI.create("${scheme()}://${host()}")

                // copy parameters
                for (p in queryParameterNames()) {
                    if (p != "") {
                        dr.addParameter(p, queryParameter(p))
                    }
                }
            }

            // copy headers
            for (h in request.headers().names()) {
                dr.addHeader(h, request.header(h))
            }

            // copy the request body
            val bodyBytes = request.body()?.let { body ->
                val buffer = Buffer()
                body.writeTo(buffer)
                IOUtils.toByteArray(buffer.inputStream())
            }

            dr.content = ByteArrayInputStream(bodyBytes ?: ByteArray(0))

            return dr
        }

        fun toSignedOkHttpRequest(
            awsRequest: DefaultRequest<Any>,
            originalRequest: Request
```

```
        ): Request {
            // copy signed request back into an OkHttp Request
            val builder = Request.Builder()

            // copy headers from the signed request
            for ((k, v) in awsRequest.headers) {
                builder.addHeader(k, v)
            }

            // start building an HttpUrl
            val urlBuilder = HttpUrl.Builder()
                .host(awsRequest.endpoint.host)
                .scheme(awsRequest.endpoint.scheme)
                .encodedPath(awsRequest.resourcePath)

            // copy parameters from the signed request
            for ((k, v) in awsRequest.parameters) {
                urlBuilder.addQueryParameter(k, v)
            }

            return builder.url(urlBuilder.build())
                .method(originalRequest.method(), originalRequest.body())
                .build()
        }
    }
}
```

アプリケーションの構築：メインアクティビティ

メインアクティビティは、ユーザーに表示されるビューを初期化します。これには以下が含まれます。

- Amazon Cognito `CredentialsProvider` をインスタンス化します。
- 署名バージョン 4をインターセプターとして登録します。
- マップスタイルを指定してマップを構成し、タイルの URL をオーバーライドして、適切なアトリビューションを表示します。

また、`MainActivity` には、ライフサイクルイベントをマップビューに転送する役割もあります。

```
package aws.location.demo.tangram
```

```
import android.os.Bundle
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import aws.location.demo.okhttp.SigV4Interceptor
import com.amazonaws.auth.CognitoCachingCredentialsProvider
import com.amazonaws.regions.Regions
import com.mapzen.tangram.*
import com.mapzen.tangram.networking.DefaultHttpHandler
import com.mapzen.tangram.networking.HttpHandler

private const val SERVICE_NAME = "geo"

class MainActivity : AppCompatActivity(), MapView.MapReadyCallback {
    private var mapView: MapView? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main)

        mapView = findViewById(R.id.map)

        mapView?.getMapAsync(this, getHttpHandler())
        findViewById<TextView>(R.id.attributionView).text =
 getString(R.string.attribution)
    }

    override fun onMapReady(mapController: MapController?) {
        val sceneUpdates = arrayListOf(
            SceneUpdate(
                "sources.mapzen.url",
                "https://maps.geo.${getString(R.string.awsRegion)}.amazonaws.com/maps/
v0/maps/${
                    getString(
                        R.string.mapName
                    )
                }/tiles/{z}/{x}/{y}"
            )
        )

        mapController?.let { map ->
            map.updateCameraPosition(
                CameraUpdateFactory.newLngLatZoom(
                    LngLat(-123.1187, 49.2819),
```

```
                    12F
                )
            )
            map.loadSceneFileAsync(
                getString(R.string.sceneUrl),
                sceneUpdates
            )
        }
    }

    private fun getHttpHandler(): HttpHandler {
        val builder = DefaultHttpHandler.getClientBuilder()

        val credentialsProvider = CognitoCachingCredentialsProvider(
            applicationContext,
            getString(R.string.identityPoolId),
            Regions.US_EAST_1
        )

        return DefaultHttpHandler(
            builder.addInterceptor(
                SigV4Interceptor(
                    credentialsProvider,
                    SERVICE_NAME
                )
            )
        )
    }

    override fun onResume() {
        super.onResume()
        mapView?.onResume()
    }

    override fun onPause() {
        super.onPause()
        mapView?.onPause()
    }

    override fun onLowMemory() {
        super.onLowMemory()
        mapView?.onLowMemory()
    }
```

```
    override fun onDestroy() {
        super.onDestroy()
        mapView?.onDestroy()
    }
}
```

このアプリケーションを実行すると、選択したスタイルで全画面マップが表示されます。このサンプルは、 の Amazon Location Service サンプルリポジトリの一部として入手できますGitHub。

Amazon Location Service で iOS 用 Tangram ES を使用する

Tangram ES は、OpenGL ES を使用してベクトルデータから 2D および 3D マップをレンダリングする C++ ライブラリです。Tangram のネイティブ版です。

Tilezen スキーマで動作するように構築されたTangram スタイルは、HERE のマップを使用する場合には Amazon Location とほぼ互換性があります。具体的には次のとおりです。

- Bubble Wrap — 興味のある地点をわかりやすく表示する便利なアイコンが付いた、フル機能の道案内スタイルです。
- Cinnabar — クラシックな外観で、一般的なマップアプリに適しています。
- Refill — Stamen Designの独創的な Toner スタイルにインスパイアされた、データ視覚化オーバーレイ用にデザインされたミニマルなマップスタイルです。
- Tron — TRONのビジュアル言語によるスケール返還を探求したスタイルです。
- Walkabout — ハイキングや外出に最適なアウトドア重視のスタイルです。

このガイドでは、Cinnabar と呼ばれる Tangram スタイルを使用して、iOS 用 Tangram ES を Amazon Location と統合する方法について説明します。このサンプルは、 の Amazon Location Service サンプルリポジトリの一部として入手できますGitHub。

他の Tangram スタイルには地形情報をエンコードするラスタータイルが最適ですが、この機能はまだ Amazon Location ではサポートされていません。

> **Important**
>
> 以下のチュートリアルの Tangram スタイルは、`VectorHereContrast` スタイルで設定された Amazon Location マップリソースとのみ互換性があります。

アプリケーションの構築：初期化

アプリケーションを初期化するには：

1. [App] テンプレートから新しい Xcode プロジェクトを作成します。
2. インターフェースには [SwiftUI] を選択します。
3. ライフサイクルには [SwiftUI] アプリケーションを選択します。
4. 言語には [Swift] を選択します。

アプリケーションの構築：依存関係の追加

依存関係を追加するには、 などの依存関係マネージャーを使用できますCocoaPods。

1. ターミナルに をインストールします CocoaPods。

```
sudo gem install cocoapods
```

2. アプリケーションのプロジェクトディレクトリに移動し、パッケージマネージャーを使用して Podfile を CocoaPods初期化します。

```
pod init
```

3. Podfile を開いて、`AWSCore` と `Tangram-es` を依存関係として追加します。

```
platform :ios, '12.0'

target 'Amazon Location Service Demo' do
  use_frameworks!

  pod 'AWSCore'
  pod 'Tangram-es'
end
```

4. 依存関係をダウンロードしインストールします。

```
pod install --repo-update
```

5. が CocoaPods 作成した Xcode ワークスペースを開きます。

```
xed .
```

アプリケーションの構築：設定

Info.plist に以下のキーと値を追加してアプリケーションを設定し、テレメトリを無効にします。

キー	値
AWSRegion	us-east-1
IdentityPoolId	us-east-1:54 f2ba88-9390-498d-aaa5-0d97f b7ca3bd
MapName	ExampleMap
SceneURL	https://www.nextzen.org/carto/cinnabar-style/9/ cinnabar-style.zip

アプリケーションの構築： ContentView レイアウト

マップをレンダリングするには、`ContentView.swift` を編集します。

- マップをレンダリングする `MapView` を追加します。
- アトリビューションを表示する `TextField` を追加します。

これにより、マップの初期中心点も設定されます。

> **Note**
>
> 使用する各データプロバイダーのワードマークまたはテキストアトリビューションを、アプリケーションまたはドキュメントに記載する必要があります。アトリビューション文字列は、スタイル記述子のレスポンスの`sources.esri.attribution`、`sources.here.attribution`、`source.grabmaptiles.att`キーに含まれています。Amazon Location リソースをデータプロバイダーと併用する場合は、サービスの利用規約を必ずお読みください。

```
import SwiftUI
import TangramMap
```

```
struct ContentView: View {
    var body: some View {
        MapView()
            .cameraPosition(TGCameraPosition(
                                center: CLLocationCoordinate2DMake(49.2819, -123.1187),
                                zoom: 10,
                                bearing: 0,
                                pitch: 0))
            .edgesIgnoringSafeArea(.all)
            .overlay(
                Text("© 2020 HERE")
                    .disabled(true)
                    .font(.system(size: 12, weight: .light, design: .default))
                    .foregroundColor(.black)
                    .background(Color.init(Color.RGBColorSpace.sRGB, white: 0.5,
 opacity: 0.5))
                    .cornerRadius(1),
                alignment: .bottomTrailing)
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

アプリケーションの構築：リクエスト変換

AWSSignatureV4URLHandler.swift という名前の Swift ファイルを作成し、AWS リクエストをインターセプトし、署名バージョン 4 を使用して署名するクラスを定義します。これは Tangram MapView 内で、URL ハンドラーとして登録されます。

```
import AWSCore
import TangramMap

class AWSSignatureV4URLHandler: TGDefaultURLHandler {
    private let region: AWSRegionType
    private let identityPoolId: String
    private let credentialsProvider: AWSCredentialsProvider

    init(region: AWSRegionType, identityPoolId: String) {
        self.region = region
```

```
        self.identityPoolId = identityPoolId
        self.credentialsProvider = AWSCognitoCredentialsProvider(regionType: region,
identityPoolId: identityPoolId)
        super.init()
    }

    override func downloadRequestAsync(_ url: URL, completionHandler: @escaping
TGDownloadCompletionHandler) -> UInt {
        if url.host?.contains("amazonaws.com") != true {
            // not an AWS URL
            return super.downloadRequestAsync(url, completionHandler:
completionHandler)
        }

        // URL-encode spaces, etc.
        let keyPath = String(url.path.dropFirst())
        guard let keyPathSafe =
keyPath.addingPercentEncoding(withAllowedCharacters: .urlPathAllowed) else {
            print("Invalid characters in path '\(keyPath)'; unsafe to sign")
            return super.downloadRequestAsync(url, completionHandler:
completionHandler)
        }

        // sign the URL
        let endpoint = AWSEndpoint(region: region, serviceName: "geo", url: url)
        let requestHeaders: [String: String] = ["host": endpoint!.hostName]
        let task = AWSSignatureV4Signer
            .generateQueryStringForSignatureV4(
                withCredentialProvider: credentialsProvider,
                httpMethod: .GET,
                expireDuration: 60,
                endpoint: endpoint!,
                keyPath: keyPathSafe,
                requestHeaders: requestHeaders,
                requestParameters: .none,
                signBody: true)
        task.waitUntilFinished()

        if let error = task.error as NSError? {
            print("Error occurred: \(error)")
        }

        if let result = task.result {
            // have Tangram fetch the signed URL
```

```
            return super.downloadRequestAsync(result as URL, completionHandler:
 completionHandler)
        }

        // fall back to an unsigned URL
        return super.downloadRequestAsync(url, completionHandler: completionHandler)
    }
}
```

アプリケーションの構築：マップビュー

マップビューは、AWSSignatureV4Delegate インスタンスを初期化し、リソースを取得してマップをレンダリングする基盤となる MGLMapView の構成を行います。また、スタイル記述子のソースからにアトリビューション文字列を ContentView に伝達する処理も行います。

以下の struct の定義を含む、MapView.swift という名前の新しい Swift ファイルを作成します。

```
import AWSCore
import TangramMap
import SwiftUI

struct MapView: UIViewRepresentable {
    private let mapView: TGMapView

    init() {
        let regionName = Bundle.main.object(forInfoDictionaryKey: "AWSRegion") as!
 String
        let identityPoolId = Bundle.main.object(forInfoDictionaryKey: "IdentityPoolId")
 as! String
        let mapName = Bundle.main.object(forInfoDictionaryKey: "MapName") as! String
        let sceneURL = URL(string: Bundle.main.object(forInfoDictionaryKey: "SceneURL")
 as! String)!

        let region = (regionName as NSString).aws_regionTypeValue()

        // rewrite tile URLs to point at AWS resources
        let sceneUpdates = [
            TGSceneUpdate(path: "sources.mapzen.url",
                          value: "https://maps.geo.\(regionName).amazonaws.com/maps/v0/
maps/\(mapName)/tiles/{z}/{x}/{y}")]

        // instantiate a TGURLHandler that will sign AWS requests
```

```
        let urlHandler = AWSSignatureV4URLHandler(region: region, identityPoolId:
 identityPoolId)

        // instantiate the map view and attach the URL handler
        mapView = TGMapView(frame: .zero, urlHandler: urlHandler)

        // load the map style and apply scene updates (properties modified at runtime)
        mapView.loadScene(from: sceneURL, with: sceneUpdates)
    }

    func cameraPosition(_ cameraPosition: TGCameraPosition) -> MapView {
        mapView.cameraPosition = cameraPosition

        return self
    }

    // MARK: - UIViewRepresentable protocol

    func makeUIView(context: Context) -> TGMapView {
        return mapView
    }

    func updateUIView(_ uiView: TGMapView, context: Context) {
    }
}
```

このアプリケーションを実行すると、選択したスタイルで全画面マップが表示されます。このサンプルは、 の Amazon Location Service サンプルリポジトリの一部として入手できますGitHub。

マップ上にデータフィーチャを描画する

Amplify、 MapLibreまたはTAKgram を使用してマップをレンダリングするアプリケーションを作成したら、マップの上に特徴を描画するのが自然な次のステップです。たとえば、顧客の所在地をマップ上のマーカーとしてレンダリングしたいかもしれません。

一般に、場所検索関数を使用してデータから場所を検索し、Amplify の機能 MapLibre、または TAKgram を使用してその場所をレンダリングできます。

さまざまなタイプのオブジェクトをマップ上にレンダリングするサンプルについては、次の MapLibre サンプルを参照してください。

- 例：マーカーの描画

- 例：クラスター化されたポイントを描画
- 例：ポリゴンを描画

その他のサンプルやチュートリアルについては、「Amazon Location Service を使用するためのコードサンプルとチュートリアル」を参照してください。

を使用してマップの拡張を設定する MapLibre

ユーザーに全世界を画面移動したりズームしてほしくない場合もあるでしょう。 MapLibreのマップコントロールを使用している場合は、 maxBoundsオプションを使用してマップコントロールの範囲または境界 を制限し、 minZoom および maxZoomオプションを使用してズームを制限できます。

次のコード例は、マップコントロールを初期化して画面移動を特定の境界 (この場合は Grab データソースの範囲) に制限する方法を示しています。

> **Note**
>
> これらのサンプルは にあり JavaScript、ウェブ ACL の作成チュートリアルのコンテキスト内で動作します。

```
// Set bounds to Grab data provider region
var bounds = [
        [90.0, -21.943045533438166], // Southwest coordinates
        [146.25, 31.952162238024968] // Northeast coordinates
    ];

var mlglMap = new maplibregl.Map(
    {
    container: 'map',
    style: mapName,
    maxBounds: bounds // Sets bounds as max
    transformRequest,
    }
);
```

同様に、マップの最小ズームレベルと最大ズームレベルを設定できます。どちらの値も 0 ~ 24 の範囲ですが、デフォルトは最小ズームが 0、最大ズームが 22 です (データプロバイダーは、すべてのズームレベルでデータを提供するとは限りません。ほとんどのマップライブラリはこれを自動的に処

理します)。次の例では、 MapLibre マップコントロールの `minZoom`および `maxZoom`オプションを初期化します。

```
// Set the minimum and maximum zoom levels
var mlglMap = new maplibregl.Map(
    {
    container: 'map',
    style: mapName,
    maxZoom: 12,
    minZoom: 5,
    transformRequest,
    }
);
```

> **Tip**
>
> MapLibre マップコントロールでは、 `get...`および `set...`関数を使用して、初期化中ではなく実行時にこれらのオプションを設定することもできます。たとえば、`getMaxBounds` と `setMaxBounds` を使用してランタイムにマップ境界を変更できます。

マップリソースの管理

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用してマップリソースを管理できます。

マップリソースのリスト

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用してマップリソースのリストを表示できます。

Console

Amazon Location コンソールを使用して既存のマップリソースのリストを表示するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左側のナビゲーションペインで、[マップ] を選択します。
3. [マイマップ] でマップリソースのリストを表示します。

API

Amazon Location Maps API の ListMaps オペレーションを使用してください。

以下の例は、AWS アカウント内のマップリソースのリストを取得する API リクエストです。

```
POST /maps/v0/list-maps
```

以下に、ListMaps のレスポンスの例を示します。

```
{
   "Entries": [
      {
         "CreateTime": 2020-10-30T01:38:36Z,
         "DataSource": "Esri",
         "Description": "string",
         "MapName": "ExampleMap",
         "UpdateTime": 2020-10-30T01:38:36Z
      }
   ],
   "NextToken": "1234-5678-9012"
}
```

CLI

list-mapコマンドを実行します。

次の例は、AWS アカウント内のマップリソースのリストを取得するための AWS CLI です。

```
aws location list-maps
```

マップリソースの詳細を取得します

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、AWS アカウントのすべてのマップリソースに関する詳細を取得できます。

Console

Amazon Location コンソールを使用してマップリソースの詳細を表示するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。

2. 左側のナビゲーションペインで、[マップ] を選択します。
3. [マイマップ] で、ターゲットマップリソースの名前リンクを選択します。

API

Amazon Location Maps API の DescribeMap オペレーションを使用してください。

次の例は、 のマップリソースの詳細を取得するための API リクエストです*ExampleMap*。

```
GET /maps/v0/maps/ExampleMap
```

以下に、DescribeMap のレスポンスの例を示します。

```
{
   "Configuration": {
      "Style": "VectorEsriNavigation"
   },
   "CreateTime": 2020-10-30T01:38:36Z,
   "DataSource": "Esri",
   "Description": "string",
   "MapArn": "arn:aws:geo:us-west-2:123456789012:maps/ExampleMap",
   "MapName": "ExampleMap",
   "Tags": {
      "Tag1" : "Value1"
   },
   "UpdateTime": 2020-10-30T01:40:36Z
}
```

CLI

describe-mapコマンドを実行します。

次の例は、 のマップリソースの詳細を取得AWS CLIするための です*ExampleMap*。

```
aws location describe-map \
    --map-name "ExampleMap"
```

マップリソースを削除する

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、AWSアカウントからマップリソースを削除できます。

> **Warning**
> このオペレーションはリソースを完全に削除します。

Console

Amazon Location コンソールを使用して既存のマップリソースを削除するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左側のナビゲーションペインで、[マップ] を選択します。
3. [マイマップ] リスト で、リストからターゲットマップを選択します。
4. マップの削除 を選択します。

API

Amazon Location Maps API の `DeleteMap` オペレーションを使用してください。

次の例は、マップリソース を削除する API リクエストです*ExampleMap*。

```
DELETE /maps/v0/maps/ExampleMap
```

以下に、`DeleteMap` の正常なレスポンスの例を示します。

```
HTTP/1.1 200
```

CLI

`delete-map`コマンドを実行します。

次の例は、マップリソース を削除する AWS CLI コマンドです*ExampleMap*。

```
aws location delete-map \
    --map-name "ExampleMap"
```

Amazon Location を使用して場所と位置情報を検索する

Amazon Location には、選択したプロバイダーの位置情報場所データを検索する機能が含まれています。検索にはいくつかの種類があります。

- ジオコーディング — ジオコーディングは、テキスト入力に基づいて、住所、地域、会社名、またはその他の興味のある地点を検索するプロセスです。見つかった結果の詳細と場所 (緯度と経度) が返されます。
- リバースジオコーディング — リバースジオコーディングでは、特定の場所に近い場所を検索することができます。
- オートコンプリート — オートコンプリートは、ユーザーがクエリを入力すると自動的に候補が表示されるプロセスです。たとえば、ユーザーが **Par** を入力した場合、1 つの提案は`Paris, France`になる可能性があります。

Amazon Location では、場所インデックスリソースを作成して設定することで、場所検索操作のデータプロバイダーを選択することができます。

リソースを作成したら、希望する言語の AWS SDK、Amplify、または REST API エンドポイントを使用してリクエストを送信できます。レスポンスからのデータを使用して、地図上の位置をマークしたり、位置データを充実させたり、位置を人間が読めるテキストに変換したりするなどのことができます。

> **Note**
>
> 検索場所の概念の概要については、場所検索を参照してください。

トピック

- 場所インデックスリソースの管理

前提条件

ジオコーディング、リバースジオコーディング、または場所の検索を開始する前に、前提条件となる手順に従ってください。

トピック

- 場所インデックスリソースの作成
- リクエストを認証する

場所インデックスリソースの作成

まず、 AWS アカウントに場所インデックスリソースを作成します。

プレイスインデックスリソースを作成すると、ジオコーディング、リバースジオコーディング、検索のクエリをサポートするデータプロバイダーを選択できます。

1. Esri — 対象地域における Esri のカバレッジの詳細については、Esri のカバレッジの詳細については、Esri のドキュメントの Esri のカバレッジの詳細については、Esri のジオコーディングカバレッジを参照してください。
2. HERE テクノロジー — 対象地域における HERE のカバレッジの詳細については、HERE ドキュメントの HERE ジオコーディングカバレッジを参照してください。
3. Grab — Grab は東南アジアのデータのみ提供します。イベントの詳細については、このガイドの「国/地域と対象地域 」を参照してください。

これは、Amazon Location Service コンソール、 AWS CLI、または Amazon Location APIsを使用して実行できます。

Console

Amazon Location Service コンソールを使用してプレイスインデックスリソースを作成するには

1. Amazon Location Service コンソール (https://console.aws.amazon.com/location/) を開きます。
2. 左のナビゲーションペインで、プレース· インデックス を選択します。
3. インデックスの作成 を選択します。

4. 次のボックスに入力します。

   - 名前 — プレイスインデックスリソースの名前を入力します。例えば、 で すExamplePlaceIndex。最大 100 文字。には、英数字、ピリオド (.)、ハイフン (-)、アンダースコア (_) を含めることができます。
   - 説明 – 任意の説明を入力します。

5. 「データプロバイダー」で、プレイスインデックスリソースで使用できるデータプロバイダーを選択します。

   > **Note**
   >
   > 配送車両や従業員など、ビジネスで使用する資産をアプリケーションで追跡またはルーティングする場合、位置情報プロバイダーに Esri を使用してはいけません。詳細については、「AWS サービス規約」のセクション 82 を参照してください。

6. 「データストレージオプション」で、プレイスインデックスリソースからの検索結果を保存するかどうかを指定します。
7. （オプショナル）タグに、タグのキーと値を入力します。これにより、新しいプレイスインデックスリソースにタグが追加されます。詳細については、リソースにタグを付けるを参照してください。
8. インデックスの作成 を選択します。

API

Amazon Location API を使用してプレイスインデックスリソースの作成

Amazon Location プレース API CreatePlaceIndex のオペレーションを利用してください。

次の例は、データプロバイダー *Esri* *ExamplePlaceIndex*を使用して という場所インデックスリソースを作成する API リクエストです。

```
POST /places/v0/indexes
Content-type: application/json

{
   "DataSource": "Esri",
   "DataSourceConfiguration": {
      "IntendedUse": "SingleUse"
```

```
    },
    "Description": "string",
    "IndexName": "ExamplePlaceIndex",
    "Tags": {
       "Tag1" : "Value1"
    }
}
```

AWS CLI

AWS CLI コマンドを使用して場所インデックスリソースの作成

create-place-indexコマンドを実行します。

次の例では、データプロバイダーとして *Esri* *ExamplePlaceIndex*を使用して、 という場所インデックスリソースを作成します。

```
aws location \
  create-place-index \
  --data-source "Esri" \
  --description "Example place index" \
  --index-name "ExamplePlaceIndex" \
  --tags Tag1=Value1
```

> **Note**
>
> 請求は使用状況によって異なります。他の AWS サービスの利用料金が請求される場合があります。詳細については、「Amazon Location Service 料金」を参照してください。

リクエストを認証する

プレイスインデックスリソースを作成し、アプリケーションに位置情報を組み込む準備ができたら、リクエストの認証方法を選択します。

- サービスにアクセスする方法については、「Amazon Location Service へのアクセス」を参照してください。
- ウェブサイトには匿名機能を利用されるユーザーがいらっしゃる場合、API キーまたは Amazon Cognito を使うのはお勧めです。

例

次の例は、AWS JavaScript SDK v3 と Amazon Location を使用して、認証に API キーを使用する方法を示していますJavaScript 認証ヘルパー。

```
import { LocationClient, SearchPlaceIndexForTextCommand } from "@aws-sdk/client-
location";
import { withAPIKey } from "@aws/amazon-location-utilities-auth-helper";

const apiKey = "v1.public.your-api-key-value"; // API key

// Create an authentication helper instance using an API key
const authHelper = await withAPIKey(apiKey);

const client = new LocationClient({
  region: "<region>", // region containing Cognito pool
  ...authHelper.getLocationClientConfig(), // Provides configuration required to make
 requests to Amazon Location
});

const input = {
  IndexName: "ExamplePlaceIndex",
  Text: "Anyplace",
  BiasPosition: [-123.4567, 45.6789]
};

const command = new SearchPlaceIndexForTextCommand(input);

const response = await client.send(command);
```

ジオコーディング

ジオコーディングは、住所、地域、会社名、対象地点などのテキストを一連の地理座標に変換するプロセスです。プレイスインデックスリソースを使用してジオコーディングリクエストを送信し、ジオコーディングから取得したデータを組み込んで Web またはモバイルアプリケーションのマップ上にデータを表示することができます。

このセクションでは、簡単なジオコーディングリクエストを送信する方法と、オプション仕様でジオコーディングリクエストを送信する方法について説明します。

ジオコーディング

住所を座標セットに変換するSearchPlaceIndexForTextオペレーションを使用して、ジオコーディングに簡単なリクエストを送信することができます。単純なリクエストには次の必須パラメータが含まれます。

- `Text`— 座標セットに変換する住所、名前、都市、または地域。たとえば、文字列`Any Town`。

1 ページあたりの結果の最大数を指定するには、以下のオプションパラメータを使用します。

- `MaxResults`— クエリレスポンスで返される結果の数を制限します。

AWS CLI または Amazon Location APIsを使用できます。

API

次の例は、場所インデックスリソース で*ExamplePlaceIndex*、Any *Town* と呼ばれる住所、名前、都市、または地域を検索するSearchPlaceIndexForTextリクエストです。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
    "Text": "Any Town",
    "MaxResults": 10
}
```

AWS CLI

次の例は、場所インデックスリソース で*ExamplePlaceIndex*、Any *Town* と呼ばれる住所、名前、都市、または地域を検索するsearch-place-index-for-textコマンドです。

```
aws location \
    search-place-index-for-text \
        --index-name ExamplePlaceIndex \
        --text "Any Town" \
        --max-results 10
```

ある位置の近くのジオコード

ジオコーディングでは、以下のオプションパラメータを使用して特定の位置の近くをジオコーディングすることができます。

- `BiasPosition`— 付近で検索したい位置。指定した位置に最も近い結果を検索して、検索範囲を絞り込みます。`[longitude, latitude]`と定義されます。

次の例は、*-123.4567*, *45.6789*という位置の近くにある「*Any Town*」という住所、名前、都市、または地域をプレイスインデックスリソースで検索するSearchPlaceIndexForTextリクエストです。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Any Town",
   "BiasPosition": [-123.4567,45.6789]
}
```

バウンディングボックス内のジオコーディング

バウンディングボックス内でジオコーディングすると、次のオプションパラメータを使用して結果を特定の境界内の座標に絞り込むことができます。

- `FilterBBox`— 検索結果をバウンディングボックス内の座標に絞り込むために指定するバウンディングボックス。`[LongitudeSW, LatitudeSW, LongitudeNE, LatitudeNE]`と定義されます。

> **Note**
>
> リクエストは`FilterBBox`と`BiasPosition`の両方のパラメーターを含むことはできません。リクエストで両方のパラメータを指定すると`ValidationException`エラーが返されます。

次の例は、「*Any Town*」呼ばれる住所、名前、都市、または地域のバウンディングボックス内を検索するSearchPlaceIndexForTextリクエストです。バウンディングボックスは次のようになります。

- 南西の角の経度は *-124.1450* です。
- 南西の角の緯度は *41.7045* です。
- 北東の角の経度は *-124.1387* です。
- 北東の角の緯度は *41.7096* です。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Any Town",
   "FilterBBox":  [
        -124.1450,41.7045,
        -124.1387,41.7096
    ]
}
```

国内のジオコード

以下のオプションパラメータを使用することで、指定した 1 つまたは複数の国でジオコードを行うことができます：

- FilterCountries— ジオコーディングする国または地域。ISO 3166 の 3 文字の国コードを使用すると、1 回のリクエストで最大 100 カ国まで定義できます。たとえば、オーストラリアではAUSです。

次の例は、ドイツとフランスの「*Any Town*」という住所、名前、都市、または地域を検索するSearchPlaceIndexForTextリクエストです。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Any Town",
   "FilterCountries": ["DEU","FRA"]
}
```

カテゴリによるフィルタリング

以下のオプションのパラメータを使用することで、ジオコードリクエストで返されるカテゴリをフィルタリングすることができます：

- FilterCategories— クエリで返してほしい結果のカテゴリ。1 つのリクエストで最大 5 つのカテゴリを指定することができます。Amazon Location Service カテゴリのリストは、カテゴリセクションにあります。例えば、Hotelを指定することで、クエリで返されるホテルのみを指定することができます。

次の例は、米国にある *Hometown Coffee* というコーヒーショップを検索するSearchPlaceIndexForTextリクエストです。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Hometown Coffee",
   "FilterCategories": ["Coffee Shop"],
   "FilterCountries": ["USA"]
}
```

カテゴリに基づくフィルタリングの詳細については、カテゴリに基づくフィルタリングの詳細については、カテゴリーの配置と結果のフィルタリングを参照してください。

優先言語でのジオコーディング

オプションLanguageパラメータを使用して、検索結果の言語プリファレンスを設定できます。たとえば、**100 Main St, Anytown, USA**を検索するとデフォルトで100 Main St, Any Town, USAが返されるか可能性があります。ただし、frとしてLanguageを選択した場合、結果は代わりに100 Rue Principale, Any Town, États-Unisを返す場合もあります。

- Language— クエリの結果をレンダリングするために使用する言語コード。値は有効な BCP 47 言語コードでなければなりません。例えば、英語では en です。

> **Note**
>
> Languageが指定されていない場合、または指定された言語が結果でサポートされていない場合、その結果ではパートナーのデフォルト言語が使用されます。

次の例は、優先言語をdeに指定して、**Any Town**という場所を検索するSearchPlaceIndexforTextリクエストである。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json
{
   "Text": "Any Town",
   "Language": "de"
}
```

レスポンスの例

Example

以下は、Amazon Location プレイス API からSearchPlaceIndexForTextオペレーションを呼び出したときのレスポンスの例です。結果には、関連する場所とリクエストの概要が含まれます。Esri または Here をパートナーとして選択したことに基づいて、2 つの応答が表示されます。

Example request

```
POST /places/v0/indexes/ExamplePlaceIndex/search/text
Content-type: application/json

{
   "Text": "Amazon",
   "MaxResults": 1,
   "FilterCountries": ["USA"],
   "BiasPosition": [-112.10, 46.32]
}
```

Example response (Esri)

```
{
    "Results": [
        {
```

```
            "Place": {
                "Country": "USA",
                "Geometry": {
                    "Point": [
                        -112.10667999999998,
                        46.319090000000074
                    ]
                },
                "Interpolated": false,
                "Label": "Amazon, MT, USA",
                "Municipality": "Amazon",
                "Region": "Montana",
                "SubRegion": "Jefferson County"
            },
            "Distance": 523.4619749879726,
            "Relevance": 1
        }
    ],
    "Summary": {
        "BiasPosition": [
            -112.1,
            46.32
        ],
        "DataSource": "Esri",
        "FilterCountries": [
            "USA"
        ],
        "MaxResults": 1,
        "ResultBBox": [
            -112.10667999999998,
            46.319090000000074,
            -112.10667999999998,
            46.319090000000074
        ],
        "Text": "Amazon"
    }
}
```


Example response (HERE)


```
{
    "Summary": {
        "Text": "Amazon",
```

```
        "BiasPosition": [
            -112.1,
            46.32
        ],
        "FilterCountries": [
            "USA"
        ],
        "MaxResults": 1,
        "ResultBBox": [
            -112.10668,
            46.31909,
            -112.10668,
            46.31909
        ],
        "DataSource": "Here"
    },
    "Results": [
        {
            "Place": {
                "Label": "Amazon, Jefferson City, MT, United States",
                "Geometry": {
                    "Point": [
                        -112.10668,
                        46.31909
                    ]
                },
                "Neighborhood": "Amazon",
                "Municipality": "Jefferson City",
                "SubRegion": "Jefferson",
                "Region": "Montana",
                "Country": "USA",
                "Interpolated": false,
                "TimeZone": {
                    "Name": "America/Denver",
                    "Offset": -25200
                }
            },
            "PlaceId": "AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9a
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ",            "Distance":
 523.4619749905755
        }
    ]
```

```
}
```

リバースジオコーディング

ジオコーディングは、住所、地域、会社名、対象地点などのテキストを一連の地理座標に変換するプロセスです。プレイスインデックスリソースを使用してリバースジオコーディングリクエストを送信し、リバースジオコーディングから取得したデータを組み込んで Web またはモバイルアプリケーションのマップ上にデータを表示できます。

このセクションでは、簡単なリバースジオコーディングリクエストを送信する方法について説明します。

リバースジオコーディング

SearchPlaceIndexForPositionオペレーションを使用して、一連の座標をリバースジオコーディングし、わかりやすい住所、対象スポット、または住所のない一般的な場所に変換する簡単なリクエストを送信できます。単純なリクエストには次の必須パラメータが含まれます。

- Position— 住所、対象地点、または大まかな位置に変換したい座標のセット。[longitude,latitude]のフォーマットで定義されます。

1 ページあたりの結果の最大数を指定するには、以下のオプションパラメータを使用します。

- MaxResults— クエリレスポンスで返される結果の数を制限します。

クエリの結果に優先言語を指定する場合は、以下のオプションパラメータを使用します。

- Language— 結果のレンダリングに使用される言語コード。値は有効な BCP 47 言語コードでなければなりません。例えば、英語では en です。

> **Note**
>
> Languageが指定されていない場合、または指定された言語が結果でサポートされていない場合、その結果ではパートナーのデフォルト言語が使用されます。

AWS CLI または Amazon Location APIsを使用できます。

API

次の例は、場所インデックスリソース を検索して*ExamplePlaceIndex*、位置 [*122.3394* , *47.6159*] の近くにある意味のある住所、対象ポイント、または一般的な場所を検索するSearchPlaceIndexForPositionリクエストです。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/position
Content-type: application/json

{
   "Position": [-122.3394,47.6159],
   "MaxResults": 5,
   "Language": "de"
}
```

AWS CLI

次の例は、場所インデックスリソース を検索して*ExamplePlaceIndex*、位置 [*122.3394* , *47.6159*] の近くにある意味のある住所、対象ポイント、または一般的な場所を検索するsearch-place-index-for-positionコマンドです。

```
aws location \
    search-place-index-for-position \
        --index-name ExamplePlaceIndex \
        --position -122.3394 47.6159 \
        --max-results 5 \
        --language de
```

レスポンスの例

Example

以下は、Amazon Location プレイス API からSearchPlaceIndexForPositionオペレーションを呼び出したときのレスポンスの例です。結果には、関連する場所とリクエストの概要が返されます。Esri または Here をパートナーとして選択したことに基づいて、2 つの応答が表示されます。

Example request

```
POST /places/v0/indexes/ExamplePlaceIndex/search/position
Content-type: application/json
```

```
{
   "Position": [-122.3394,47.6159],
   "MaxResults": 1
}
```

Example response (Esri)

```
{
    "Results": [
        {
            "Place": {
                "AddressNumber": "2111",
                "Country": "USA",
                "Geometry": {
                    "Point": [
                        -122.33937999999995,
                        47.61591000000004
                    ]
                },
                "Interpolated": false,
                "Label": "The Spheres, 2111 7th Ave, Seattle, WA, 98121, USA",
                "Municipality": "Seattle",
                "Neighborhood": "Belltown",
                "PostalCode": "98121",
                "Region": "Washington",
                "SubRegion": "King County"
            },
            "Distance": 1.8685861313438727
        }
    ],
    "Summary": {
        "DataSource": "Esri",
        "MaxResults": 1,
        "Position": [
            -122.3394,
            47.6159
        ]
    }
}
```

Example response (HERE)

```
{
```

```
    "Summary": {
        "Position": [
            -122.3394,
            47.6159
        ],
        "MaxResults": 1,
        "DataSource": "Here"
    },
    "Results": [
        {
            "Place": {
                "Label": "2111 7th Ave, Seattle, WA 98121-5114, United States",
                "Geometry": {
                    "Point": [
                        -122.33938,
                        47.61591
                    ]
                },
                "AddressNumber": "2111",
                "Street": "7th Ave",
                "Neighborhood": "Belltown",
                "Municipality": "Seattle",
                "SubRegion": "King",
                "Region": "Washington",
                "Country": "USA",
                "PostalCode": "98121-5114",
                "Interpolated": false,
                "TimeZone": {
                    "Name": "America/Los_Angeles",
                    "Offset": -28800
                }
            },
            "PlaceId": "AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9a
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ",
            "Distance": 1.868586125090601
        }
    ]
}
```

自動入力

オートコンプリートは、エンドユーザーが検索クエリを入力しているときに、応答性の高いフィードバックを提供します。部分的またはスペルミスのある自由形式のテキストに基づいて、住所および対象となるポイントの候補を生成する許可を付与 プレースインデックスリソースを使用してオートコンプリート候補を要求し、その結果の候補をアプリケーションに表示することができます。

Amazon Location では、オートコンプリート候補の保存はサポートされていません。オートコンプリート呼び出しに使用される場所インデックスが、保存されたジオコードで使用するように設定されている場合、エラーが返されます。保存されたジオコードを使用して候補を検索するには、複数のプレイスインデックスを作成して設定します。

このセクションでは、自動入力リクエストを送信する方法について説明します。リクエストの最も基本的な形式から始まり、オートコンプリート検索結果の関連性を高めるために使用できるオプションのパラメータが表示されます。

オートコンプリートの使用

オートコンプリート候補の簡単なリクエストを送信するには、SearchPlaceIndexForSuggestionsオペレーションを使用します。リクエストの最も単純な形式には、クエリTextという必須パラメータが 1 つあります：

- `Text`— 場所の候補を生成するために使用するフリーフォームの部分テキスト。たとえば、文字列`eiffel tow`。

返される結果の数を制限するには、オプションのMaxResultsパラメータを追加します。

- `MaxResults`— クエリレスポンスで返される結果の数を制限します。

Amazon Location API または AWS CLIの使用ができます。

API

次の例は、場所インデックスリソース を検索し*ExamplePlaceIndex*、部分的な場所名 *kamp* に基づいて最大 *5* つの候補を検索するSearchPlaceIndexForSuggestionsリクエストです。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
```

```
    "Text": "kamp",
    "MaxResults": 5
}
```

AWS CLI

次の例は、場所インデックスリソース を検索し*ExamplePlaceIndex*、部分的な場所名 *kamp* に基づいて最大 *5* つの候補を検索するsearch-place-index-for-suggestionsコマンドです。

```
aws location \
              search-place-index-for-suggestions \
              --index-name ExamplePlaceIndex \
              --text kamp \
              --max-results 5
```

SearchPlaceIndexForSuggestionsを呼び出すと、それぞれの場所の名前と ID を含む場所のリストが生成されます。これらの結果を使用して、テキストボックスの下に選択肢のドロップダウンリストの表示や、ユーザーが入力するときに検索内容の候補の表示などができます。例えば、ユーザーが *kamp* と入力した場合のサジェスト結果は以下の通りです。

```
{
    "Summary": {
        "Text": "kamp",
        "MaxResults": 5,
        "DataSource": "Esri"
    },
    "Results": [
        {
            "Text": "Kampuchea",
            "PlaceId": "AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9ajT3
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ"
        },
        {
            "Text": "Kampoul, Kabul, AFG",
            "PlaceId":
 "AQAAAIAAA1mxl_-9ffzXD07rBgo9fh6E01Pd1YKvuT5rz2qBDxqBkhTlgkeiOPR2s5sa3YBLxUqQI8bhymsYcu9R-
DkX3L9QSi3CB5LhNPu160iSFJo6H8S1CrxO3QsJALhrr9mdbg0R4R4YDywkhkeBlnbn7g5C5LI_wYx873WeQZuilwtsGm8j
UeXcb_bg"
        },
```

```
        {
            "Text": "Kampala, UGA",
            "PlaceId":
 "AQAAAIAAzZfZt3qMruKGObyhP6MM0pqy2L8SULlVWT7a3ertLBRS6Q5n7I4s9D7E0nRHADAj7mL7kvX1Q8HD-
mpuiATXNJ1Ix4_V_1B15zHe8jlYKMWvXbgbO8cMpgR2fqYqZMR1x-
dfBOO8OoqujKZldvPIDK1kNe3GwcaqvvMWWPMeaGd203brFynubAe-MmFF-Gjz-WBMfUy9og6MV7bkk6NGCA"
        },
        {
            "Text": "Kampar, Riau, IDN",
            "PlaceId": "AQAAAIAAvbXXx-
srOi111tHOkPdao0GF7WQ_KaZ444SEnevycp6Gtf_2JWgPfCE5bIQCYwya1uZQpX2a8YJoFm2K7Col4fLu7IK0yYOLhZx4k
        },
        {
            "Text": "Kampung Pasir Gudang Baru, Johor, MYS",
            "PlaceId":
 "AQAAAIAA4HLQHdjUDcaaXLE9wtNIT1cjQYLgkBnMoG2eNN0AaQ8PJoWabLRXmmPUaAj8MAD6vT0i6zqaun5Mixyj7vnYX
        }
    ]
}
```


次のセクションでは、これらの結果から導き出された PlaceID の使い方を説明します。

自動入力結果の使用

SearchPlaceIndexForSuggestionsを呼び出すと、それぞれの場所の名前と ID を含む場所のリストが生成されます。これらの結果を使用して、テキストボックスの下に選択肢のドロップダウンリストの表示や、ユーザーが入力するときに検索内容の候補の表示などができます。ユーザーがいずれかの結果を選択したら、選択した ID で GetPlaceオペレーションを呼び出して、場所、住所、その他の詳細など、その場所の詳細を返します。

> **Note**
>
> PlaceId は、元の検索リクエストとGetPlaceへの呼び出しにおいて、以下のすべてが同じ場合にのみ有効です。
>
> - お客様 AWS アカウント
> - AWS リージョン
> - 場所インデックスリソースで指定されているデータプロバイダー

通常、 GetPlace は Amazon Location API と一緒に使います。次の例は、前のセクションの提案の 1 つを見つけるためのGetPlaceリクエストです。この例は *kamp* という地名の一部に基づいています。

```
POST /places/v0/indexes/ExamplePlaceIndex/
places/AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9ajT3
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ
```

ある位置の近くでオートコンプリート

SearchPlaceIndexForSuggestionsを使用してオートコンプリート候補を検索する場合、次のオプションパラメータを追加すると、より地域に適した候補が表示されます。

- BiasPosition— 付近で検索したい位置。[longitude, latitude]と定義されています。

次の例では、 SearchPlaceIndexForSuggestionsリクエストを使用して場所インデックスリソースを検索し、位置 [*32.5827* , *0.3169*] の近くにある部分クエリ *kamp* に一致する場所候補*ExamplePlaceIndex*を探します。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
    "Text": "kamp",
    "BiasPosition": [32.5827,0.3169]
}
```

-96.7977, *32.7776*のように、異なるものが選択された場合、同じ Text に対して返されるサジェストは異なる BiasPosition になる可能性があります。

バウンディングボックス内の自動入力

次のオプションパラメータを追加することで、オートコンプリート検索を絞り込んで、特定の境界内にある場所の候補のみが表示されるようにすることができます。

- FilterBBox— 検索結果をバウンディングボックス内の座標に絞り込むために指定するバウンディングボックス。[LongitudeSW, LatitudeSW, LongitudeNE, LatitudeNE]と定義されます。

> **Note**
>
> リクエストはFilterBBoxとBiasPositionの両方のパラメーターを含むことはできません。リクエストで両方のパラメータを指定するとValidationExceptionエラーが返されます。

次の例では、 SearchPlaceIndexForSuggestionsリクエストを使用して場所インデックスリソースを検索し、部分クエリ *kamp* に一致する場所候補と、境界ボックスに含まれる場所候補*ExamplePlaceIndex*を検索します。

- バウンディングボックスの南西の角の経度は *32.5020* です。
- バウンディングボックスの南西の角の緯度は *0.2678* です。
- バウンディングボックスの北東の角の経度は *32.6129* です。
- バウンディングボックスの北東の角の緯度は *0.3502* です。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
    "Text": "kamp",
    "FilterBBox": [
        32.5020, 0.2678,
        32.6129, 0.3502
    ]
}
```

-97.9651, *32.0640*, *-95.1196*, *34.0436* のように、異なるものが選択された場合、同じ Text に対して返されるサジェストは異なる FilterBBox になる可能性があります。

国内でのオートコンプリート

次のオプションパラメータを追加することで、オートコンプリート検索を絞り込んで、特定の境界内にある場所の候補のみが表示されるようにすることができます。

- FilterCountries— 検索したい国の候補が表示されます。ISO 3166 の 3 文字の国コードを使用すると、1 回のリクエストで最大 100 か国を指定することができます。たとえば、オーストラリアではAUSです。

次の例では、 SearchPlaceIndexForSuggestionsリクエストを使用して場所インデックスリソースを検索し、部分クエリ *kamp* に一致し、ウンダ、ケニア、またはタガニアに含まれる場所候補*ExamplePlaceIndex*を探します。

```
POST /places/v0/indexes/ExamplePlaceIndex/search/suggestions
Content-type: application/json

{
    "Text": "kamp",
    "FilterCountries": ["UGA", "KEN", "TZA"]
}
```

同じTextに対して返される候補は、*USA*のように異なるリストが選択された場合、異なるFilterCountriesとなる。

レスポンスの例

以下は、「*kamp*」というテキストを使用したSearchPlaceIndexForSuggestionsオペレーションのオートコンプリート候補のレスポンス例です。

```
{
    "Summary": {
        "Text": "kamp",
        "MaxResults": 5,
        "DataSource": "Esri"
    },
    "Results": [
        {
            "Text": "Kampuchea",
            "PlaceId": "AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9ajT3
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ"
        },
        {
            "Text": "Kampoul, Kabul, AFG",
            "PlaceId":
 "AQAAAIAAA1mxl_-9ffzXD07rBgo9fh6E01Pd1YKvuT5rz2qBDxqBkhTlgkeiOPR2s5sa3YBLxUqQI8bhymsYcu9R-
```

```
DkX3L9QSi3CB5LhNPu160iSFJo6H8S1CrxO3QsJALhrr9mdbg0R4R4YDywkhkeBlnbn7g5C5LI_wYx873WeQZuilwtsGm8j
UeXcb_bg"
        },
        {
            "Text": "Kampala, UGA",
            "PlaceId":
 "AQAAAIAAzZfZt3qMruKGObyhP6MM0pqy2L8SULlVWT7a3ertLBRS6Q5n7I4s9D7E0nRHADAj7mL7kvX1Q8HD-
mpuiATXNJ1Ix4_V_1B15zHe8jlYKMWvXbgbO8cMpgR2fqYqZMR1x-
dfBOO8OoqujKZldvPIDK1kNe3GwcaqvvMWWPMeaGd203brFynubAe-MmFF-Gjz-WBMfUy9og6MV7bkk6NGCA"
        },
        {
            "Text": "Kampar, Riau, IDN",
            "PlaceId": "AQAAAIAAvbXXx-
srOi111tHOkPdao0GF7WQ_KaZ444SEnevycp6Gtf_2JWgPfCE5bIQCYwya1uZQpX2a8YJoFm2K7Col4fLu7IK0yYOLhZx4k
        },
        {
            "Text": "Kampung Pasir Gudang Baru, Johor, MYS",
            "PlaceId":
 "AQAAAIAA4HLQHdjUDcaaXLE9wtNIT1cjQYLgkBnMoG2eNN0AaQ8PJoWabLRXmmPUaAj8MAD6vT0i6zqaun5Mixyj7vnYX
        }
    ]
}
```


プレイス ID の使用

場所を検索すると、結果のリストが返されます。ほとんどの結果には、その結果のPlaceIdが含まれています。GetPlaceオペレーションで PlaceId を使用すると、その場所に関する情報 (名前、住所、場所、その他の詳細を含む) が返されます。

> **Note**
>
> を使用するとSearchPlaceIndexForSuggestions、任意のデータソースで作成された任意の場所インデックスPlaceIdの結果が返されます。SearchPlaceIndexForText または を使用すると、使用するデータソースが HERE であるPlaceId場合にのみ が返されSearchPlaceIndexForPositionます。

それぞれの PlaceId は、それが指す場所を一意に定義するが、一つの場所は、時間の経過とともに、また文脈に応じて、複数の PlaceId が含まれる場合があります。以下のルールは、 PlaceId のユニークさと耐久性を説明するものである。

- 実行する呼び出しでPlaceId返される は AWS アカウント、、 AWS リージョン、およびPlaceIndexリソース内のデータプロバイダーに固有です。 GetPlaceは、これら 3 つの属性が を作成した元の呼び出しと一致した場合にのみ結果を検索しますPlaceId。
- PlaceIdという場所に関するデータが変わると、その場所の番号も変わります。例えば、参照するビジネスが所在地を移転したり、名称を変更した場合などです。
- 繰り返される検索の呼び出しから返されるPlaceIdは、バックエンド・サービスが更新を行ったときに変更される可能性があります。元の PlaceId は引き続き検索されるが、新たに検索を呼び出すと別の ID が返されることがあります。

PlaceIdは文字列です。 PlaceId の長さに特に制限されていません。次は、PlaceId の例です。

```
AQAAAIAADsn2T3KdrRWeaXLeVEyjNx_JfeTsMB0NVCEAnAZoJ-
o3nqdlJZAdgcT2oWi1w9pS4wXXOk3O1vsKlGsPyHjV4EJxsu289i3hVO_BUPgP7SFoWAi8BW2v7LvAjQ5NfUPy7a1v9ajT3
et39ZQDWSPLZUzgcjN-6VD2gyKkH0Po7gSm8YSJNSQ
```

データが変更された場所(例えば、廃業した事業所など)のために PlaceId で GetPlace を呼び出すと、404,ResourceNotFound エラーが発生します。有効でない 、または別の からの など、コンテキスト外の で PlaceIdを呼び出すGetPlaceと AWS アカウント、400ValidationExceptionエラー が返されます。

後続のリクエストで PlaceID を使用できますが、 PlaceID は永続的な識別子ではなく、ID は連続する API コール間で変更される可能性があります。データプロバイダーごとに、次の PlaceID の動作を参照してください。

- Esri : プレイス IDsは少なくとも四半期ごとに変更されます。これらの変更の一般的な期間は、3 月、6 月、9 月、12 月です。プレイス IDs通常の四半期ごとの変更間で変更されることもありますが、頻度ははるかに低くなります。
- HERE : データを最新の状態に保つため、1 週間以内はデータをキャッシュすることをお勧めします。ID シフトが 1% 未満になると、リリース時にリリースされると想定できます。これは、1 週間に約 1 ~ 2 回です。
- Grab: 次の状況ではIDs の有効期限が切れたり無効になったりする可能性があります。
  - データオペレーション: POI は、現実世界のクローズ、重複した POI として検出された、または誤った情報を持つなど、グラウンドトゥルースに基づいて Grab Map Ops によって Grab POI データベースから削除される可能性があります。Grab は毎週データを Waypoint 環境に同期します。

- 補間 POI : 補間 POI は、リクエストを処理するときにリアルタイムで生成される一時的な POI であり、レスポンスの `place.result_type`フィールドで派生としてマークされます。補間された POIs の情報は少なくとも 30 日間保持されます。つまり、30 日以内に、プレイスの詳細 API からプレイス ID で POI の詳細を取得できます。30 日後、補間された POIs (場所 ID と詳細の両方) の有効期限が切れ、場所の詳細 API からアクセスできなくなる場合があります。

カテゴリーの配置と結果のフィルタリング

場所は分類されています。たとえば、商業施設を検索すると、その商業施設は `Restaurant` である可能性があります。所在地を検索した結果でも、所在地、道路、交差点と一致したかどうかで分類できます。

Amazon Location Service、おおまかに場所をプレスタイプに分類します。興味のある地点はさらに「興味のあるポイントタイプ」に分類されます。

> **Note**
>
> すべての結果にカテゴリが含まれるわけではありません。

カテゴリを使用してジオコーディング検索を絞り込むことができます。

結果のフィルタ処理

`SearchPlaceIndexForText`を利用している場合は、使いたいカテゴリによって返される結果をフィルタリングすることができます。例:

- 「Hometown Coffee」という場所を検索して、コーヒーショップとして分類された結果のみを返したい場合は、`SearchPlaceIndexForText`とポイントオブインタレストを呼び出して、`Coffee Shop` のカテゴリを`FilterCategories`パラメータに含めてください。
- 「123 Main St, Anytown, WA, 98123, USA」を検索する場合、結果を住所だけに絞り込むことができるため、たとえば郵便番号に一致するものは見つかりません。`FilterCategories`のパラメータにプレイスタイプ、`AddressType`を含めることで、住所のみを絞り込むことができます。

> **Note**
>
> すべてのデータプロバイダーがフィルターをサポートしているわけではなく、また同じ方法でサポートしているわけでもありません。詳細については、「データプロバイダーによるフィルタリングの制限」を参照してください。

次のセクションでは、フィルタリングできるカテゴリーをリストアップします。

カテゴリ

次のリストは、Amazon Location Service が分類とフィルタリングに使用するカテゴリを示しています。これらのカテゴリは、言語パラメータが異なる言語に設定されているかどうかに関係なく、すべての言語で使用されます。

> **Note**
>
> Amazon Location Serviceは、データプロバイダのカテゴリをこのカテゴリのセットにマッピングしています。データプロバイダが Amazon Location Service カテゴリリストに含まれていないカテゴリに場所を入力した場合、プロバイダカテゴリは補助カテゴリとして結果に含まれます。

プレイスタイプ — これらのタイプは、結果を見つけるために使用された一致のタイプを示すために使用されます。

- `AddressType`— 結果が住所と一致したときに返されます。
- `StreetType`— 結果が道路と一致した場合に返されます。
- `IntersectionType`— 結果が 2 つの道路の交差点と一致した場合に返されます。
- `PointOfInterestType`— 結果がビジネスや公共の場所など、興味のある場所と一致した場合に返されます。
- `CountryType`— 結果が国または主要地域と一致した場合に返されます。
- `RegionType`— 結果が州や都道府県などの国内の地域と一致した場合に返されます。
- `SubRegionType`— 結果が国内の小地域（郡や大都市圏など）と一致した場合に返されます。
- `MunicipalityType`— 結果が道路と一致した場合に返されます。
- `NeighborhoodType`— 結果が都市内の近所または地域と一致した場合に返されます。

- PostalCodeType— 結果が郵便番号と一致した場合に返されます。

ポイント・オブ・インタレスト・カテゴリ — これらのカテゴリは、ビジネスや場所の種類を示すために使用されます。

- Airport
- Amusement Park
- Aquarium
- Art Gallery
- ATM
- Bakery
- Bank
- Bar
- Beauty Salon
- Bus Station
- Car Dealer
- Car Rental
- Car Repair
- Car Wash
- Cemetery
- Cinema
- City Hall
- Clothing Store
- Coffee Shop
- Consumer Electronics Store
- Convenience Store
- Court House
- Dentist
- Embassy
- Fire Station
- Fitness Center

- Gas Station
- Government Office
- Grocery
- Higher Education
- Hospital
- Hotel
- Laundry
- Library
- Liquor Store
- Lodging
- Market
- Medical Clinic
- Motel
- Museum
- Nightlife
- Nursing Home
- Park
- Parking
- Pet Store
- Pharmacy
- Plumbing
- Police Station
- Post Office
- Religious Place
- Restaurant
- School
- Shopping Mall
- Sports Center
- Storage

- Taxi Stand
- Tourist Attraction
- Train Station
- Veterinary Care
- Zoo

データプロバイダーによるフィルタリングの制限

すべてのプロバイダーが同じフィルタリング機能を備えっているわけではありません。以下の表では、階層の違いについて説明しています。

プロバイダー	フィルターをサポートする API	フィルタリングに対応するカテゴリー	戻り値
Esri	SearchPlaceIndexForText , SearchPlaceIndexForSuggestions	プレイスタイプとポイントオブインタレストカテゴリでフィルタリングします。	カテゴリーはSearchPlaceIndexForText 、GetPlace、SearchPlaceIndexForPosition で返されます。
こちら	SearchPlaceIndexForText , SearchPlaceIndexForSuggestions	プレイスタイプのみで絞り込みます。	カテゴリーはSearchPlaceIndexForText 、SearchPlaceIndexForSuggestions 、SearchPlaceIndexForPosition 、GetPlaceで返されます。
grab	サポート外	サポート外	サポート外

プロバイダー	フィルターをサポートする API	フィルタリングに対応するカテゴリー	戻り値
オープンデータ	n/a (場所の検索はサポートされていません)	該当なし	該当なし

Amazon Aurora PostgreSQL ユーザー-Amazon Location Service 義関数

Amazon Location Service を使用すると、データベーステーブルに保存されている座標と住所を操作して、地理空間データを整理して充実させることができます。

例:

- ジオコーディングを使用して住所を座標に変換し、データベーステーブルに格納されている住所のデータを正規化して補うことができます。
- 住所をジオコーディングして位置を取得し、その座標をデータベースの空間関数 (指定した地域の行を表示する関数など) での使用ができます。
- 特定の地域のすべてのデバイスを示す自動レポートや、位置情報の更新時に失敗率が高い地域を示す機械学習用の自動レポートを生成するなど、豊富なデータを使用して自動レポートを生成できます。

このチュートリアルでは、Amazon Location Service を使って、 Amazon Aurora PostgreSQL データベースのテーブルに保存されている住所をフォーマットし、リッチ化する方法を紹介します。

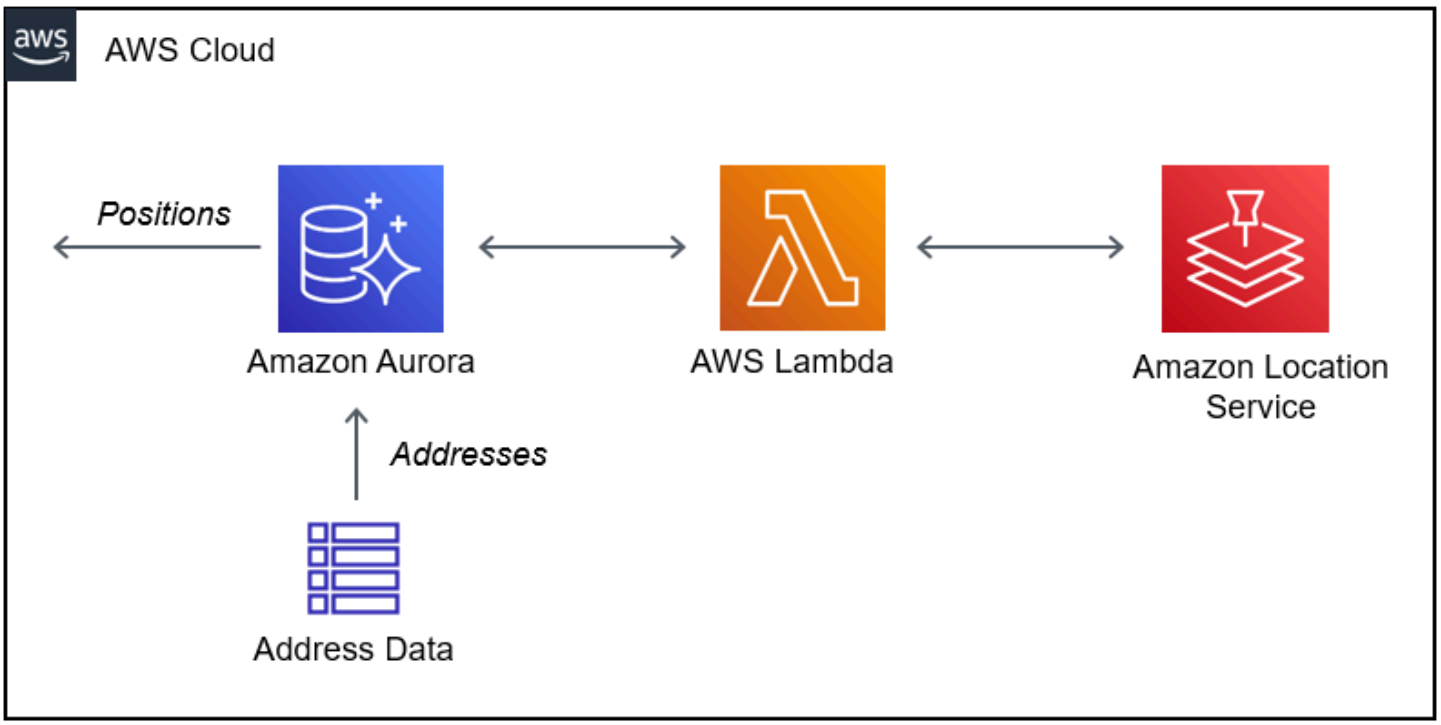
- Amazon Aurora PostgreSQL— MySQL と PostgreSQL と互換性のあるフルマネージドのリレーショナルデータベースエンジンで、既存のアプリケーションのほとんどを変更することなく、MySQL と PostgreSQL のスループットの 5 倍、PostgreSQL のスループットの 3 倍を実現します。詳細については、「Amazon Aurora ユーザーガイド」の「Amazon Aurora とは」を参照してください。

> **Important**
>
> このチュートリアルで作成されたアプリケーションでは、ジオコーディング結果を格納する場所インデックスを使用します。ジオコーディング結果の保存に適用される料金については、「Amazon Location Service 料金表」を参照してください。

サンプルコードはGitHub、 AWS CloudFormation テンプレートを含む の Amazon Location Service サンプルリポジトリにあります。

トピック

概要



このアーキテクチャには以下の統合が含まれます。

- このソリューションでは、Amazon Location Place インデックスリソースを使用して、`SearchPlaceIndexForText`オペレーションを使用したジオコーディングクエリをサポートします。
- AWS Lambdaは、IAM ポリシーが、AWS Lambda が Amazon Location のジオコーディングオペレーション `SearchPlaceIndexForText` を呼び出すことを許可したときに、住所をジオコーディングする Python Lambda を使用します。
- SQL ユーザー定義関数を使用してジオコーディング Lambda 関数を呼び出す権限をAmazon Aurora PostgreSQLに付与します。

前提条件

開始する前に、以下の前提条件を満たしている必要があります。

- クラスター。Amazon Aurora PostgreSQL詳細については、「Amazon Aurora ユーザーガイド」の「Amazon Aurora DB クラスターの暗号化」を参照してください。

> **Note**
>
> Amazon Aurora クラスターがパブリックアクセス可能でない場合は、AWS アカウントの仮想プライベートクラウド (VPC)内のAWS Lambda に接続するように Amazon Aurora を設定する必要があります。詳細については、「Amazon Aurora PostgreSQLアクセス権を付与する対象AWS Lambda」を参照してください。

- クラスターに接続するための SQL 開発者ツール。Amazon Aurora PostgreSQL

クイックスタート

このチュートリアルのステップを実行する代わりに、クイックスタックを起動して Amazon Location オペレーション`SearchPlaceIndexForText`をサポートするAWS Lambda関数をデプロイすることもできます。これにより、Amazon Aurora からAWS Lambdaの呼び出しを許可するように AWS アカウントが自動的に設定されます。

AWSアカウントを設定したら、次のことを行う必要があります。

- Amazon Aurora にLambda 機能を追加します。Amazon Aurora PostgreSQLアクセス権を付与する対象AWS Lambdaの「IAM ロールを Amazon Aurora DB クラスターに追加する」を参照してください。
- ユーザー定義関数をデータベースにロードします。AWS Lambda 関数を呼び出す を参照してください。

Launch Stack

場所インデックスリソースの作成

まず、ジオコーディングクエリをサポートするプレイスインデックスリソースを作成します。

1. Amazon Location Service コンソール (https://console.aws.amazon.com/location/) を開きます。
2. 左のナビゲーションペインで、プレース・インデックス を選択します。
3. 次のボックスに入力します。

- 名前 — プレイスインデックスリソースの名前を入力します。例えば、 ですAuroraPlaceIndex。最大 100 文字。 には、英数字、ピリオド (.)、ハイフン (-)、アンダースコア (_) を含めることができます。
- 説明 – 任意の説明を入力します。たとえば、Amazon Aurora ##########などです。

4. 「データプロバイダー」で、プレイスインデックスリソースで使用できるデータプロバイダーを選択します。特にご希望がなければ、Esri から始めることをお勧めします。
5. データストレージオプションで はい、結果を保存しますを指定します。これは、ジオコーディング結果をデータベースに保存するつもりであることを示しています。
6. （オプショナル）タグに、タグのキーと値を入力します。これにより、新しいプレイスインデックスリソースにタグが追加されます。詳細については、リソースにタグを付けるを参照してください。
7. インデックスの作成 を選択します。

ジオコーディング用のAWS Lambda関数を作成します。

Amazon Aurora PostgreSQLと Amazon Location Service との接続を作成するには、データベースエンジンからのリクエストを処理するAWS Lambda関数が必要です。この関数は Lambda ユーザー定義関数イベントを変換し、Amazon Location オペレーションSearchPlaceIndexForTextを呼び出します。

この関数は、AWS Lambdaコンソール、AWS Command Line Interface、または AWS Lambda API を使用して作成できます。

コンソールを使ってLambdaのユーザー定義関数を作成するには

1. AWS Lambda コンソールを https://console.aws.amazon.com/lambda/ で開きます。
2. ナビゲーションペインで、関数 を選択します。
3. 関数を作成を選択し、最初から作成するが選択されていることを確認します。
4. 次のボックスに入力します。
   - 関数の名前 に、 関数の名前を入力します。有効なエントリには、英数字、ハイフン、およびスペースなしのアンダースコアが含まれます。例えば、 などですAuroraGeocoder。
   - ランタイム:Python 3.8を選択します。
5. 機能の作成を選択します。

6. JSON タブを選択して JSON エディタを開きます。
7. lambda_function.pyのプレースホルダーコードを次のように上書きします。

```
from os import environ

import boto3
from botocore.config import Config

# load the place index name from the environment, falling back to a default
PLACE_INDEX_NAME = environ.get("PLACE_INDEX_NAME", "AuroraPlaceIndex")

location = boto3.client("location", config=Config(user_agent="Amazon Aurora
 PostgreSQL"))

"""
This Lambda function receives a payload from Amazon Aurora and translates it to
an Amazon Location `SearchPlaceIndex` call and returns the results as-is, to be
post-processed by a PL/pgSQL function.
"""
def lambda_handler(event, context):
    kwargs = {}

    if event.get("biasPosition") is not None:
        kwargs["BiasPosition"] = event["biasPosition"]

    if event.get("filterBBox") is not None:
        kwargs["FilterBBox"] = event["filterBBox"]

    if event.get("filterCountries") is not None:
        kwargs["FilterCountries"] = event["filterCountries"]

    if event.get("maxResults") is not None:
        kwargs["MaxResults"] = event["maxResults"]

    return location.search_place_index_for_text(
        IndexName=PLACE_INDEX_NAME,
        Text=event["text"],
        **kwargs)["Results"]
```

8. 場所インデックスに 以外の名前を付けた場合は*AuroraPlaceIndex*、リソース名を割り当てる PLACE_INDEX_NAME という名前の環境変数を作成します。
   - 設定 を選択してから、環境変数 を選択します。

- 編集を選択し、環境変数を追加を選択します。
- キー にPLACE_INDEX_NAMEと入力します。
- Valueには、プレイスインデックスリソースの名前を入力します。

9. デプロイ を選択して、関数を更新します。
10. テスト ボタンの横にあるドロップダウンメニューから テストイベントの設定 を選択します。
11. Create new test eventを選択します。
12. 次のテストイベントを入力します。

```
{
  "text": "Baker Beach",
  "biasPosition": [-122.483, 37.790],
  "filterCountries": ["USA"]
}
```

13. テスト を選択して Lambda 関数をテストします。
14. 設定タブを選択します。
15. 一般設定で 権限を選択します。
16. 実行ロールで、ハイパーリンクされたロール名を選択し、Amazon Location Service に Lambda 関数へのアクセス権限を付与します。
17. 許可 タブで、許可の追加 、インラインポリシーの作成 をクリックします。
18. JSONタブを選択します。
19. 次の IAM ポリシーは、以下のことを行います。

- 次のポリシーは、場所インデックスリソース SearchPlaceIndexForTextに送信するアクセス許可を付与します*AuroraPlaceIndex*。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "geo:SearchPlaceIndexForText",
      "Resource": "arn:aws:geo:<Region>:<AccountId>:place-index/AuroraPlaceIndex"
    }
  ]
}
```

20. ポリシーの確認を選択します。
21. ポリシー名を入力します。例えば、 です*AuroraPlaceIndexReadOnly*。
22. ポリシーを作成を選択します。

Amazon Aurora PostgreSQLアクセス権を付与する対象AWS Lambda

Amazon Aurora PostgreSQLがAWS Lambda関数を呼び出す前に、アクセス権限を付与する必要があります。

Amazon Aurora PostgreSQLクラスターがパブリックにアクセスできない場合、Amazon Aurora が Lambda 関数を呼び出すには、まずAWS Lambdaの VPC エンドポイントを作成する必要があります。

AWS Lambdaの VPC エンドポイントの作成

> **Note**
>
> この手順は、Amazon Aurora PostgreSQLクラスターがパブリックアクセス可能でない場合にのみ必要です。

1. Amazon Virtual Private Cloud Consoleを開きます。
2. 左のナビゲーションペインで エンドポイント を選択します。
3. エンドポイントの作成 を選択します。
4. サービス名フィルターに「lambda」と入力し、`com.amazonaws.<region>.lambda`を選択します。
5. Aurora クラスターが含まれる VPC を選択します。
6. アベイラビリティーゾーンごとに、サブネットを 1 つ選択します。
7. セキュリティグループフィルターで、「default」または Aurora クラスターが所属するセキュリティグループの名前を入力し、セキュリティグループを選択します。
8. エンドポイントの作成 を選択します。

AWS Lambda関数への呼び出しを許可する IAM ポリシーを作成する

1. IAM コンソールを開きます。

2. ナビゲーションペインの アクセス管理 を展開し、ポリシー を選択します。
3. ポリシーを作成を選択します。
4. JSON タブに、以下のポリシーを貼り付けます。

   - 以下は、AuroraGeocoder AWS Lambda関数の呼び出しAmazon Aurora PostgreSQL 許可を与えるIAMポリシーの例です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": [
                "arn:aws:lambda:<Region>:<AccountId>:function:AuroraGeocoder"
            ]
        }
    ]
}
```

5. 次へ: タグ を選択し、オプショナルタグを追加します。
6. 次へ: レビュー を選択します。
7. ポリシーを確認して、ポリシーに関する以下の詳細情報を入力します。

   - 名前 — 英数字と「+=, .@-_」文字を使用してください。最大 128 文字 例えば、 などで すAuroraGeocoderInvoke。
   - 説明 – 任意の説明を入力します。英数字と 「+=, .@-_」 文字を使用してください。最大 1000 文字
8. ポリシーを作成を選択します。ポリシーを IAM ロールにアタッチする場合に使用する、このポリシーの ARN を書きます。

IAM ロールを作成して Amazon Relational Database Service (Amazon RDS) にアクセス許可を付与します

IAM ロールを作成することで、Amazon Aurora PostgreSQLはユーザーに代わって Lambda 関数にアクセスするための、ロールを、引き受けることができます。詳細については、IAM ユーザーガイドの「IAM ユーザーにアクセス許可を委任するロールの作成」を参照してください。

次の例は、 という名前のロールを作成する AWS CLI コマンドで
す*AuroraGeocoderInvokeRole*。

```
aws iam create-role  --role-name rds-lambda-role --assume-role-policy-document '{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Principal": {
            "Service": "rds.amazonaws.com"
        },
        "Action": "sts:AssumeRole"
        }
    ]
}'
```

IAM ポリシーを IAM ロールにアタッチします。

IAM ロールがある場合は、作成した IAM ポリシーをアタッチします。

次の例は、ポリシー をロール *AuroraGeocoderInvoke*にアタッチする AWS CLI コマンドで
す*AuroraGeocoderInvokeRole*。

```
aws iam attach-role-policy  --policy-arn AuroraGeocoderInvoke  --role-
name AuroraGeocoderInvokeRole
```

IAM ロールを Amazon Aurora DB クラスターに追加します。

次の例は、 という名前の Amazon Aurora PostgreSQL DB クラスターに IAM ロールを追加する
AWS CLI コマンドです*MyAuroraCluster*。

```
aws rds add-role-to-db-cluster \
--db-cluster-identifier MyAuroraCluster \
--feature-name Lambda \
--role-arn AuroraGeocoderInvokeRole   \
--region your-region
```

AWS Lambda 関数を呼び出す

ジオコーディング Lambda 関数を呼び出す許可を Amazon Aurora PostgreSQL に与えた後、ジ
オコーディング関数を呼び出すユーザー定義AWS Lambda関数をAmazon Aurora PostgreSQLに

作成することができます。詳細については、Amazon AuroraユーザーガイドのAmazon Aurora PostgreSQLDBクラスタからAWS Lambda関数を呼び出すを参照してください。

必要な PostgreSQL 拡張機能をインストールします。

必要な PostgreSQL aws_lambda と aws _commons 拡張と拡張機能をインストールするには、Amazon Aurora ユーザーガイドの「Lambda 関数の使用の概要」を参照してください。

```
CREATE EXTENSION IF NOT EXISTS aws_lambda CASCADE;
```

必要な PostGIS 拡張機能をインストールします。

PostGIS は PostgreSQL のエクステンションであり、空間情報の保存と管理に使用します。詳細については、『Amazon リレーショナルデータベースサービス ユーザーガイド』の「PostGIS エクステンションの使用」を参照してください。

```
CREATE EXTENSION IF NOT EXISTS postgis;
```

Lambda 関数を呼び出す SQL ユーザー定義関数を作成する

SQL エディタで、新しいユーザー定義関数 を作成して、関数 をf_SearchPlaceIndexForText呼び出します*AuroraGeocoder*。

```
CREATE OR REPLACE FUNCTION f_SearchPlaceIndexForText(
  text text,
  bias_position geometry(Point, 4326) DEFAULT NULL,
  filter_bbox box2d DEFAULT NULL,
  filter_countries text[] DEFAULT NULL,
  max_results int DEFAULT 1
)
 RETURNS TABLE (
   label text,
   address_number text,
   street text,
   municipality text,
   postal_code text,
   sub_region text,
   region text,
   country text,
   geom geometry(Point, 4326)
 )
 LANGUAGE plpgsql
```

```
 IMMUTABLE
AS $function$
begin
    RETURN QUERY
    WITH results AS (
      SELECT json_array_elements(payload) rsp
      FROM aws_lambda.invoke(
        aws_commons.create_lambda_function_arn('AuroraGeocoder'),
        json_build_object(
          'text', text,
          'biasPosition',
          CASE WHEN bias_position IS NOT NULL THEN
            array_to_json(ARRAY[ST_X(bias_position), ST_Y(bias_position)])
          END,
          'filterBBox',
          CASE WHEN filter_bbox IS NOT NULL THEN
            array_to_json(ARRAY[ST_XMin(filter_bbox), ST_YMin(filter_bbox),
 ST_XMax(filter_bbox), ST_YMax(filter_bbox)])
          END,
          'filterCountries', filter_countries,
          'maxResults', max_results
        )
      )
    )
    SELECT
      rsp->'Place'->>'Label' AS label,
      rsp->'Place'->>'AddressNumber' AS address_number,
      rsp->'Place'->>'Street' AS street,
      rsp->'Place'->>'Municipality' AS municipality,
      rsp->'Place'->>'PostalCode' AS postal_code,
      rsp->'Place'->>'SubRegion' AS sub_region,
      rsp->'Place'->>'Region' AS region,
      rsp->'Place'->>'Country' AS country,
      ST_GeomFromGeoJSON(
        json_build_object(
          'type', 'Point',
            'coordinates', rsp->'Place'->'Geometry'->'Point'
        )
      ) geom
    FROM results;
end;
$function$;
```

SQL 関数を呼び出して Aurora からジオコーディングする

SQL ステートメントを実行すると*AuroraGeocoder*、Lambda 関数 が呼び出されます。この関数は、Amazon Aurora PostgreSQLデータベース内のデータベーステーブルからアドレスレコードを取得し、場所インデックスリソースを使用してそれらをジオコーディングします。

> **Note**
>
> Amazon Aurora PostgreSQLはSQL ユーザー定義関数を呼び出すたびに Lambda 関数を呼び出します。
> 50 行をジオコーディングする場合、Amazon Aurora PostgreSQLはLambda 関数を 50 回呼び出します。各行に対して 1 回の呼び出し。

次の `f_SearchPlaceIndexForText` SQL 関数は、*AuroraGeocoder*Lambda 関数を介して Amazon Location の `SearchPlaceIndexForText` API にリクエストを送信します。この関数は、PostGISジオメトリである `geom` 列を返し、`ST_AsText(geom)`はそれをテキストに変換します。

```
SELECT *, ST_AsText(geom)
FROM f_SearchPlaceIndexForText('Vancouver, BC');
```

デフォルトでは、戻り値には 1 行が含まれます。`MaxResults`制限までの追加行を要求するには、`BiasPosition` を指定し、カナダでの結果に制限しながら、以下のSQL文を実行する。

```
SELECT *
FROM f_SearchPlaceIndexForText('Mount Pleasant', ST_MakePoint(-123.113, 49.260), null,
 '{"CAN"}', 5);
```

バウンディングボックスを使用して結果をフィルタリングするには、`filter_bbox`として`Box2D`を渡します。

- `FilterBBox`— バウンディングボックス内の場所を返すことで結果をフィルタリングします。このパラメータはオプションです。

```
SELECT *
FROM f_SearchPlaceIndexForText('Mount Pleasant', null, 'BOX(-139.06 48.30, -114.03
 60.00)'::box2d, '{"CAN"}', 5);
```

PostGIS のタイプと機能の詳細については、『PostGIS リファレンス』を参照してください。

住所データを含むデータベースの強化

Amazon Location オペレーション SearchPlaceIndexForText を使って、フォーマットされた住所をコンストラクトし、同時に正規化とジオコーディングを行うことができます：

- id
- address
- city
- state
- zip

```
WITH source_data AS (
  SELECT
    id,
    address || ', ' || city || ', ' || state || ', ' || zip AS formatted_address
  FROM addresses
),
geocoded_data AS (
  SELECT
    *,
    (f_SearchPlaceIndexForText(formatted_address)).*
  FROM source_data
)
SELECT
  id,
  formatted_address,
  label normalized_address,
  ST_Y(geom) latitude,
  ST_X(geom) longitude
FROM geocoded_data
-- limit the number of rows that will be geocoded; remove this to geocode the entire
 table
LIMIT 1;
```

次の例では、結果の 1 つのデータテーブル行を示しています。

```
 id |      formatted_address         |           normalized_address              |
 latitude     |     longitude
```

```
----+--------------------------------+-------------------------------------------
+------------------+-------------------
 42 | 123 Anytown Ave N, Seattle, WA | 123 Anytown Ave N, Seattle, WA, 12345, USA |
 47.6223000127926 | -122.336745971039
(1 row)
```

データベーステーブルを更新し、列にデータを入力します。

次の例では、テーブルを更新し、SearchPlaceIndexForTextクエリの結果を列に入力します。

```
WITH source_data AS (
  -- select rows that have not been geocoded and created a formatted address for each
  SELECT
    id,
    address || ', ' || city || ', ' || state || ', ' || zip AS formatted_address
  FROM addresses
  WHERE label IS NULL
  -- limit the number of rows that will be geocoded; remove this to geocode the entire
 table
  LIMIT 1
),
geocoded_data AS (
  -- geocode each row and keep it linked to the source's ID
  SELECT
    id,
    (f_SearchPlaceIndexForText(formatted_address)).*
  FROM source_data
)
UPDATE addresses
-- populate columns
SET
  normalized_address = geocoded_data.label,
  latitude = ST_Y(geocoded_data.geom),
  longitude = ST_X(geocoded_data.geom)
FROM geocoded_data
-- ensure that rows match
WHERE addresses.id = geocoded_data.id;
```

次のステップ

サンプルコードはGitHub、 AWS CloudFormation テンプレートを含む の Amazon Location Service サンプルリポジトリにあります。

場所インデックスリソースの管理

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、場所インデックスリソースを管理できます。

場所インデックスリソースのリストを表示する

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、場所インデックスリソースのリストを表示できます。

Console

Amazon Location コンソールを使用して場所インデックスリソースのリストを表示するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左のナビゲーションペインから、[場所インデックス] を選択します。
3. [マイ場所インデックス]で、場所インデックスリソースのリストを確認できます。

API

Amazon Location プレース API `ListPlaceIndexes` のオペレーションを利用してください。

以下は、AWS アカウント内の場所インデックスのリストを取得する API リクエストの例です。

```
POST /places/v0/list-indexes
```

以下に、`ListPlaceIndexes` のレスポンスの例を示します。

```
{
   "Entries": [
      {
         "CreateTime": 2020-10-30T01:38:36Z,
         "DataSource": "Esri",
         "Description": "string",
         "IndexName": "ExamplePlaceIndex",
         "UpdateTime": 2020-10-30T01:40:36Z
      }
   ],
   "NextToken": "1234-5678-9012"
}
```

CLI

list-place-indexesコマンドを実行します。

以下は、AWS アカウント内の場所インデックスリソースのリストを取得するための AWS CLI の例です。

```
aws location list-place-indexes
```

場所インデックスリソースの詳細を取得する

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、AWS アカウント内の任意の場所インデックスリソースに関する詳細を取得できます。

Console

Amazon Location コンソールを使用して場所インデックスリソースの詳細を確認するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左のナビゲーションペインから、[場所インデックス] を選択します。
3. [マイ場所インデックス]で、対象となる場所インデックスリソースの名前リンクを選択します。

API

Amazon Location Places API の DescribePlaceIndex の オペレーションを使用してください。

次の例は、 の場所インデックスリソースの詳細を取得するための API リクエストです*ExamplePlaceIndex*。

```
GET /places/v0/indexes/ExamplePlaceIndex
```

以下に、DescribePlaceIndex のレスポンスの例を示します。

```
{
   "CreateTime": 2020-10-30T01:38:36Z,
   "DataSource": "Esri",
   "DataSourceConfiguration": {
```

```
      "IntendedUse": "SingleUse"
   },
   "Description": "string",
   "IndexArn": "arn:aws:geo:us-west-2:123456789012:place-indexes/ExamplePlaceIndex",
   "IndexName": "ExamplePlaceIndex",
   "Tags": {
      "string" : "string"
   },
   "UpdateTime": 2020-10-30T01:40:36Z
}
```

CLI

describe-place-indexコマンドを実行します。

次の例は、 の場所インデックスリソースの詳細を取得AWS CLIするための です*ExamplePlaceIndex*。

```
aws location describe-place-index \
    --index-name "ExamplePlaceIndex"
```

場所インデックスリソースを削除する

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、AWS アカウントから場所インデックスリソースを削除できます。

Console

Amazon Location コンソールを使用して場所インデックスリソースを削除するには

> **Warning**
> このオペレーションはリソースを完全に削除します。

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左のナビゲーションペインから、[場所インデックス] を選択します。
3. [マイ場所インデックス]で、対象の場所インデックスリソースを選択します。
4. [場所インデックスを削除] を選択します。

API

Amazon Location プレース API DeletePlaceIndex のオペレーションを利用してください。

次の例は、場所インデックスリソース を削除する API リクエストですExamplePlaceIndex。

```
DELETE /places/v0/indexes/ExamplePlaceIndex
```

以下に、DeletePlaceIndex の正常なレスポンスの例を示します。

```
HTTP/1.1 200
```

CLI

delete-place-indexコマンドを実行します。

次の例は、場所インデックスリソース を削除する AWS CLI コマンドですExamplePlaceIndex。

```
aws location delete-place-index \
    --index-name "ExamplePlaceIndex"
```

Amazon Location Service によるルートの計算

Amazon Location を利用すれば、ルート計算リソースを作成して設定することで、ルートを計算するデータプロバイダーを選択することができます。

ルート計算リソースを利用すれば、AWS SDK または REST API エンドポイントの利用を通して、特定のパラメータを指定してルートを計算することができます。このルート計算リソースを利用すれば、さまざまな交通手段、回避路線、交通状況に応じて、出発地と目的地、および最大 23 のウェイポイント間のルートを計算することができます。

また、ルート計算リソースを利用して、ルートマトリックスを計算することにより、ルート計画アルゴリズムまたは新しい製品のインプットを作成することもできます。様々な出発地と目的地の間の移動時間と移動距離を計算します。ルートプランニングソフトウェアは、移動時間と移動距離のデータを読み込みんで、 1 つまたは複数のルートを改善することができます。たとえば、複数の配達ルートを計画する場合、このサービスが各目的地への最適なルートと時間を計算することができます。さまざまな交通手段、回避路線、交通状況に応じたルートのマトリックスを計算することもできます。

> **Note**
>
> ルーティングの概念の概要については、「ルート」を参照してください。

トピック

前提条件

ルートを計算する前に、以下の手順に従って操作してください。

トピック

ルート計算リソースの作成

ルートを計算する前に、まずはAWSアカウントにルート計算リソースを追加してください。

ルート計算リソースを作成するとき、利用可能なデータプロバイダーから 1 つを選択してください。

1. — 対象地域における Esri のカバレッジの詳細については、道路網とトラフィックの対象地域に関する Esri の詳細を参照してください。
2. – 対象地域における HERE Technologies のカバレッジに関する追加情報については、「HERE 自動車ルーティングのカバレッジ」および「HERE トラックルーティングのカバレッジ」を参照してください。
3. Grab — Grab のカバレッジの詳細については、「国/地域と対象地域」を参照してください。

> **Note**
>
> 配送車両や従業員など、ビジネスで使用する資産をアプリケーションで追跡またはルーティングする場合、位置情報プロバイダーに Esri を使用してはいけません。詳細については、「AWS サービス規約」のセクション 82 を参照してください。

Amazon Location Service コンソール、AWS CLI、または Amazon Location API を利用することができます。

Console

Amazon Location コンソールを利用してルート計算リソースを作成すること。

1. Amazon Location Service コンソール (https://console.aws.amazon.com/location/) を開きます。
2. 左のナビゲーションペインの選択肢の中に ルート計算ツール を選択してください。
3. 「ルート計算ツールを作成」 を選択してください。
4. 次のボックスに入力します。
   - コードネーム — ルート計算リソースのコードネームを入力してください。例えば、 などです*ExampleCalculator*。最大 100 文字。には、英数字、ピリオド (.)、ハイフン (-)、アンダースコア (_) を含めることができます。
   - 説明 – 任意の説明を入力します。
5. データプロバイダーについては、ご利用されたいルート計算ツールのデータプロバイダーを選択してください。
6. （オプショナル）タグに、タグのキーと値を入力します。これにより、新しいルート計算リソースにタグが追加されます。詳細については、リソースにタグを付けるを参照してください。
7. 「ルート計算ツールを作成」 を選択してください。

API

Amazon Location API を利用してルート計算リソースを作成すること

Amazon Location プレース API `CreateRouteCalculator` のオペレーションを利用してください。

次の例は、データプロバイダー *Esri* *ExampleCalculator*を使用して というルート計算リソースを作成する API リクエストです。

```
POST /routes/v0/calculators
Content-type: application/json

{
   "CalculatorName": "ExampleCalculator",
   "DataSource": "Esri",
   "Description": "string",
   "Tags": {
      "Tag1" : "Value1"
   }
}
```

AWS CLI

AWS CLIコマンドを利用してルート計算リソースを作成すること

`create-route-calculator` コマンドを実行します。

次の例では、データプロバイダーとして *Esri* *ExampleCalculator*を使用して、 というルート計算リソースを作成します。

```
aws location \
  create-route-calculator \
  --calculator-name "ExampleCalculator" \
  --data-source "Esri" \
  --tags Tag1=Value1
```

> **Note**
>
> 請求は使用状況によって異なります。他の AWS サービスの利用料金が請求される場合があります。詳細については、「Amazon Location Service 料金」を参照してください。

リクエストを認証する

ルート計算リソースの作成、並びに位置情報をアプリケーションに組み込む準備ができたら、リクエストの認証方法を選択してください。

- サービスにアクセスする方法については、「Amazon Location Service へのアクセス」を参照してください。
- ウェブサイトには匿名機能を利用されるユーザーがいらっしゃる場合、API キーまたは Amazon Cognito を使うのはお勧めです。

  例

  次の例は、AWS JavaScript SDK v3 と Amazon Location を使用して、認証に API キーを使用する方法を示していますJavaScript 認証ヘルパー。

```
import { LocationClient, CalculateRouteCommand } from "@aws-sdk/client-location";
import { withAPIKey } from "@aws/amazon-location-utilities-auth-helper";

const apiKey = "v1.public.your-api-key-value"; // API key

// Create an authentication helper instance using an API key
const authHelper = await withAPIKey(apiKey);

const client = new LocationClient({
  region: "<region>", // region containing Cognito pool
  ...authHelper.getLocationClientConfig(), // Provides configuration required to make
 requests to Amazon Location
});

const input = {
  CalculatorName: "ExampleCalculator",
  DeparturePosition: [-123.4567, 45.6789],
  DestinationPosition: [-123.123, 45.234],
};

const command = new CalculateRouteCommand(input);

const response = await client.send(command);
```

1 つのルートの計算

Amazon Location Service を使って、さまざまな交通手段、回避路線、交通状況に応じて、出発地から目的地まで、最大には 23 のウェイポイントを設置することができ、そのすべての経由地点の間のルートを計算することができます。

> **Note**
>
> まず、ルート計算リソースを作成して、それからAmazon Location へのリクエストの認証を設定するのは必要です。詳細については、「前提条件」を参照してください。

ルート計算の開始

`CalculateRoute`オペレーションで簡単にクエストを送信します。シンプルなリクエストには、次のフィールド必要です。

- `DeparturePosition`— ルートを計算する開始位置。`[longitude, latitude]`と定義されます。
- `DestinationPosition`— ルートを計算する終了位置。`[longitude, latitude]`と定義されています。

> **Note**
>
> 出発地または目的地の位置が地図に載っていない場合、Amazon Locationはその位置を最も近い道路に変更します。

もしリクエストがあれば、オプションでウェイポイント、出発時刻、または移動モードを選択することができます。

AWS CLIまたは Amazon Location API を使用することができます。

API

次の例は、ルート計算リソース を使用したCalculateRouteリクエストです*ExampleCalculator*。このリクエストは、出発位置 *-122.7565*, *49.0021*から目的地位置 *-122.3394*, *47.6159*までのルートを計算することを説明します。

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159]
}
```

AWS CLI

次の例は、ルート計算リソース を使用するcalculate-routeコマンドで すExampleCalculator。このリクエストは、出発位置 -122.7565, 49.0021から目的地位置 -122.3394, 47.6159までのルートを計算することを説明します。

```
aws location \
    calculate-route \
        --calculator-name ExampleCalculator \
        --departure-position -122.7565 49.0021 \
        --destination-position -122.3394 47.6159
```

デフォルトでは、Distance は JSON 形式で応答を返します。以下のオプションの中のパラメータを利用すれば、計測単位をマイルに変更することができます。

- DistanceUnit— 距離を計算するシステムを指定します。

Example

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159],
   "DistanceUnit": "Miles"
}
```

ウェイポイントの設定

ルートを計算するとき、ウェイポイントの位置を利用して、出発地と目的地の間の経由地点が最大 23 か所を設置することができます。

- WaypointPositions— 出発地と目的地の間のルートに含まれる経由地点の順序を指定することが必要です。

> **Note**
>
> 出発地または目的地の位置が地図に載っていない場合、Amazon Locationはその位置を最も近い道路に変更します。

Example

次のCalculateRouteリクエストは、2 つのウェイポイントのあるルートを計算します。

- 出発地は -122.7565、49.0021で、目的地の位置は -122.3394、47.6159です。
- リクエストパラメータ
  - 最初のウェイポイントは *-122.1884#48.0936*です。
  - 2 番目のウェイポイントは *-122.3493#47.6205*です。
- *2 ## true* に設定してください。
  - IncludeLegGeometry— 2 つの位置の間の各路線の形状データがレスポンスに含まれます。

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159],
   "WaypointPositions":[
        [-122.1884,48.0936],
        [-122.3493,47.6205]
    ],
   "IncludeLegGeometry": true
}
```

レスポンスの例

以下は、*true IncludeLegGeometry* に設定した Amazon Location Routes API CalculateRoute からのオペレーションを呼び出したときの、対応するレスポンスを含むリクエストの例です。これには、レスポンス内のポジションペアの間の各パスのラインストリングジオメトリが含まれます。

Example request

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159],
   "IncludeLegGeometry": true
```

```
}
```

Example response

```
{
   "Legs": [
      {
         "Distance": 178.5,
         "DurationSeconds": 6480,
         "EndPosition": [-122.3394,47.6159],
         "Geometry": {
            "LineString": [
               [-122.7565,49.0021],
               [-122.3394,47.6159]
            ]
         },
         "StartPosition": [-122.7565,49.0021],
         "Steps": [
            {
               "Distance": 178.5,
               "DurationSeconds": 6480,
               "EndPosition": [-122.3394,47.6159],
               "GeometryOffset": 0,
               "StartPosition": [-122.7565,49.0021]
            }
         ]
      }
   ],
   "Summary": {
      "DataSource": "Esri",
      "Distance": 178.5,
      "DistanceUnit": "Kilometers",
      "DurationSeconds": 6480,
      "RouteBBox": [
        -122.7565,49.0021,
        -122.3394,47.6159
    ]
   }
}
```

ルートマトリックスを利用するルート計画

Amazon Location Service を利用すれば、ルートの計画および改善したソフトウェアへの入力ができます。出発地と目的地間のルートについて、所要時間や移動距離などのルートの計算が作成できます。

たとえば、出発地の A と B、目的地の X と Y を入力すると、Amazon Location Service は A から X、A から Y、B から X、B から Y までのルートの移動時間と移動距離を計算してルートを出します。

さまざまな交通手段、回避路線、交通状況に基づいてルートを計算することができます。たとえば、車両は長さが 35 フィートのトラックであると入力すると、それらの規制に基づいて移動時間と移動距離を計算してルートを決めます。

返された結果 (そして計算されたルート数) は、出発地点の数を目的地の数の掛け算の結果です。料金は、サービスへのリクエストごとではなく、計算されたルートごとに請求されるため、10 か所出発地と 10 か所の目的地の掛け算の結果、つまり、ルートマトリックスは 100 ルートとして請求されます。

ルートマトリックスの計算

複数の出発地と目的地の間のルートのマトリックスを計算することができます。ルートの結果には、移動時間と移動距離が含まれます。

前提条件

- まず、ルート計算リソースを作成して、それからAmazon Location へのリクエストの認証を設定するのは必要です。詳細については、「前提条件」を参照してください。

`CalculateRouteMatrix`オペレーションで簡単にクエストを送信します。最小のリクエストには次のフィールドが必ず含まれます：

- `DeparturePositions`— ルートを計算するためのすべての出発地です。`[longitude, latitude]`の配列として定義されます。
- `DestinationPositions`— ルートを計算するためのすべての目的地です。`[longitude, latitude]`の配列として定義されます。

> **Note**
>
> 出発地または目的地の位置が地図に載っていない場合、Amazon Locationはその位置を最も近い道路に変更します。

オプションで出発時間や旅行モードを指定することもできます。

AWS CLIまたは Amazon Location API を使用することができます。

API

次の例は、ルート計算リソース を使用したCalculateRouteMatrixリクエストです*ExampleCalculator*。このリクエストでは、出発位置 *-122.7565*, *49.0021*と *-122.2014*, *47.6101*から目的地位置 *-122.3394*, *47.6159*と *-122.4813*, *48.7511*までのルートマトリックスを計算することを指定しています。

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route-matrix
Content-type: application/json
{
   "DeparturePositions": [
       [-122.7565,49.0021],
       [-122.2014,47.6101]
   ],
   "DestinationPositions": [
       [-122.3394, 47.6159],
       [-122.4813,48.7511]
   ]
}
```

AWS CLI

次の例は、ルート計算リソース を使用するcalculate-route-matrixコマンドです*ExampleCalculator*。このリクエストでは、出発位置 *-122.7565*, *49.0021*と *-122.2014*, *47.6101*から目的地位置 *-122.3394*, *47.6159*と *-122.4813*, *48.7511*までのルートマトリックスを計算することを指定しています。

```
aws location \
    calculate-route-matrix \
        --calculator-name ExampleCalculator \
        --departure-positions "[[-122.7565,49.0021],[-122.2014,47.6101]]" \
```

```
    --destination-positions "[[-122.3394,47.6159],[-122.4813,48.7511]]"
```

デフォルトでは、レスポンスはDistanceキロメートル単位で返される。以下のオプションの中のパラメータを利用すれば、計測単位をマイルに変更することができます。

- DistanceUnit— 距離を計算するシステムを指定します。

Example

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route-matrix
Content-type: application/json
{
   "DeparturePositions": [
       [-122.7565,49.0021],
       [-122.2014,47.6101]
   ],
   "DestinationPositions": [
       [-122.3394, 47.6159],
       [-122.4813,48.7511]
   ],
   "DistanceUnit": "Miles"
}
```

出発地と目的地のポジションの制限

ルートマトリックスを計算する場合、出発地と目的地の位置には制限があります。これらの制限は、RouteCalculator リソースを利用するプロバイダーによって異なります。

制限	Esri	grab	こちら
ポジション数	出発地と目的地はいずれも10カ所までです。	出発地350カ所、目的地350カ所までです。	出発地350カ所、目的地350カ所までです。 より長いルートには、これ以上の制限があります。セクションを参照してください。

制限	Esri	grab	こちら
ポジションの間の距離	一組の出発地と目的地の最大距離は、400km以内（徒歩ルートの場合は40km以内）です。		すべての出発地と目的地を中心とした円の最大直径 は180 km です。 より長いルートには、これ以上の制限があります。セクションを参照してください。
ルートの長さ	ルートの移動総時間が 400 分を超えると、ルートの計算が完成できません。		出発地と目的地を中心とした円から 10 km 以上離れているルートは計算できません。 より長いルートには、これ以上の制限があります。セクションを参照してください。
リージョン	韓国ではルートマトリックスの計算はサポートされていません。	東南アジアでは利用可能です。対応できる国/地域のリストと詳細については、国/地域と対象地域を参照してください。	追加の制限

より長いルートの計画

ルート結果のマトリックスを計算すると効率的なルートプランニングに役立ちますが、計算には時間がかかる場合があります。Amazon Location Service のデータプロバイダーはすべて、計算できるルートの数または距離に制限が設けされています。たとえば、HERE では 350 の出発地と目的地の

間のルートを作成できますが、それらの位置は 180 km の範囲内にある必要があります。もっと長いルートを計画したい場合はどうでしょうか？

データプロバイダーとして HERE を使用して、少数のルートについて長さに制限のないルートマトリックスを計算することができます。RouteCalculatorこれによって CalculateRouteMatrix API を呼び出す方法が変わるわけではありません。Amazon Location では、要件を満たせばより長いルートが許可されるのみです。

より長いルート計算の要件は以下のとおりです。

- RouteCalculatorHERE データプロバイダーを使用する必要があります。
- 出発地の数は最大 15 か所です。
- 計算するルートの総数は最大 100 です。
- ルートが 1,000 km を超える場合、通行料回避を伴うトラックルートには長距離ルーティングは許可されません。 この組み合わせは計算に時間がかかり、通話がタイムアウトする可能性があります。これらのルートは、 CalculateRouteオペレーションを使用して個別に計算できます。

通話がこれらの要件を満たさない場合 (たとえば、1 回の通話で 150 のルート計算が要求される場合) 、CalculateRouteMatrix短い方のルートのルールのみを許可するように返されます。位置が180kmの円以内であれば、ルートの計算ができます。

長いルートを計算する際には、次の点に注意してください。

- ルートが長くなると、Amazon Location API の最大時間よりも計算に時間がかかることがあります。特定のルートでタイムアウトが頻発する場合は、CalculateRouteMatrixを呼び出すたびに、より少ない数のルートを試すことができます。
- CalculateRouteMatrixリクエストに目的地や出発地を追加すると、操作がより制限されたモードに切り替わり、作成するルートが少なくなると、問題なく計算できるルートでもエラーが発生することがあります。この場合、目的地または出発地の数を減らし、必要なルート計算のフルセットを取得するために複数のリクエストを行ってください。

レスポンスの例

以下は、Amazon Location ルート API CalculateRouteMatrix からオペレーションを呼び出した場合のリクエストとそれに対応するレスポンスを含むリクエストの例です。

Example request

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route-matrix
Content-type: application/json
{
   "DeparturePositions": [
       [-122.7565,49.0021],
       [-122.2014,47.6101]
   ],
   "DestinationPositions": [
       [-122.3394, 47.6159],
       [-122.4813,48.7511]
   ]
}
```

Example response

```
{
    "RouteMatrix": [
        [
            {
                "Distance": 178.764,
                "DurationSeconds": 7565
            },
            {
                "Distance": 39.795,
                "DurationSeconds": 1955
            }
        ],
        [
            {
                "Distance": 15.31,
                "DurationSeconds": 1217
            },
            {
                "Distance": 142.506,
                "DurationSeconds": 6279
            }
        ]
    ],
    "Summary": {
        "DataSource": "Here",
        "RouteCount": 4,
```

```
        "ErrorCount": 0,
        "DistanceUnit": "Kilometers"
    }
}
```

地図に載っていない位置

ユーザーがCalculateRouteやCalculateRouteMatrixを使って、地図に載っていない出発地、目的地、またはウェイポイントの位置を検索する場合、Amazon Location はその位置を最も近い位置に移動します。

次のCalculateRouteリクエストでは、地図に載っていない出発地と目的地の位置を指定しています。

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-123.128014, 49.298472],
   "DestinationPosition": [-123.134701, 49. 294315]
}
```

結果のレスポンスでは、近くの道路にスナップされた位置が返されます。

```
{
   "Legs": [
      {
         "StartPosition": [-123.12815, 49.29717],
         "EndPosition":   [-123.13375, 49.2926],
         "Distance": 4.223,
         "DurationSeconds": 697,
         "Steps": [
            {
               "StartPosition": [ -123.12815, 49.29717 ],
               "EndPosition":   [ -123.12806, 49.29707 ],
               "Distance": 0.013,
               "DurationSeconds": 8
            },
            {
               "StartPosition": [ -123.12806, 49.29707 ],
               "EndPosition":   [ -123.1288, 49.29659 ],
               "Distance": 0.082,
```

```
            "DurationSeconds": 36
        },
        {
            "StartPosition": [ -123.1288, 49.29659 ],
            "EndPosition":   [ -123.12021, 49.29853 ],
            "Distance": 0.742,
            "DurationSeconds": 128
        },
        {
            "StartPosition": [ -123.12021, 49.29853 ],
            "EndPosition":   [ -123.1201, 49.29959 ],
            "Distance": 0.131,
            "DurationSeconds": 26
        },
        {
            "StartPosition": [ -123.1201, 49.29959 ],
            "EndPosition":   [ -123.13562, 49.30681 ],
            "Distance": 1.47,
            "DurationSeconds": 238
        },
        {
            "StartPosition": [ -123.13562, 49.30681 ],
            "EndPosition":   [ -123.13693, 49.30615 ],
            "Distance": 0.121,
            "DurationSeconds": 28
        },
        {
            "StartPosition": [ -123.13693, 49.30615 ],
            "EndPosition":   [ -123.13598, 49.29755 ],
            "Distance": 0.97,
            "DurationSeconds": 156
        },
        {
            "StartPosition": [ -123.13598, 49.29755 ],
            "EndPosition":   [ -123.13688, 49.29717 ],
            "Distance": 0.085,
            "DurationSeconds": 15
        },
        {
            "StartPosition": [ -123.13688, 49.29717 ],
            "EndPosition":   [ -123.13375, 49.2926 ],
            "Distance": 0.609,
            "DurationSeconds": 62
        }
```

```
         ]
      }
   ],
   "Summary": {
      "RouteBBox": [ -123.13693, 49.2926, -123.1201, 49.30681 ],
      "DataSource": "Here",
      "Distance": 4.223,
      "DurationSeconds": 697,
      "DistanceUnit": "Kilometers"
   }
}
```

出発時刻

デフォルトでは、CalculateRouteやCalculateRouteMatrixを指定する場合、リクエストに出発時刻を指定しなかったりすると、計算されたルートには最適な交通状況が反映されます。

以下のオプションのいずれか利用して、特定の出発時間を設定して、選択したデータプロバイダーのライブ交通状況と交通状況の予測が計算できます。

- DepartNow— *###* に設定すると、リアル交通状況を利用して最速の移動経路を計算します。
- DepartureTime— 指定すると、要求された時間帯の予測と既知の交通状況で計算することができます。次の形式で定義されます:YYYY-MM-DDThh:mm:ss.sssZ.

Example

次のCalculateRouteリクエストでは、出発時刻を 2024 年 7 月 2 日 12:15:20 UTC に設定しています。

```
POST /routes/v0/calculators/ExampleCalculator/calculate/route
Content-type: application/json
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159],
   "WaypointPositions":[
        [-122.1884,48.0936],
        [-122.3493,47.6205]
    ]
    "IncludeLegGeometry": true,
    "DepartureTime": 2024-07-02T12:15:20.000Z,
}
```

旅行モード

CalculateRouteCalculateRouteMatrixまたはを使用するときに旅行モードを設定することができます。交通手段は移動速度と道路の適合性に影響しています。デフォルトでは旅行モードの交通手段は車ですが、以下のオプション・パラメータで交通手段を指定することができます：

- TravelMode – *Walking*またがルートを計算するとき、交通手段を指定することもできます。たとえば、*Bicycle*、*Car*、*Motorcycle*、*Truck*などです。

制約事項

- 旅行モードを Walking に指定し、データプロバイダーが Esri の場合、出発地と目的地は 40 km 以内でなければなりません。
- BicycleまたはMotorcycle、Grab をデータプロバイダーとして使用している場合にのみ利用できます。
- 特定の都市では、Grab は、BicycleやWalkingのみのルートが提供されています。詳細については、「国/地域と対象地域」を参照してください。
- Grabをデータプロバイダーとして使用する場合、Truckでのルートは計返されません。

その他の設定

TravelModeでルートを計算するとき、*Car*を指定する場合：

- CarModeOptions— 車で移動することを優先順位に指定してください、例えば、*AvoidFerries*、*AvoidTolls*などです。

TravelModeでルートを計算するとき、*Truck*を指定する場合：

- TruckModeOptions— トラックで移動する際に、*AvoidFerries*や*AvoidTolls*といったルートを希望に加え、*TruckDimensions*や*TruckWeight*が通れるルートを指定します。

Example

次のCalculateRouteリクエストは、旅行モードとして*Truck*を指定しています。その他のルート制限には、フェリーを使用するルートの回避や、トラックの寸法や重量に対応できない道路の回避などがあります。

```
{
   "DeparturePosition": [-122.7565,49.0021],
   "DestinationPosition": [-122.3394, 47.6159],
   "DepartNow": true,
   "TravelMode": "Truck",
   "TruckModeOptions": {
      "AvoidFerries": true,
      "AvoidTolls": false,
      "Dimensions": {
         "Height": 4.5,
         "Length": 15.5,
         "Unit": "Meters",
         "Width": 4.5
      },
      "Weight": {
         "Total": 7500,
         "Unit": "Pounds"
      }
   }
}
```

ルート計算リソースの管理

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、ルート計算リソースを管理できます。

ルート計算ツールのリストを表示する

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、ルート計算ツールのリストを表示できます。

Console

Amazon Location コンソールを使用してルート計算ツールのリストを表示するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左側のナビゲーションペインから、[ルート計算ツール] を選択します。
3. [マイルート計算ツール] でルート計算ツールの詳細を確認できます。

API

Amazon Location Routes API の ListRouteCalculatorsの オペレーションを使用してください。

以下は、AWS アカウント内のルート計算ツールのリストを取得する API リクエストの例です。

```
POST /routes/v0/list-calculators
```

以下に、ListRouteCalculators のレスポンスの例を示します。

```
{
       "Entries": [
          {
             "CalculatorName": "ExampleCalculator",
             "CreateTime": 2020-09-30T22:59:34.142Z,
             "DataSource": "Esri",
             "Description": "string",
             "UpdateTime": 2020-09-30T23:59:34.142Z
          }
       ],
       "NextToken": "1234-5678-9012"
    }
```

CLI

`list-route-calculators`コマンドを実行します。

以下は、AWS アカウント内のルート計算ツールのリストを取得するための AWS CLI の例です。

```
aws location list-route-calculators
```

ルート計算ツールの詳細を取得する

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、AWS アカウント内のあらゆるルート計算ツールの詳細を取得できます。

Console

Amazon Location コンソールを使用してルート計算ツールの詳細を表示するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左側のナビゲーションペインから、[ルート計算ツール] を選択します。
3. [マイルート計算ツール] で、対象となるルート計算ツールの名前リンクを選択します。

API

Amazon Location Routes API の DescribeRouteCalculatorの オペレーションを使用してください。

次の例は、 のルート計算ツールの詳細を取得するための API リクエストで すExampleCalculator。

```
GET /routes/v0/calculators/ExampleCalculator
```

以下に、DescribeRouteCalculator のレスポンスの例を示します。

```
{
       "CalculatorArn": "arn:aws:geo:us-west-2:123456789012:route-
calculator/ExampleCalculator",
       "CalculatorName": "ExampleCalculator",
       "CreateTime": 2020-09-30T22:59:34.142Z,
       "DataSource": "Esri",
       "Description": "string",
       "Tags": {
          "Tag1" : "Value1"
       },
       "UpdateTime": 2020-09-30T23:59:34.142Z
    }
```

CLI

describe-route-calculatorコマンドを実行します。

次の例は、 のルート計算ツールの詳細を取得AWS CLIするための ですExampleCalculator。

```
aws location describe-route-calculator \
        --calculator-name "ExampleCalculator"
```

ルート計算ツールを削除する

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、AWS アカウントからルート計算ツールを削除できます。

Console

Amazon Location コンソールを使用してルート計算ツールを削除するには

> **Warning**
> このオペレーションはリソースを完全に削除します。

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左側のナビゲーションペインから、[ルート計算ツール] を選択します。
3. [マイルート計算ツール] で、対象のルート計算ツールを選択します。
4. [ルート計算ツールを削除] を選択します。

API

Amazon Location Routes API の `DeleteRouteCalculator`の オペレーションを使用してください。

次の例は、ジオフェンスコレクション を削除する API リクエストです*ExampleCalculator*。

```
DELETE /routes/v0/calculators/ExampleCalculator
```

以下に、`DeleteRouteCalculator` のレスポンスの例を示します。

```
HTTP/1.1 200
```

CLI

`delete-route-calculator`コマンドを実行します。

次の例は、ジオフェンスコレクション を削除する AWS CLI コマンドです*ExampleCalculator*。

```
aws location delete-route-calculator \
        --calculator-name "ExampleCalculator"
```

Amazon Location を使用して対象エリアをジオフェンシングする

ジオフェンシングアプリケーションは、追跡対象デバイスの位置を、事前に登録された対象エリアと比較して評価します。これにより、位置の更新に基づいてアクションを実行できます。例えば、モバイルアプリでコーヒーを注文した顧客が店舗の近くに来たときに通知をするイベントを開始できます。

> **Note**
>
> ジオフェンシングとトラッカーの概念の概要については、「ジオフェンスとトラッカー」を参照してください。

ガイドのこのセクションでは、Amazon Location Service を使用してジオフェンシングアプリケーションを作成する step-by-step 手順について説明します。

手順の概要

1. 対象エリア周辺にジオフェンスを追加し、ジオフェンスコレクションリソースに保存します。
2. 追跡対象デバイスの追跡を開始し、デバイスの位置履歴をトラッカーリソースに保存します。
3. トラッカーリソースをジオフェンスコレクションリソースにリンクして、デバイス位置情報の更新がすべてのジオフェンスに対して自動的に評価されるようにします。
4. Amazon Location Tracker を使用してデバイスの位置履歴を保存したくない場合は、デバイスの位置をジオフェンスコレクションリソースと直接比較して評価できます。

ジオフェンシングソリューションを実装すると、ジオフェンスコレクションリソースから次のイベントが送信されます。

- `ENTER` — 追跡対象デバイスがジオフェンスコレクションのジオフェンスに入る。
- `EXIT` — 追跡対象デバイスがジオフェンスコレクションのジオフェンスから出る。

Amazon を使用して EventBridge 、イベントを他の場所にルーティングすることでイベントに反応できます。

各デバイスから Amazon Location Service API を介して更新を送信する代わりに、MQTT を使用してデバイスの更新を送信することもできます。

以下のトピックでは、これらの手順と代替方法について詳しく説明します。

トピック

ジオフェンスを追加する

ジオフェンスは、点と頂点から成る閉じた境界で、対象地域を定義します。ジオフェンスコレクションは 1 つ以上のジオフェンスを格納および管理します。

Amazon Location ジオフェンスコレクションには、GeoJSON (RFC 7946) と呼ばれる標準の地理空間データ形式を使用して定義されたジオフェンスが格納されます。geojson.io などのツールを無料で使用して、ジオフェンスをグラフィカルに描画し、GeoJSON 出力ファイルを保存できます。

> **Note**
>
> Amazon Location は、穴のあるポリゴン、マルチポリゴン、時計回りのポリゴン、および反時計午線を横切るジオフェンスをサポートしていません。

ジオフェンスコレクションを作成する

Amazon Location コンソール、、または Amazon Location APIs を使用して AWS CLIジオフェンスを保存および管理するためのジオフェンスコレクションを作成します。

Console

Amazon Location コンソールを使用してジオフェンスコレクションを作成するには

1. Amazon Location Service コンソール (https://console.aws.amazon.com/location/) を開きます。
2. 左のナビゲーションペインから、[ジオフェンスコレクション] を選択します。
3. [ジオフェンスコレクションの作成] を選択します。
4. 次のボックスに入力します。
   - 名前 - 一意の名前を入力します。例えば、 などです*ExampleGeofenceCollection*。最大 100 文字。英数字、ピリオド (.)、ハイフン (-)、アンダースコア (_) を含めることができます。
   - 説明 — リソースを区別するための説明を任意で入力します。
5. EventBridge をターゲット CloudWatch とするルール では、ジオフェンスイベントへの対応を開始するオプションの EventBridge ルールを作成できます。 ???これにより、Amazon Location は Amazon CloudWatch Logs にイベントを発行できます。
6. （ オプショナル ）タグに、タグのキーと値を入力します。これにより、新しいジオフェンスコレクションにタグが追加されます。詳細については、「Amazon Location Service リソースにタグを付ける」を参照してください。
7. (オプション) [顧客管理キーの暗号化] で、[顧客管理キーの追加] を選択できます。これにより、デフォルトの AWS 所有暗号化で作成、所有、管理する対称カスタマーマネージドキーが追加されます。詳細については、「保管中のデータの暗号化」を参照してください。
8. [ジオフェンスコレクションの作成] を選択します。

API

Amazon Location API を使用してジオフェンスコレクションを作成するには

Amazon Location Geofences API の `CreateGeofenceCollection` オペレーションを使用してください。

次の例では、 API リクエストを使用して、 というジオフェンスコレクションを作成します*ExampleGeofenceCollection*。ジオフェンスコレクションは、カスタマーデータ を暗号化するためのカスタマーマネージド AWS KMS キーに関連付けられています。

```
POST /geofencing/v0/collections
```

```
Content-type: application/json

{
   "CollectionName": "ExampleGeofenceCollection",
   "Description": "Geofence collection 1 for shopping center",
   "KmsKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
   "Tags": {
      "Tag1" : "Value1"
   }
}
```

AWS CLI

AWS CLI コマンドを使用してジオフェンスコレクションを作成するには

create-geofence-collectionコマンドを実行します。

次の例では、 を使用して AWS CLI 、 というジオフェンスコレクションを作成しますExampleGeofenceCollection。ジオフェンスコレクションは、カスタマーデータ を暗号化するためのカスタマーマネージド AWS KMS キーに関連付けられています。

```
aws location \
  create-geofence-collection \
  --collection-name "ExampleGeofenceCollection" \
  --description "Shopping center geofence collection" \
  --kms-key-id "1234abcd-12ab-34cd-56ef-1234567890ab" \
  --tags Tag1=Value1
```

> **Note**
>
> 請求は使用状況によって異なります。他の AWS サービスの利用料金が請求される場合があります。詳細については、「Amazon Location Service 料金」を参照してください。

ジオフェンスを描く

ジオフェンスコレクションを作成したので、ジオフェンスを定義できるようになりました。ジオフェンスはポリゴンまたは円として定義されます。ポリゴンジオフェンスを描画するには、geojson.io などの GeoJSON 編集ツールを使用できます。

円形ジオフェンスを描画するには、円の中心点と半径を定義する必要があります。例えば、デバイスが特定の位置から 50 メートル以内に入ったときに通知を受け取るようにジオフェンスを作成する場合は、その位置の緯度と経度を使用して半径を 50 メートルと指定します。

Amazon Location Service API を使用すると、キーと値をペアとしたメタデータをジオフェンスに追加することもできます。これは、ジオフェンスの種類、アプリケーション固有のその他の情報など、ジオフェンスの情報を保存する上で役立ちます。このメタデータは、次のような場合に使用できます。Amazon での Amazon Location Service イベントへの対応 EventBridge

ポリゴンジオフェンスを追加する

このセクションでは、ポリゴンジオフェンスの作成について説明します

GeoJSON ツールを使用してジオフェンスを描画する

ジオフェンスコレクションを作成したので、geojson.io などの GeoJSON 編集ツールを使用してジオフェンスを定義できます。

GeoJSON ファイルを作成するには

1. GeoJSON 編集ツールを開きます。例えば、geojson.io を使用します。
2. [ポリゴンを描画] アイコンを選択し、対象エリアを描画します。
3. [保存] を選択し、ドロップダウンメニューから [GeoJSON] を選択します。

GeoJSON ジオフェンスをジオフェンスコレクションに入れる

結果の GeoJSON ファイルを使用して、Amazon Location Service コンソール、 AWS CLI、または Amazon Location APIs を使用してジオフェンスをアップロードできます。

Console

Amazon Location Service コンソールを使用してジオフェンスコレクションにジオフェンスを追加するには

1. Amazon Location Service コンソール (https://console.aws.amazon.com/location/) を開きます。
2. 左のナビゲーションペインから、[ジオフェンスコレクション] を選択します。
3. [ジオフェンスコレクション] リストから、対象のジオフェンスコレクションの名前リンクを選択します。

4. [ジオフェンス] で [ジオフェンスを作成] を選択します。
5. [ジオフェンスを追加] ウィンドウで、GeoJSON をウィンドウにドラッグアンドドロップします。
6. [ジオフェンスを追加] を選択します。

API

Amazon Location API を使用してジオフェンスを追加するには

Amazon Location Geofences API の PutGeofence オペレーションを使用してください。

次の例では、 API リクエストを使用して、ID *GEOFENCE-EXAMPLE1* を というジオフェンスコレクションに追加します*ExampleGeofenceCollection*。キー Type と値 loadingArea 1 つのペアを有するジオフェンスメタデータプロパティも指定します。

```
PUT /geofencing/v0/collections/ExampleGeofenceCollection/geofence/GEOFENCE-EXAMPLE1
    Content-type: application/json

    {
       "GeofenceProperties": {
          "Type" : "loadingArea"
       },
       "Geometry": {
          "Polygon": [
            [
                [-5.716667, -15.933333],
                [-14.416667, -7.933333],
                [-12.316667, -37.066667],
                [-5.716667, -15.933333]
            ]
          ]
       }
    }
```

または、BatchPutGeofence オペレーションを使用して複数のジオフェンスを追加することもできます。

```
POST /geofencing/v0/collections/ExampleGeofenceCollection/put-geofences
    Content-type: application/json
```

```
    {
       "Entries": [
          {
             "GeofenceProperties": {
                "Type" : "loadingArea"
             },
             "GeofenceId": "GEOFENCE-EXAMPLE1",
             "Geometry": {
                "Polygon": [
                   [
                      [-5.716667, -15.933333],
                      [-14.416667, -7.933333],
                      [-12.316667, -37.066667],
                      [-5.716667, -15.933333]
                   ]
                ]
             }
          }
       ]
    }
```

AWS CLI

AWS CLI コマンドを使用してジオフェンスコレクションにジオフェンスを追加するには

put-geofenceコマンドを実行します。

次の例では、 を使用して AWS CLI 、 というジオフェンスコレクションにジオフェンスを追加しますExampleGeofenceCollection。

```
$ aws location \
        put-geofence \
            --collection-name ExampleGeofenceCollection \
            --geofence-id ExampleGeofenceTriangle \
            --geofence-properties '{"Type": "loadingArea"}' \
            --geometry 'Polygon=[[[-5.716667, -15.933333],[-14.416667, -7.933333],
[-12.316667, -37.066667],[-5.716667, -15.933333]]]'
    {
        "CreateTime": "2020-11-11T00:16:14.487000+00:00",
        "GeofenceId": "ExampleGeofenceTriangle",
        "UpdateTime": "2020-11-11T00:19:59.894000+00:00"
    }
```

円形ジオフェンスを追加する

このセクションでは、円形ジオフェンスの作成について説明します。円の中心にしたい点の緯度と経度、および円の半径 (メートル単位) を知っておく必要があります。Amazon Location API または AWS CLIを使用して、円形ジオフェンスを作成できます。

API

Amazon Location API を使用して円形ジオフェンスを追加するには

Amazon Location Geofences API の PutGeofence オペレーションを使用してください。

次の例では、API リクエストを使用して、ID *GEOFENCE-EXAMPLE2* が指定されたジオフェンスを というジオフェンスコレクションに追加します*ExampleGeofenceCollection*。

```
PUT /geofencing/v0/collections/ExampleGeofenceCollection/geofence/GEOFENCE-EXAMPLE2
    Content-type: application/json

    {
       "Geometry": {
          "Circle": {
            "Center": [-5.716667, -15.933333],
            "Radius": 50
          }
       }
    }
```

AWS CLI

AWS CLI コマンドを使用して円形ジオフェンスをジオフェンスコレクションに追加するには

put-geofenceコマンドを実行します。

次の例では、 を使用して AWS CLI 、 というジオフェンスコレクションにジオフェンスを追加します*ExampleGeofenceCollection*。

```
$ aws location \
        put-geofence \
            --collection-name ExampleGeofenceCollection \
            --geofence-id ExampleGeofenceCircle \
            --geometry 'Circle={Center=[-5.716667, -15.933333], Radius=50}'
```

> **Note**
>
> 次の例のように、複雑なジオメトリの JSON を独自のファイルに入れることもできます。
>
> ```
> $ aws location \
>         put-geofence \
>             --collection-name ExampleGeofenceCollection \
>             --geofence-id ExampleGeofenceCircle \
>             --geometry file:circle.json
> ```
>
> この例では、circle.json ファイルに円ジオメトリの JSON が含まれています。
>
> ```
> {
>     "Circle": {
>         "Center": [-74.006975, 40.717127],
>         "Radius": 287.7897969218057
>     }
> }
> ```

追跡を開始する

このセクションでは、デバイスの位置をキャプチャする追跡アプリケーションを構築する手順を説明します。

トラッカーを作成する

トラッカーリソースを作成して、デバイスからの位置の更新を保存して処理します。Amazon Location Service コンソール、 AWS CLI、または Amazon Location APIsを使用できます。

トラッカーリソースに保存される位置の更新はそれぞれ、位置精度の測定値と、位置またはデバイスに関するメタデータで保存したいものを最大 3 つ含めることができます。メタデータはキーと値がペアとして保存されており、速度、方向、タイヤ空気圧、エンジン温度などの情報を保存できます。

トラッカーは、位置の更新を受信するとフィルタリングします。これにより、ジッターと呼ばれるデバイス経路の視覚的なノイズが減り、誤ったジオフェンスへの出入りイベントの数が減ります。そして、ジオフェンス評価が開始される数も減り、コストを管理できます。

トラッカーには 3 つの位置フィルターオプションがあります。コスト管理と位置情報の更新におけるジッターの軽減に役立ちます。

- 精度ベース — 精度測定が可能なあらゆるデバイスで使用できます。精度測定は、ほとんどのモバイル機器で提供されます。各位置測定の精度は、GPS 衛星の受信状況、地形、Wi-Fi デバイスや Bluetooth デバイスまでの距離など、さまざまな環境要因の影響を受けます。モバイル機器を含むほとんどのデバイスでは、測定と同時にその精度も推定できます。`AccuracyBased` フィルタリングでは、デバイスの移動距離が測定精度より少ない場合、Amazon Location は位置の更新を無視します。例えば、デバイスから 2 つの連続した更新を受信し、その精度が 5 m と 10 m だった場合、デバイスの移動距離が 15 m 未満であれば 2 回目の更新は無視されます。Amazon Location は、無視した更新をジオフェンスと比較して評価したり、保存したりしません。

  精度が提供されない場合はゼロとして扱われ、測定値は完全に正確であると見なされます。

  > **Note**
  >
  > 精度ベースのフィルタリングを使用してすべてのフィルタリングを削除することもできます。精度ベースのフィルタリングを選択したものの、すべての精度データをゼロに上書きしたり、精度を完全に無視した場合、Amazon Location は更新をフィルタリングしません。

- 距離ベース — 使用しているデバイスでは正確な測定値が得られないが、フィルタリングを活用してジッターを減らし、コストを管理したい場合に使用します。`DistanceBased` フィルタリングでは、デバイスの移動が 30 m (98.4 フィート) 未満の場合、ロケーションの更新は無視されます。`DistanceBased` 位置フィルターを使用すると、Amazon Location は無視した更新をジオフェンスと比較して評価したり、保存したりしません。

  iOS および Android デバイスの平均精度を含め、ほとんどのモバイル機器の精度は 15m 以内です。ほとんどのアプリケーションでは、`DistanceBased` フィルタリングを使用することで、デバイスの軌跡を地図上に表示するときに位置が不正確になることによる影響や、デバイスがジオフェンスの境界近くにあるときに複数の連続した出入りイベントによるバウンス効果を減らすことができます。また、リンクされたジオフェンスと比較して評価するリクエストや、デバイスの位置を取得するリクエストが減るので、アプリケーションのコスト削減にも役立ちます。

- 時間ベース — (デフォルト) デバイスが位置の更新を非常に頻繁に (30 秒に 1 回以上) 送信し、すべての更新を保存せずにほぼリアルタイムでジオフェンスの評価を行いたい場合に使用します。`TimeBased` フィルタリングでは、位置の更新ごとに、リンクされたジオフェンスコレクションと比較して評価しますが、すべてが保存されるわけではありません。更新頻度が 30 秒を超える場合、一意のデバイス ID ごとに 30 秒あたり 1 つの更新のみが保存されます。

> **Note**
>
> フィルタリング方法や位置更新の頻度を決める際には、トラッキングアプリケーションのコストに留意してください。位置の更新ごとに請求され、リンクされた各ジオフェンスコレクションと比較して評価する際にも 1 回請求されます。例えば、時間ベースのフィルタリングを使用する場合で、トラッカーが 2 つのジオフェンスコレクションにリンクされている場合は、位置の更新ごとに 1 回の位置更新リクエストと 2 回のジオフェンスコレクション評価としてカウントされます。デバイスの位置更新を 5 秒ごとに報告し、時間ベースのフィルタリングを使用している場合は、デバイスごとに 1 時間あたり 720 回の位置更新リクエストと 1,440 回のジオフェンス評価について課金されます。
>
> 請求額は、各コレクション内のジオフェンスの数に影響されません。各ジオフェンスコレクションには最大 50,000 のジオフェンスを含めることができるため、ジオフェンスの評価コストを削減するために、ジオフェンスをできるだけ少ないコレクションにまとめることをお勧めします。

デフォルトでは、追跡対象デバイスがリンクされたジオフェンスに出入りするたびに EventBridge イベントが発生します。詳細については、「トラッカーをジオフェンスコレクションにリンクする」を参照してください。

トラッカーリソースのすべてのフィルターされた位置更新のイベントを有効にできます。詳細については、「トラッカーの更新イベントを有効にします。」を参照してください。

> **Note**
>
> 独自の AWS KMS カスタマーマネージドキーを使用してデータを暗号化する場合、バウンディングポリゴンクエリ機能はデフォルトで無効になります。これは、このバウンディングポリゴンクエリ機能を使用すると、デバイスの位置の表現が AWS KMS マネージドキーを使用して暗号化されないためです。ただし、デバイスの正確な位置は引き続き顧客管理キーを使用して暗号化されます。
>
> Bounding Polygon Queries 機能を使用するには、トラッカーを作成または更新するときに `KmsKeyEnableGeospatialQueries` パラメータを true に設定します。

Console

Amazon Location コンソールを使用してトラッカーを作成するには

1. Amazon Location Service コンソール (https://console.aws.amazon.com/location/) を開きます。
2. 左のナビゲーションペインから、[トラッカー] を選択します。
3. トラッカーを作成を選択します。
4. 以下のフィールドに入力します。

   - 名前 - 一意の名前を入力します。例えば、 などです*ExampleTracker*。最大 100 文字。英数字、ピリオド (.)、ハイフン (-)、アンダースコア (_) を含めることができます。
   - 説明 – 任意の説明を入力します。

5. [位置フィルター] で、トラッカーリソースの使用目的に最も適したオプションを選択します。[位置フィルター] を設定しない場合、デフォルト設定は `TimeBased` です。詳細については、このガイドの「トラッカー」と「Amazon Location Service トラッカー API リファレンス」の「`PositionFiltering`」を参照してください。
6. （オプショナル）タグに、タグのキーと値を入力します。これにより、新しいジオフェンスコレクションにタグが追加されます。詳細については、リソースにタグを付けるを参照してください。
7. (オプション) [顧客管理キーの暗号化] で、[顧客管理キーの追加] を選択できます。これにより、デフォルトの AWS 所有暗号化で作成、所有、管理する対称カスタマーマネージドキーが追加されます。詳細については、「保管中のデータの暗号化」を参照してください。
8. （オプション) でKmsKeyEnableGeospatialQueries、地理空間クエリ を有効にすることを選択できます。これにより、Bounding Polygon Queries 機能を使用しながら、顧客の AWS KMS 管理キーを使用してデータを暗号化できます。

   > **Note**
   >
   > Bounding Polygon Queries 機能を使用する場合、デバイスの位置情報は AWS KMS 顧客管理キーでは暗号化されません。ただし、デバイスの正確な位置は引き続き顧客管理キーを使用して暗号化されます。

9. （オプション) EventBridge 設定 で、フィルタリングされた位置更新の EventBridge イベントを有効にするように選択できます。これにより、このトラッカー内のデバイスの位置更新が位置フィルターの評価を満たすたびにイベントが送信されます。
10. トラッカーを作成を選択します。

API

Amazon Location API を使用してトラッカーを作成するには

Amazon Location Trackers API の `CreateTracker` オペレーションを使用してください。

次の例では、 API リクエストを使用して、 というトラッカーを作成します*ExampleTracker*。トラッカーリソースは、カスタマーデータ を暗号化するためのカスタマーマネージド AWS KMS キーに関連付けられており、 で位置の更新を有効に EventBridgeしません。

```
POST /tracking/v0/trackers
Content-type: application/json

{

   "TrackerName": "ExampleTracker",
   "Description": "string",
   "KmsKeyEnableGeospatialQueries": false,
   "EventBridgeEnabled": false,
   "KmsKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
   "PositionFiltering": "AccuracyBased",
   "Tags": {
      "string" : "string"
   }
}
```

`KmsKeyEnableGeospatialQueries` を有効にしたトラッカーを作成する

次の例では、KmsKeyEnableGeospatialQueries パラメータが true に設定されています。これにより、カスタマー AWS KMS マネージドキーを使用してデータを暗号化しながら、バウンディングポリゴンクエリ機能を使用できます。

Bounding Polygon Queries 機能の使用方法については、「???」を参照してください

> **Note**
>
> バウンディングポリゴンクエリ機能を使用する場合、デバイスの位置の表現は AWS KMS マネージドキーを使用して暗号化されません。ただし、デバイスの正確な位置は引き続き顧客管理キーを使用して暗号化されます。

```
POST /tracking/v0/trackers
```

```
Content-type: application/json

{

   "TrackerName": "ExampleTracker",
   "Description": "string",
   "KmsKeyEnableGeospatialQueries": true,
   "EventBridgeEnabled": false,
   "KmsKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
   "PositionFiltering": "AccuracyBased",
   "Tags": {
      "string" : "string"
   }
}
```

AWS CLI

AWS CLI コマンドを使用してトラッカーを作成するには

create-trackerコマンドを実行します。

次の例では、 を使用して AWS CLI というトラッカーを作成しますExampleTracker。トラッカーリソースは、カスタマーデータ を暗号化するためにカスタマーマネージド AWS KMS キーに関連付けられ、 で位置の更新を有効に EventBridgeしません。

```
aws location \
  create-tracker \
  --tracker-name "ExampleTracker" \
  --position-filtering "AccuracyBased" \
  --event-bridge-enabled false \
  --kms-key-enable-geospatial-queries false \
  --kms-key-id "1234abcd-12ab-34cd-56ef-1234567890ab"
```

KmsKeyEnableGeospatialQueries を有効にしたトラッカーを作成する

次の例では、KmsKeyEnableGeospatialQueries パラメータが true に設定されています。これにより、カスタマー AWS KMS マネージドキーを使用してデータを暗号化しながら、バウンディングポリゴンクエリ機能を使用できます。

Bounding Polygon Queries 機能の使用方法については、「???」を参照してください

> **Note**
>
> バウンディングポリゴンクエリ機能を使用する場合、デバイスの位置の表現は AWS KMS マネージドキーを使用して暗号化されません。ただし、デバイスの正確な位置は引き続き顧客管理キーを使用して暗号化されます。

```
aws location \
  create-tracker \
  --tracker-name "ExampleTracker" \
  --position-filtering "AccuracyBased" \
  --event-bridge-enabled false \
  --kms-key-enable-geospatial-queries true \
  --kms-key-id "1234abcd-12ab-34cd-56ef-1234567890ab"
```

> **Note**
>
> 請求は使用状況によって異なります。他の AWS サービスの利用料金が請求される場合があります。詳細については、「Amazon Location Service 料金」を参照してください。

トラッカーの作成後に説明 、位置フィルタリング 、およびEventBridge 設定を編集するには、トラッカーの編集 を選択します。

リクエストを認証する

トラッカーリソースを作成し、ジオフェンスに対してデバイス位置の評価を始める準備ができたら、リクエストの認証方法を選択します。

- サービスにアクセスする方法については、「Amazon Location Service へのアクセス」を参照してください。
- 認証されていないリクエストでデバイスの位置をパブリッシュしたい場合は、Amazon Cognito を使用するとよいでしょう。

  例

  次の例は、AWS JavaScript SDK v3 と Amazon Location を使用して、Amazon Cognito ID プールを認証に使用する方法を示しています JavaScript 認証ヘルパー。

```
import { LocationClient, BatchUpdateDevicePositionCommand } from "@aws-sdk/client-
location";
import { withIdentityPoolId } from "@aws/amazon-location-utilities-auth-helper";

// Unauthenticated identity pool you created
const identityPoolId = "us-east-1:1234abcd-5678-9012-abcd-sample-id";

// Create an authentication helper instance using credentials from Cognito
const authHelper = await withIdentityPoolId(identityPoolId);

const client = new LocationClient({
  region: "us-east-1", // The region containing both the identity pool and tracker
 resource
  ...authHelper.getLocationClientConfig(), // Provides configuration required to make
 requests to Amazon Location
});

const input = {
  TrackerName: "ExampleTracker",
  Updates: [
    {
      DeviceId: "ExampleDevice-1",
      Position: [-123.4567, 45.6789],
      SampleTime: new Date("2020-10-02T19:09:07.327Z"),
    },
    {
      DeviceId: "ExampleDevice-2",
      Position: [-123.123, 45.123],
      SampleTime: new Date("2020-10-02T19:10:32Z"),
    },
  ],
};

const command = new BatchUpdateDevicePositionCommand(input);

// Send device position updates
const response = await client.send(command);
```

トラッカーをデバイスの位置で更新する

デバイスを追跡するには、デバイスの位置の更新をトラッカーに送信します。これらのデバイス位置やデバイス位置履歴は、後でトラッカーリソースから取得できます。

位置の更新には、デバイス ID、タイムスタンプ、位置が含まれている必要があります。他にも、精度や、キーと値がペアとなったメタデータを最大 3 つまで含めることもできます。

トラッカーが 1 つ以上のジオフェンスコレクションにリンクされている場合、(トラッカーに指定したフィルタリングルールに従って) それらのジオフェンスに対して更新が評価されます。デバイスがジオフェンスエリアに違反した場合 (エリア内から外部に移動するか、その逆の場合) 、 でイベントを受信します EventBridge。これらの `ENTER` や `EXIT` のイベントには、デバイス ID、タイムスタンプ、関連するメタデータなどの位置の更新詳細が含まれます。

> **Note**
>
> 位置フィルターの詳細については、「トラッカーを作成する」を参照してください。
> ジオフェンスイベントの詳細については、「Amazon での Amazon Location Service イベントへの対応 EventBridge」を参照してください。

デバイスの更新を送信するには、以下のいずれかの方法を使用してください。

- MQTT 更新を AWS IoT Core リソースに送信し、トラッカーリソースにリンクします。
- Amazon Location Trackers API、 AWS CLI、または Amazon Location API を使用して位置情報の更新を送信する。AWS SDK を使用して、iOS または Android アプリケーションから API を呼び出すことができます。

API

Amazon Location API を使用して位置更新を送信するには

Amazon Location Trackers API の `BatchUpdateDevicePosition` オペレーションを使用してください。

次の例では、 API リクエストを使用して、 のデバイス位置の更新をトラッカー *`ExampleDevice`* に投稿します *`ExampleTracker`*。

```
POST /tracking/v0/trackers/ExampleTracker/positions
Content-type: application/json
```

```
{
 "Updates": [
    {
    "DeviceId": "1",
    "Position": [
   -123.12245146162303, 49.27521118043802
    ],
    "SampleTime": "2022-10-24T19:09:07.327Z",
     "PositionProperties": {
            "name" : "device1"
        },
        "Accuracy": {
           "Horizontal": 10
        }
    },

    {
    "DeviceId": "2",
    "Position": [
   -123.1230104928471, 49.27752402723152
    ],
    "SampleTime": "2022-10-02T19:09:07.327Z"
    },
    {
    "DeviceId": "3",
    "Position": [
    -123.12325592118916, 49.27340530543111
    ],
    "SampleTime": "2022-10-02T19:09:07.327Z"
    },
    {
    "DeviceId": "4",
    "Position": [
    -123.11958813096311, 49.27774641063121
    ],
    "SampleTime": "2022-10-02T19:09:07.327Z"
    },
    {
    "DeviceId": "5",
    "Position": [
    -123.1277418058896, 49.2765989015285
    ],
    "SampleTime": "2022-10-02T19:09:07.327Z"
    },
```

```
    {
    "DeviceId": "6",
    "Position": [
   -123.11964267059481, 49.274188155916534
    ],
    "SampleTime": "2022-10-02T19:09:07.327Z"
    }
    ]
}
```

AWS CLI

AWS CLI コマンドを使用して位置更新を送信するには

batch-update-device-positionコマンドを実行します。

次の例では、 を使用して AWS CLI *ExampleDevice-1* と *ExampleDevice-2* のデバイス位置の更新をトラッカー に投稿します*ExampleTracker*。

```
aws location batch-update-device-position \
--tracker-name ExampleTracker \
--updates '[{"DeviceId":"ExampleDevice-1","Position":
[-123.123,47.123],"SampleTime":"2021-11-30T21:47:25.149Z"},
{"DeviceId":"ExampleDevice-2","Position":
[-123.123,47.123],"SampleTime":"2021-11-30T21:47:25.149Z","Accuracy":
{"Horizontal":10.30},"PositionProperties":{"field1":"value1","field2":"value2"}}]'
```

トラッカーからデバイスの位置履歴を取得する

Amazon Location トラッカーリソースは、追跡対象のすべてのデバイスの位置履歴を 30 日間保持します。トラッカーリソースから、関連するすべてのメタデータを含むデバイスの位置履歴を取得できます。次の例では AWS CLI、、、または Amazon Location APIsを使用します。

API

Amazon Location API を使用してトラッカーからデバイスの位置履歴を取得するには

Amazon Location Trackers API の GetDevicePositionHistory オペレーションを使用してください。

次の例では、API URI リクエストを使用して、 の (を含む) から*ExampleTracker*始まる 19:05:07 というトラッカー*ExampleDevice*から のデバイス位置履歴を取得し、 の 19:20:07 (を除く) で終了します2020–10–02。

```
POST /tracking/v0/trackers/ExampleTracker/devices/ExampleDevice/list-positions
Content-type: application/json
{
  "StartTimeInclusive": "2020-10-02T19:05:07.327Z",
  "EndTimeExclusive": "2020-10-02T19:20:07.327Z"
}
```

AWS CLI

AWS CLI コマンドを使用してトラッカーからデバイスの位置履歴を取得するには

get-device-position-historyコマンドを実行します。

次の例では、 を使用して AWS CLI 、 の (を含む) から*ExampleTracker*始まる 19:05:07 というトラッカー*ExampleDevice*から のデバイス位置履歴を取得し、 の 19:20:07 (を除く) で終了します2020–10–02。

```
aws location \
    get-device-position-history \
        --device-id "ExampleDevice" \
        --start-time-inclusive "2020-10-02T19:05:07.327Z" \
        --end-time-exclusive "2020-10-02T19:20:07.327Z" \
        --tracker-name "ExampleTracker"
```

デバイスの位置のリストを表示する

トラッカーのデバイス位置のリストは AWS CLI、、または Amazon Location APIsで表示できます。ListDevicePositions ListDevicePositions API を呼び出すと、特定のトラッカーに関連付けられているすべてのデバイスの最新の位置のリストが返されます。デフォルトでは、この API は特定のトラッカーの最新のデバイス位置を結果ページごとに 100 件返します。特定の地域内のデバイスのみを返したい場合は、FilterGeometry パラメータを使用して Bounding Polygon Query を作成します。これにより、 を呼び出すと ListDevicePositions、ポリゴン内のデバイスのみが返されます。

> **Note**
>
> 独自の AWS KMS カスタマーマネージドキーを使用してデータを暗号化する場合、バウンディングポリゴンクエリ機能はデフォルトで無効になります。これは、この機能を使用すると、デバイスの位置の表現が AWS KMS マネージドキーを使用して暗号化されないためです。ただし、デバイスの正確な位置は引き続き顧客管理キーを使用して暗号化されます。Bounding Polygon Queries 機能の使用を選択することもできます。その場合は、トラッカーを作成または更新するときに `KmsKeyEnableGeospatialQueries` パラメータを true に設定します。

API

Amazon Location Trackers API の `ListDevicePositions` オペレーションを使用してください。

以下の例は、オプションのパラメータ `FilterGeometry` を使用して多角形エリア内のデバイス位置のリストを取得するための API リクエストです。この例では、`Polygon` 配列で定義されたエリア内に存在する 3 つのデバイス位置が返されます。

```
POST /tracking/v0/trackers/TrackerName/list-positions HTTP/1.1
Content-type: application/json

{
   "FilterGeometry": {
        "Polygon": [
          [
            [
              -123.12003339442259,
              49.27425121147397
            ],
            [
              -123.1176984148229,
              49.277063620879744
            ],
            [
              -123.12389509145294,
              49.277954183760926
            ],
            [
              -123.12755921328647,
```

```
              49.27554025235713
            ],
            [
              -123.12330236586217,
              49.27211836076236
            ],
            [
              -123.12003339442259,
              49.27425121147397
            ]
          ]
        ]
    },
   "MaxResults": 3,
   "NextToken": "1234-5678-9012"
}
```

以下に、ListDevicePositions のレスポンスの例を示します。

```
{
    "Entries": [
        {
            "DeviceId": "1",
            "SampleTime": "2022-10-24T19:09:07.327Z",
            "Position": [
                -123.12245146162303,
                49.27521118043802
            ],
            "Accuracy": {
                "Horizontal": 10
            },
            "PositionProperties": {
                "name": "device1"
            }
        },
        {
            "DeviceId": "3",
            "SampleTime": "2022-10-02T19:09:07.327Z",
            "Position": [
                -123.12325592118916,
                49.27340530543111
            ]
        },
```

```
        {
            "DeviceId": "2",
            "SampleTime": "2022-10-02T19:09:07.327Z",
            "Position": [
                -123.1230104928471,
                49.27752402723152
            ]
        }
    ],
   "NextToken": "1234-5678-9012"
}
```

CLI

[list-trackers](#)コマンドを実行します。

次の例は、多角形エリア内のデバイスのリストを取得 AWS CLI するための です。

```
aws location list-device-positions TODO: add arguments add props for filter geo
```

トラッカーをジオフェンスコレクションにリンクする

ジオフェンスコレクションとトラッカーを作成したので、これらをリンクして、位置情報の更新がすべてのジオフェンスに対して自動的に評価されるようにします。位置情報の更新をすべて評価したくない場合や、一部の位置情報をトラッカーリソースに保存していない場合は、オンデマンドで[ジオフェンスに対してデバイス位置を評価](#)することができます。

ジオフェンスに対してデバイス位置を評価すると、イベントが生成されます。これらのイベントにはアクションを設定できます。ジオフェンスイベントに設定できるアクションの詳細については、「Amazon [を使用した Amazon Location Service イベントへの対応 EventBridge](#)」を参照してください。

Amazon Location イベントには、そのイベントを生成するデバイス位置更新の属性と、出入りしたジオフェンスの属性の一部が含まれます。ジオフェンスイベントに含まれるデータについては、「[Amazon Location Service の Amazon EventBridge イベントの例](#)」を参照してください。

次の例では、コンソール、、 AWS CLIまたは Amazon Location APIs。

Console

Amazon Location Service コンソールを使用してトラッカーリソースをジオフェンスコレクションにリンクするには

1. Amazon Location Service コンソール (https://console.aws.amazon.com/location/) を開きます。
2. 左のナビゲーションペインから、[トラッカー] を選択します。
3. [デバイストラッカー] で、対象トラッカーの名前リンクを選択します。
4. [リンクされたジオフェンスコレクション] で [ジオフェンスコレクションをリンク] を選択します。
5. [リンクされたジオフェンスコレクション] ウィンドウのドロップダウンメニューからジオフェンスコレクションを選択します。
6. [Link (リンク)] を選択します。

トラッカーリソースをリンクすると、そのリソースには[アクティブ] ステータスが割り当てられます。

API

Amazon Location API を使用してトラッカーリソースをジオフェンスコレクションにリンクするには

Amazon Location Trackers API の `AssociateTrackerConsumer` オペレーションを使用してください。

次の例では、Amazon リソースネーム (ARN) を使用してジオフェンスコレクション*ExampleTracker*に関連付ける API リクエストを使用します。

```
POST /tracking/v0/trackers/ExampleTracker/consumers
Content-type: application/json

{
   "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/ExampleGeofenceCollection"
}
```

AWS CLI

AWS CLI コマンドを使用してトラッカーリソースをジオフェンスコレクションにリンクするには

associate-tracker-consumerコマンドを実行します。

次の例では、 を使用して AWS CLI 、 というジオフェンスコレクションを作成しますExampleGeofenceCollection。

```
aws location \
    associate-tracker-consumer \
        --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/ExampleGeofenceCollection" \
        --tracker-name "ExampleTracker"
```

ジオフェンスに対してデバイスの位置を評価する

ジオフェンスに対して位置を評価してジオフェンスイベントを生成する方法は 2 つあります。

- トラッカーとジオフェンスコレクションをリンクできます。詳細については、「トラッカーをジオフェンスコレクションにリンクする」セクションを参照してください。
- BatchEvaluateGeofences API を使用して、ジオフェンスコレクションリソースに直接リクエストして、1 つ以上の位置を評価できます。

さらに、ジオフェンス内でデバイスが入ってくる、終了する、またはアイドル状態のままの受信ジオフェンスイベントを予測できます。ForecastGeofenceEvents API を使用してイベントを予測します。

デバイスの位置履歴を追跡したり、地図上に位置を表示したりしたい場合は、トラッカーをジオフェンスコレクションとリンクしてください。あるいは、位置情報の更新をすべて評価したくない、または、位置データをトラッカーリソースに保存したくない場合もあるかもしれません。どちらかに当てはまる場合は、ジオフェンスコレクションに直接リクエストして、ジオフェンスに対して 1 つ以上のデバイスの位置を評価できます。

ジオフェンスに対してデバイス位置を評価すると、イベントが生成されます。これらのイベントに反応して、他の AWS サービスにルーティングできます。ジオフェンスイベントを受信するときに実行できるアクションの詳細については、「Amazon による Amazon Location Service イベントへの対応 EventBridge」を参照してください。

Amazon Location イベントには、そのイベントを生成するデバイス位置更新の属性 (タイムスタンプ、位置、精度、キーと値のペアから成るメタデータ、出入りしたジオフェンスの属性の一部など)

が含まれます。ジオフェンスイベントに含まれるデータについては、「Amazon Location Service の Amazon EventBridge イベントの例」を参照してください。

次の例では AWS CLI、、、または Amazon Location APIsを使用します。

API

Amazon Location API を使用して、デバイスの位置とジオフェンスの位置を比較して評価するには

Amazon Location Geofences API の BatchEvaluateGeofences オペレーションを使用してください。

次の例では、API リクエストを使用して、関連付けられたジオフェンスコレクション *ExampleGeofenceCollection*に対するデバイスの位置を評価します*ExampleDevice*。これらの値は、独自のジオフェンスとデバイス ID に置き換えてください。

```
POST /geofencing/v0/collections/ExampleGeofenceCollection/positions HTTP/1.1
Content-type: application/json

{
   "DevicePositionUpdates": [
      {
         "DeviceId": "ExampleDevice",
         "Position": [-123.123, 47.123],
         "SampleTime": "2021-11-30T21:47:25.149Z",
         "Accuracy": {
            "Horizontal": 10.30
         },
         "PositionProperties": {
            "field1": "value1",
            "field2": "value2"
         }
      }
   ]
}
```

AWS CLI

AWS CLI コマンドを使用してジオフェンスの位置と照らし合わせてデバイスの位置を評価するには

batch-evaluate-geofencesコマンドを実行します。

次の例では AWS CLI 、 を使用して、関連付けられたジオフェンスコレクション *ExampleDevice*に対して の位置を評価します*ExampleGeofenceCollection*。これらの値は、独自のジオフェンスとデバイス ID に置き換えてください。

```
aws location \
    batch-evaluate-geofences \
        --collection-name ExampleGeofenceCollection \
        --device-position-updates '[{"DeviceId":"ExampleDevice","Position":
[-123.123,47.123],"SampleTime":"2021-11-30T21:47:25.149Z","Accuracy":
{"Horizontal":10.30},"PositionProperties":{"field1":"value1","field2":"value2"}}]'
```

ジオフェンスに対してデバイス位置を評価すると、イベントが生成されます。従来、 を使用してイベントに反応することはできますがAmazon EventBridge、このプロセスでは、イベントが発生した後にのみイベントに反応できます。例えば、デバイスが境界を越えていて、その結果として別の規制の対象となる場合など、デバイスがジオフェンスに出入りするタイミングを予測する必要がある場合は、 ForecastGeofenceEvents API を使用して将来のジオフェンスイベントを予測できます。

ForecastGeofenceEvents API は、デバイスの time-to-breach、近接性、速度、位置などの基準を使用してイベントを予測します。この API は ForecastedBreachTime、ジオフェンスイベントが発生する推定時間を示す を返します。

次の例では、Amazon Location APIsを使用します。

API

Amazon Location APIs

Amazon Location Geofences API の ForecastGeofenceEvents オペレーションを使用してください。

次の例では、 API リクエストを使用して、 *ExampleDevice*に関連する のジオフェンスイベントを予測します*ExampleGeofence*。これらの値は、独自のジオフェンスとデバイス ID に置き換えてください。

```
POST /geofencing/v0/collections/CollectionName/forecast-geofence-events HTTP/1.1
Content-type: application/json

{
   "DeviceState": {
      "Position": [ number ],
      "Speed": number
```

```
   },
   "DistanceUnit": "string",
   "MaxResults": number,
   "NextToken": "string",
   "SpeedUnit": "string",
   "TimeHorizonMinutes": number
}
```

デバイスの位置を確認する

デバイスの位置の整合性を確認するには、 VerifyDevicePosition API を使用します。この API は、デバイスのセル信号、Wi-Fi アクセスポイント、Ipv4 アドレス、プロキシが使用されているかどうかなどのプロパティを評価することで、デバイスの位置の整合性に関する情報を返します。

前提条件

デバイス検証にリストされている APIs を使用する前に、次の前提条件を満たしていることを確認してください。

- チェックするデバイス用のトラッカーを作成しました。詳細については、「追跡を開始する」を参照してください。

次の例は、Amazon Location VerifyDevicePosition API のリクエストを示しています。

API

Amazon Location APIs を使用してデバイスの位置を確認するには

Amazon Location Tracking API の `VerifyDevicePosition`オペレーションを使用します。
APIs

次の例は、デバイスの位置の整合性を評価する API リクエストを示しています。これらの値を独自のデバイス IDs。

```
POST /tracking/v0/trackers/TrackerName/positions/verify HTTP/1.1
Content-type: application/json

{
   "DeviceState": {
      "Accuracy": {
         "Horizontal": number
```

```
      },
      "CellSignals": {
         "LteCellDetails": [
            {
               "CellId": number,
               "LocalId": {
                  "Earfcn": number,
                  "Pci": number
               },
               "Mcc": number,
               "Mnc": number,
               "NetworkMeasurements": [
                  {
                     "CellId": number,
                     "Earfcn": number,
                     "Pci": number,
                     "Rsrp": number,
                     "Rsrq": number
                  }
               ],
               "NrCapable": boolean,
               "Rsrp": number,
               "Rsrq": number,
               "Tac": number,
               "TimingAdvance": number
            }
         ]
      },
      "DeviceId": "ExampleDevice",
      "Ipv4Address": "string",
      "Position": [ number ],
      "SampleTime": "string",
      "WiFiAccessPoints": [
         {
            "MacAddress": "string",
            "Rss": number
         }
      ]
   },
   "DistanceUnit": "string"
}
```

> **Note**
>
> Location Integrity SDK は、デバイス検証に関連する機能を強化し、リクエストに応じて使用できます。SDK にアクセスするには、 セールスサポートにお問い合わせください。

Amazon での Amazon Location Service イベントへの対応 EventBridge

Amazon EventBridge は、Amazon Location. EventBridge receives などの AWS のサービスからのデータを使用して効率的にアプリケーションを接続し、そのデータを などのターゲットにルーティングするサーバーレスイベントバスですAWS Lambda。お客様は、データの送信先を判断するためのルーティングルールを設定して、すべてのデータソースにリアルタイムで反応するアプリケーションアーキテクチャを構築できます。

EventBridge デフォルトでは、ジオフェンスイベント (ENTERデバイスがジオフェンスエリアに出入りするときの および EXITイベント) のみが に送信されます。トラッカーリソースのすべてのフィルターされた位置更新のイベントを有効にできます。詳細については、「トラッカーの更新イベントを有効にします。」を参照してください。

詳細については、「Amazon EventBridge ユーザーガイド」の「イベントとイベントパターン」を参照してください。

トピック

- トラッカーの更新イベントを有効にします。
- Amazon Location のイベントルールを作成します。
- Amazon Location Service の Amazon EventBridge イベントの例

トラッカーの更新イベントを有効にします。

デフォルトでは、Amazon Location は ENTERおよび EXIT ジオフェンスイベントのみを に送信します EventBridge。トラッカーのすべてのフィルタリングされた位置UPDATEイベントを に送信できます EventBridge。これはトラッカーを作成または更新するときに設定できます。

例えば、 を使用して既存のトラッカーを更新するにはAWS CLI、次のコマンドを使用できます (の代わりにトラッカーリソースの名前を使用しますMyTracker) 。

```
aws location update-tracker --tracker-name MyTracker --event-bridge-enabled
```

トラッカーの位置イベントをオフにするには、API または Amazon Location Service コンソールを使用する必要があります。

Amazon Location のイベントルールを作成します。

のイベントバスごとに最大 300 のルールを作成して EventBridge 、Amazon Location イベントに応答して実行されるアクションを設定できます。

たとえば、ジオフェンスされた境界内で電話が検出されるとプッシュ通知が送信されるジオフェンスイベント用のルールを作成できます。

Amazon Location イベントのルールを作成する

次の値を使用して、Amazon Location イベントに基づいて EventBridge ルールを作成します。

- ルールタイプでは、イベントパターンを持つルールを選択します。
- イベントパターン ボックスに次のパターンを貼り付けます。

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"]
}
```

トラッカー位置更新ルールを作成するには、代わりに次のパターンを使用できます。

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Device Position Event"]
}
```

detailオプションでタグを追加することで ENTER または EXIT イベントのみを指定できます (ルールがトラッカーの位置更新の場合、 EventType は1つだけなので、フィルタリングする必要はありません) :

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"],
  "detail": {
    "EventType": ["ENTER"]
  }
```

```
}
```

オプションでポジションやジオフェンスのプロパティをフィルタリングすることもできます。

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"],
  "detail": {
    "EventType": ["ENTER"],
    "GeofenceProperties": {
      "Type": "LoadingDock"
    },
    "PositionProperties": {
      "VehicleType": "Truck"
    }
  }
}
```

- ターゲットを選択では、Amazon Location Service からイベントを受信したときに実行するターゲットアクションを選択します。

  例えば、Amazon Simple Notification Service (SNS) トピックを使用して、イベントが発生したときに E メールまたはテキストメッセージを送信できます。先ず、Amazon SNS コンソールを使用して Amazon SNS トピックを作成する必要があります。詳細については、「Amazon SNS を利用してユーザー通知」を参照してください。

> **Warning**
>
> イベントルールが正常に適用されたことを確認するのがベストプラクティスです。そうしないと、自動アクションが期待どおりに開始されない場合があります。イベントルールを確認するには、イベントルールの条件を設定します。たとえば、デバイスがジオフェンスエリアに入る様子をシミュレートします。

`detail-type`セクションを除外するだけで、Amazon Location からのすべてのイベントをキャプチャすることもできます。例:

```
{
  "source": [
    "aws.geo"
```

```
  ]
}
```

> **Note**
>
> 同じイベントが複数回配信される場合があります。イベント ID を使用して、受信したイベントの重複を排除できます。

Amazon Location Service の Amazon EventBridge イベントの例

`BatchUpdateDevicePosition`の呼び出しによって開始されるジオフェンスに入るイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "aa11aa22-33a-4a4a-aaa5-example",
  "detail-type": "Location Geofence Event",
  "source": "aws.geo",
  "account": "636103698109",
  "time": "2020-11-10T23:43:37Z",
  "region": "eu-west-1",
  "resources": [
    "arn:aws:geo:eu-west-1:0123456789101:geofence-collection/GeofenceEvents-
GeofenceCollection_EXAMPLE",
    "arn:aws:geo:eu-west-1:0123456789101:tracker/Tracker_EXAMPLE"
  ],
  "detail": {
    "EventType": "ENTER",
    "GeofenceId": "polygon_14",
    "DeviceId": "Device1-EXAMPLE",
    "SampleTime": "2020-11-10T23:43:37.531Z",
    "Position": [
      -123.12390073297821,
      49.23433613216247
    ],
    "Accuracy": {
      "Horizontal": 15.3
    },
    "GeofenceProperties": {
      "ExampleKey1": "ExampleField1",
      "ExampleKey2": "ExampleField2"
```

```
    },
    "PositionProperties": {
      "ExampleKey1": "ExampleField1",
      "ExampleKey2": "ExampleField2"
    }
  }
}
```

BatchUpdateDevicePositionの呼び出しによって開始されるジオフェンスを終了するためのイベントの例を以下に示します。

```
{
  "version": "0",
  "id": "aa11aa22-33a-4a4a-aaa5-example",
  "detail-type": "Location Geofence Event",
  "source": "aws.geo",
  "account": "123456789012",
  "time": "2020-11-10T23:41:44Z",
  "region": "eu-west-1",
  "resources": [
    "arn:aws:geo:eu-west-1:0123456789101:geofence-collection/GeofenceEvents-
GeofenceCollection_EXAMPLE",
    "arn:aws:geo:eu-west-1:0123456789101:tracker/Tracker_EXAMPLE"
  ],
  "detail": {
    "EventType": "EXIT",
    "GeofenceId": "polygon_10",
    "DeviceId": "Device1-EXAMPLE",
    "SampleTime": "2020-11-10T23:41:43.826Z",
    "Position": [
      -123.08569321875426,
      49.23766166742559
    ],
    "Accuracy": {
      "Horizontal": 15.3
    },
    "GeofenceProperties": {
      "ExampleKey1": "ExampleField1",
      "ExampleKey2": "ExampleField2"
    },
    "PositionProperties": {
      "ExampleKey1": "ExampleField1",
      "ExampleKey2": "ExampleField2"
```

```
    }
  }
}
```

BatchUpdateDevicePositionの呼び出しによって開始される位置更新イベントの例を以下に示します。

```
{
  "version": "0",
  "id": "aa11aa22-33a-4a4a-aaa5-example",
  "detail-type": "Location Device Position Event",
  "source": "aws.geo",
  "account": "123456789012",
  "time": "2020-11-10T23:41:44Z",
  "region": "eu-west-1",
  "resources": [
    "arn:aws:geo:eu-west-1:0123456789101:tracker/Tracker_EXAMPLE"
  ],
  "detail": {
    "EventType": "UPDATE",
    "TrackerName": "tracker_2",
    "DeviceId": "Device1-EXAMPLE",
    "SampleTime": "2020-11-10T23:41:43.826Z",
    "ReceivedTime": "2020-11-10T23:41:39.235Z",
    "Position": [
      -123.08569321875426,
      49.23766166742559
    ],
    "Accuracy": {
      "Horizontal": 15.3
    },
    "PositionProperties": {
      "ExampleKey1": "ExampleField1",
      "ExampleKey2": "ExampleField2"
    }
  }
}
```

Amazon Location Service による AWS IoT と MQTT を使ったトラッキング

MQTT は、制約のあるデバイス向けに設計された軽量で広く採用されているメッセージングプロトコルです。 は、MQTT プロトコルと MQTT over WebSocket Secure (WSS) プロトコルを使用するデバイス接続AWS IoT Coreをサポートしています。

AWS IoT Core はデバイスをAWSに接続し、デバイス間でメッセージの送受信を可能にする。AWS IoT Core ルールエンジンは、デバイスのメッセージトピックに関するクエリを保存し、Amazon Location Service などの他の AWS サービスにメッセージを送信するためのアクションを定義できるようにします。位置を座標として認識するデバイスは、ルールエンジンを通じて Amazon Location に位置情報を転送できます。

> **Note**
>
> デバイスは内蔵の GPS などを通じて自分の位置を知ることができます。 AWS IoT はサードパーティのデバイスの位置追跡もサポートしています。詳細については、AWS IoT 開発者ガイド (デベロッパーガイド) のAWS IoTコアデバイスのデータ を参照してください。

以下のウォークスルーでは、AWS IoT Coreルールを使ったトラッキングについて説明します。Amazon Location に送信する前に処理する必要がある場合は、デバイス情報を独自の AWS Lambda 機能に送信することもできます。Lambda を使用してデバイスの位置を処理する方法の詳細については、「MQTT AWS Lambda との併用」を参照してください。

トピック

- 前提条件
- AWS IoT Core ルールを作成します
- コンソールで AWS IoT Core ルールをテストします。
- MQTT AWS Lambda との併用

前提条件

開始する前に、以下の前提条件と設定を完了する必要があります。

- デバイスの位置データを送信するトラッカーリソースを作成します。
- トラッカーへのAWS IoT Coreアクセスを許可する IAM ロールを作成します。

これらの手順を実行するときは、以下のポリシーを使用してトラッカーへのアクセスを許可してください。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "WriteDevicePosition",
      "Effect": "Allow",
      "Action": "geo:BatchUpdateDevicePosition",
      "Resource": "arn:aws:geo:*:*:tracker/*"
    }
  ]
}
```

AWS IoT Core ルールを作成します

次に、デバイスの位置テレメトリを Amazon Location Service に転送する AWS IoT Core ルールを作成します。Lambda 関数 URL の詳細については、AWS IoT Core デベロッパーガイドの以下のトピックを参照してください。

- 「AWS IoTルールの作成」には、新しいルールの作成に関する情報が記載されています。
- ロケーションアクション Amazon Location に公開するためのルール作成に固有の情報

コンソールで AWS IoT Core ルールをテストします。

現在位置情報を含むテレメトリを公開しているデバイスがない場合は、AWS IoT Coreコンソールを使用してルールをテストすることができます。コンソールにはテストクライアントがあり、サンプルメッセージを公開してソリューションの結果を検証することができます。

1. https://console.aws.amazon.com/iot/ で AWS IoT Coreコンソールにサインインします。
2. 左側のナビゲーションメニューの テスト で、MQTT テストクライアント を選択します。
3. トピックに公開で、トピック名を *iot/topic* (異なる場合は AWS IoT Core ルールで設定したトピックの名前) に設定し、メッセージペイロードには次のように入力します。タイムスタンプ *1604940328* を過去 30 日以内の有効なタイムスタンプに置き換えます (30 日を超えるタイムスタンプは Amazon Location Service ストラッカーによって無視されます)。

```
{
  "payload": {
    "deviceid": "thing123",
    "timestamp": 1604940328,
    "location": { "lat": 49.2819, "long": -123.1187 },
    "accuracy": { "Horizontal": 20.5 },
    "positionProperties": { "field1": "value1", "field2": "value2" }
  }
}
```

4. MQTT メッセージを送信するには、トピックに発行 を選択します。
5. メッセージが Amazon Location Service によって受信されたことを確認するには、次の AWS CLI コマンドを使用します。セットアップ中に変更した場合は、トラッカー名を使用していたものに置き換えてください。

```
aws location batch-get-device-position --tracker-name MyTracker --device-ids
 thing123
```

MQTT AWS Lambda との併用

デバイスの位置情報をトラッキングのために Amazon Location に送信する場合、 AWS Lambda を使う必要はなくなったが、場合によってはまだLambdaを使いたいかもしれない。たとえば、デバイスの位置情報データを Amazon Location に送信する前に、ご自身で処理したい場合などです。以下のトピックでは、トラッカーに送信する前に、Lambda を使用してメッセージを処理する方法について説明します。このパターンの詳細については、リファレンスアーキテクチャをご覧ください。

トピック

- 前提条件
- Lambda 関数を作成する
- AWS IoT Core ルールを作成します
- コンソールで AWS IoT Core ルールをテストします。

前提条件

トラッキングを開始する前に、トラッカーリソースを作成する必要があります。トラッカーリソースを作成するには、Amazon Location コンソール、AWS CLI、または Amazon Location API を使用できます。

次の例では、Amazon Location Service コンソールを使用してトラッカーリソースを作成します。

1. Amazon Location Service コンソール (https://console.aws.amazon.com/location/) を開きます。
2. 左のナビゲーションペインから、[トラッカー] を選択します。
3. トラッカーを作成を選択します。
4. 次のボックスに入力します。

   - 名前 — 最大 100 文字のユニークな名前に入力します。有効な文字として英数字、ハイフン、ピリオド、アンダースコアを使用できます。例えば、 です*MyTracker*。
   - 説明 – 任意の説明を入力します。たとえば、*AWS IoT Core#########################*。
   - 位置フィルター — 位置の更新に使用するフィルターを選択します。たとえば、精度ベースのフィルタリングなどです。
5. トラッカーを作成を選択します。

Lambda 関数を作成する

AWS IoT Coreと Amazon Location Service との接続を確立するには、 AWS IoT Core から転送されたメッセージを処理する AWS Lambda 関数が必要です。この関数は、位置データを抽出し、Amazon Location Service 用にフォーマットし、Amazon Location トラッカー API を通じて送信します。この関数は AWS Lambda コンソールから作成することも、AWS Command Line Interface AWS CLI や AWS Lambda API を使用することもできます。

コンソールを使用して Amazon Location に位置更新を公開する Lambda 関数を作成するには:

1. AWS Lambda コンソールを https://console.aws.amazon.com/lambda/ で開きます。
2. ナビゲーションペインで、関数 を選択します。
3. 関数を作成を選択し、最初から作成するが選択されていることを確認します。
4. 次のボックスに入力します。

   - 関数の名前 に、 関数の名前を入力します。有効なエントリには、英数字、ハイフン、およびスペースなしのアンダースコアが含まれます。例えば、 です*MyLambda*。
   - ランタイム:*Python 3.8*を選択します。
5. 機能の作成を選択します。
6. JSON タブを選択して JSON エディタを開きます。

7. lambda_function.py のプレースホルダーコードを以下のように書き換え、 TRACKER_NAME に割り当てられた値を前提条件として作成したトラッカーの名前に置き換える。

```
from datetime import datetime
import json
import os

import boto3

# Update this to match the name of your Tracker resource
TRACKER_NAME = "MyTracker"

"""
This Lambda function receives a payload from AWS IoT Core and publishes device
 updates to
Amazon Location Service via the BatchUpdateDevicePosition API.

Parameter 'event' is the payload delivered from AWS IoT Core.

In this sample, we assume that the payload has a single top-level key 'payload' and
 a nested key
'location' with keys 'lat' and 'long'. We also assume that the name of the device
 is nested in
the payload as 'deviceid'. Finally, the timestamp of the payload is present as
 'timestamp'. For
example:

>>> event
{ 'payload': { 'deviceid': 'thing123', 'timestamp': 1604940328,
  'location': { 'lat': 49.2819, 'long': -123.1187 },
  'accuracy': {'Horizontal': 20.5 },
  'positionProperties': {'field1':'value1','field2':'value2'} }
}

If your data doesn't match this schema, you can either use the AWS IoT Core rules
 engine to
format the data before delivering it to this Lambda function, or you can modify the
 code below to
match it.
"""
def lambda_handler(event, context):
  update = {
      "DeviceId": event["payload"]["deviceid"],
```

```
      "SampleTime": datetime.fromtimestamp(event["payload"]
["timestamp"]).strftime("%Y-%m-%dT%H:%M:%SZ"),
      "Position": [
        event["payload"]["location"]["long"],
        event["payload"]["location"]["lat"]
        ]
    }
  if "accuracy" in event["payload"]:
      update["Accuracy"] = event["payload"]['accuracy']
  if "positionProperties" in event["payload"]:
      update["PositionProperties"] = event["payload"]['positionProperties']

  client = boto3.client("location")
  response = client.batch_update_device_position(TrackerName=TRACKER_NAME,
 Updates=[update])

  return {
    "statusCode": 200,
    "body": json.dumps(response)
  }
```

8. デプロイ を選択して、関数を更新します。
9. 設定タブを選択します。
10. 権限セクションで、ハイパーリンク付きのロール名を選択し、Lambda 関数に Amazon Location Service アクセス権限を付与します。
11. ロールの概要ページからアクセス権限の追加を選択し、ドロップダウンリストからインラインポリシーの作成を選択します。
12. JSON タブを選択して、次の JSON ポリシードキュメントを入力します。これにより、Lambda 関数は、すべてのリージョンのすべてのトラッカーリソースによって管理されているデバイスの位置の更新ができます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "WriteDevicePosition",
      "Effect": "Allow",
      "Action": "geo:BatchUpdateDevicePosition",
      "Resource": "arn:aws:geo:*:*:tracker/*"
    }
  ]
```

```
}
```

13. ポリシーの確認を選択します。
14. ポリシー名を入力します。例えば、 です*AmazonLocationTrackerWriteOnly*。
15. ポリシーを作成を選択します。

この関数コードは必要に応じて変更して、独自のデバイスメッセージスキーマに適合させることができます。

AWS IoT Core ルールを作成します

次に、デバイスの位置テレメトリを AWS Lambda ファンクションに転送し、変換して Amazon Location Service に公開する AWS IoT Core ルールを作成する。提供されているサンプルルールは、デバイスペイロードの必要な変換がすべて Lambda 関数によって処理されることを前提としています。このルールは、AWS IoT Coreコンソール、AWS Command Line Interface AWS CLI、または AWS IoT Core API を使用して作成できます。

> **Note**
>
> AWS IoTLambda関数を呼び出すために AWS IoT Core に必要なパーミッションは AWS IoT コンソールで処理されますが、 AWS CLI やSDKからルールを作成する場合は、 にパーミッションを付与するポリシーを設定する必要があります。

コンソールを使用して を作成するには

1. https://console.aws.amazon.com/iot/ で AWS IoT Coreコンソールにサインインします。
2. 左のナビゲーションペインで、ACTを拡張し、ルール の順に選択します。
3. ルールを作成を選択して新しいルールウィザードを起動します。
4. ルールの名前と説明を入力します。
5. ルールクエリステートメントでは、 FROM 属性を更新して、少なくとも 1 つのデバイスが位置を含むテレメトリを公開しているトピックを参照する。ソリューションをテストする場合、変更は必要ありません。

   ```
   SELECT * FROM 'iot/topic'
   ```

6. 1 つ以上のアクションを設定する で、アクションの追加 を選択します。

7. Lambda 関数にメッセージを送信するを選択します。
8. アクションの設定を選択します。
9. リストから Lambda 関数を選択します。
10. Add actionを選択します。
11. ルールの作成を選択します。

コンソールで AWS IoT Core ルールをテストします。

現在位置情報を含むテレメトリを公開しているデバイスがない場合は、AWS IoT Coreコンソールを使用してルールとこのソリューションをテストすることができます。コンソールにはテストクライアントがあり、サンプルメッセージを公開してソリューションの結果を検証することができます。

1. https://console.aws.amazon.com/iot/ で AWS IoT Coreコンソールにサインインします。
2. 左側のナビゲーションメニューの テスト で、MQTT テストクライアント を選択します。
3. トピックに公開で、トピック名を *iot/topic* (異なる場合は AWS IoT Core ルールで設定したトピックの名前) に設定し、メッセージペイロードには次のように入力します。タイムスタンプ *1604940328* を過去 30 日間の有効なタイムスタンプに置き換えます (30 日を超えるタイムスタンプは無視されます)。

```
{
  "payload": {
    "deviceid": "thing123",
    "timestamp": 1604940328,
    "location": { "lat": 49.2819, "long": -123.1187 },
    "accuracy": { "Horizontal": 20.5 },
    "positionProperties": { "field1": "value1", "field2": "value2" }
  }
}
```

4. MQTT メッセージを送信するには、トピックに発行 を選択します。
5. メッセージが Amazon Location Service によって受信されたことを確認するには、次の AWS CLI コマンドを使用します。セットアップ中に変更した場合は、トラッカー名とデバイス ID を使用していたものに置き換えてください。

```
aws location batch-get-device-position --tracker-name MyTracker --device-ids
 thing123
```

ジオフェンスコレクションリソースの管理

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用してジオフェンスコレクションを管理します。

ジオフェンスコレクションリソースを一覧表示します

ジオフェンスコレクションリストは、Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して表示できます。

Console

Amazon Location コンソールを使用してジオフェンスコレクションのリストを表示するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左側のナビゲーションペインから、[ジオフェンスコレクション] を選択します。
3. [マイジオフェンスコレクション] にジオフェンスコレクションのリストが表示されます。

API

Amazon Location Geofences API の `ListGeofenceCollections` オペレーションを使用してください。

以下の例は、AWS アカウント内のジオフェンスコレクションのリストを取得する API リクエストです。

```
POST /geofencing/v0/list-collections
```

以下に、`ListGeofenceCollections` のレスポンスの例を示します。

```
{
    "Entries": [
    {
        "CollectionName": "ExampleCollection",
        "CreateTime": 2020-09-30T22:59:34.142Z,
        "Description": "string",
        "UpdateTime": 2020-09-30T23:59:34.142Z
    },
    "NextToken": "1234-5678-9012"
```

```
}
```

CLI

[list-geofence-collections](#)コマンドを実行します。

次の例は、AWS アカウント内のジオフェンスコレクションのリストを取得するための AWS CLI です。

```
aws location list-geofence-collections
```

ジオフェンスコレクションの詳細を取得する

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、AWSアカウント内のジオフェンスコレクションリソースに関する詳細を取得できます。

Console

Amazon Location コンソールを使用してジオフェンスコレクションの詳細を表示するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左側のナビゲーションペインから、[ジオフェンスコレクション] を選択します。
3. [マイジオフェンスコレクション] で、対象のジオフェンスコレクションの名前リンクを選択します。

API

Amazon Location Geofences API の [DescribeGeofenceCollection](#) オペレーションを使用してください。

次の例は、 のジオフェンスコレクションの詳細を取得するための API リクエストです*ExampleCollection*。

```
GET /geofencing/v0/collections/ExampleCollection
```

以下に、DescribeGeofenceCollection のレスポンスの例を示します。

```
{
```

```
    "CollectionArn": "arn:aws:geo:us-west-2:123456789012:geofence-collection/
GeofenceCollection",
    "CollectionName": "ExampleCollection",
    "CreateTime": 2020-09-30T22:59:34.142Z,
    "Description": "string",
    "KmsKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
    "Tags": {
        "Tag1" : "Value1"
    },
    "UpdateTime": 2020-09-30T23:59:34.142Z
}
```

CLI

describe-geofence-collectionコマンドを実行します。

次の例は、 のジオフェンスコレクションの詳細を取得AWS CLIするための です*ExampleCollection*。

```
aws location describe-geofence-collection \
    --collection-name "ExampleCollection"
```

ジオフェンスコレクションを削除する

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、AWS アカウントからジオフェンスコレクションを削除できます。

Console

Amazon Location コンソールを使用してジオフェンスコレクションを削除するには

> ⚠ Warning
> このオペレーションはリソースを完全に削除します。

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左側のナビゲーションペインから、[ジオフェンスコレクション] を選択します。
3. [マイジオフェンスコレクション] で、対象のジオフェンスコレクションを選択します。

4. [ジオフェンスコレクションを削除] を選択します。

API

Amazon Location API DeleteGeofenceCollection のオペレーションを使用してください。

次の例は、ジオフェンスコレクション を削除する API リクエストです*ExampleCollection*。

```
DELETE /geofencing/v0/collections/ExampleCollection
```

以下に、DeleteGeofenceCollection のレスポンスの例を示します。

```
HTTP/1.1 200
```

CLI

delete-geofence-collectionコマンドを実行します。

次の例は、ジオフェンスコレクション を削除する AWS CLI コマンドで
す*ExampleCollection*。

```
aws location delete-geofence-collection \
    --collection-name "ExampleCollection"
```

保存されているジオフェンスを一覧表示する

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、指定したジオフェンスコレクションに保存されているジオフェンスを一覧表示できます。

Console

Amazon Location コンソールを使用してジオフェンスのリストを表示するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左側のナビゲーションペインから、[ジオフェンスコレクション] を選択します。
3. [マイジオフェンスコレクション] で、対象のジオフェンスコレクションの名前リンクを選択します。

4. [ジオフェンス] で、ジオフェンスコレクション内のジオフェンスを表示します。

API

Amazon Location Geofences API の ListGeofences オペレーションを使用してください。

次の例は、ジオフェンスコレクション に保存されているジオフェンスのリストを取得する API リクエストです*ExampleCollection*。

```
POST /geofencing/v0/collections/ExampleCollection/list-geofences
```

以下に、ListGeofences のレスポンスの例を示します。

```
{
   "Entries": [
      {
         "CreateTime": 2020-09-30T22:59:34.142Z,
         "GeofenceId": "geofence-1",
         "Geometry": {
             "Polygon": [
                 [-5.716667, -15.933333,
                 [-14.416667, -7.933333],
                 [-12.316667, -37.066667],
                 [-5.716667, -15.933333]
             ]
         },
         "Status": "ACTIVE",
         "UpdateTime": 2020-09-30T23:59:34.142Z
      }
   ],
   "NextToken": "1234-5678-9012"
}
```

CLI

list-geofencesコマンドを実行します。

次の例はAWS CLI、ジオフェンスコレクション に保存されているジオフェンスのリストを取得する です*ExampleCollection*。

```
aws location list-geofences \
    --collection-name "ExampleCollection"
```

ジオフェンスの詳細を取得する

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、ジオフェンスコレクションから作成時間、更新時間、ジオメトリ、ステータスなどの特定のジオフェンスの詳細を取得できます。

Console

Amazon Location コンソールを使用してジオフェンスのステータスを表示するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左側のナビゲーションペインから、[ジオフェンスコレクション] を選択します。
3. [マイジオフェンスコレクション] で、対象のジオフェンスコレクションの名前リンクを選択します。
4. [ジオフェンス] で、ジオフェンスの状態を表示できます。

API

Amazon Location Geofences API の `GetGeofence` オペレーションを使用してください。

次の例は、ジオフェンスコレクション からジオフェンスの詳細を取得するための API リクエストです*ExampleCollection*。

```
GET /geofencing/v0/collections/ExampleCollection/geofences/ExampleGeofence1
```

以下に、GetGeofence のレスポンスの例を示します。

```
{
   "CreateTime": 2020-09-30T22:59:34.142Z,
   "GeofenceId": "ExampleGeofence1",
   "Geometry": {
      "Polygon": [
          [-1,-1],
          [1,-1],
          [0,1],
          [-1,-1]
      ]
   },
   "Status": "ACTIVE",
   "UpdateTime": 2020-09-30T23:59:34.142Z
```

```
}
```

CLI

get-geofenceコマンドを実行します。

次の例は、 のジオフェンスコレクションの詳細を取得AWS CLIするための です*ExampleCollection*。

```
aws location get-geofence \
    --collection-name "ExampleCollection" \
    --geofence-id "ExampleGeofence1"
```

ジオフェンスを削除する

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、ジオフェンスコレクションからジオフェンスを削除できます。

Console

Amazon Location コンソールを使用してジオフェンスを削除するには

> **Warning**
> このオペレーションはリソースを完全に削除します。

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/)。
2. 左側のナビゲーションペインから、[ジオフェンスコレクション] を選択します。
3. [マイジオフェンスコレクション] で、対象のジオフェンスコレクションの名前リンクを選択します。
4. [ジオフェンス] で、ターゲットジオフェンスを選択します。
5. [ジオフェンスを削除] を選択します。

API

Amazon Location Geofences API の BatchDeleteGeofence オペレーションを使用してください。

次の例は、ジオフェンスコレクション からジオフェンスを削除する API リクエストです*ExampleCollection*。

```
POST /geofencing/v0/collections/ExampleCollection/delete-geofences
Content-type: application/json

{
   "GeofenceIds": [ "ExampleGeofence11" ]
}
```

以下に、BatchDeleteGeofence の正常なレスポンスの例を示します。

```
HTTP/1.1 200
```

CLI

batch-delete-geofenceコマンドを実行します。

次の例は、ジオフェンスコレクション からジオフェンスを削除する AWS CLI コマンドです*ExampleCollection*。

```
aws location batch-delete-geofence \
    --collection-name "ExampleCollection" \
    --geofence-ids "ExampleGeofence11"
```

トラッカーリソースの管理

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用してトラッカーを管理できます。

トラッカーリストを表示

トラッカーリストは、Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して表示できます。

Console

Amazon Location コンソールを使用して既存のトラッカーリストを表示するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/) 。

2. 左側のナビゲーションペインから、[トラッカー] を選択します。
3. [マイトラッカー] でトラッカーリソースのリストを表示します。

API

Amazon Location Trackers API の ListTrackers オペレーションを使用してください。

以下の例は、AWS アカウント内のトラッカーリストを取得するための API リクエストです。

```
POST /tracking/v0/list-trackers
```

以下に、ListTrackers のレスポンスの例を示します。

```
{
   "Entries": [
      {
         "CreateTime": 2020-10-02T19:09:07.327Z,
         "Description": "string",
         "TrackerName": "ExampleTracker",
         "UpdateTime": 2020-10-02T19:10:07.327Z
      }
   ],
   "NextToken": "1234-5678-9012"
}
```

CLI

list-trackersコマンドを実行します。

以下の例では、AWS CLI を使用してAWS アカウント内のトラッカーリストを取得します。

```
aws location list-trackers
```

トラッカーをジオフェンスコレクションから切り離す

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、トラッカーをジオフェンスコレクションから切り離すことができます。

Console

Amazon Location コンソールを使用して、トラッカーとジオフェンスコレクションの関連付けを解除するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/) 。
2. 左側のナビゲーションペインから、[トラッカー] を選択します。
3. [マイトラッカー] で、対象トラッカーの名前リンクを選択します。
4. [リンクされたジオフェンスコレクション] で、ステータスが [リンク済み] のジオフェンスコレクションを選択します。
5. [リンクの解除] を選択します。

API

Amazon Location Trackers API の DisassociateTrackerConsumer オペレーションを使用してください。

以下の例は、トラッカーとジオフェンスコレクションの関連付けを解除ための API リクエストです。

```
DELETE /tracking/v0/trackers/ExampleTracker/consumers/arn:aws:geo:us-
west-2:123456789012:geofence-collection/ExampleCollection
```

以下に、DisassociateTrackerConsumer のレスポンスの例を示します。

```
HTTP/1.1 200
```

CLI

disassociate-tracker-consumerコマンドを実行します。

次の例は、トラッカーとジオフェンスコレクションの関連付けを解除するための AWS CLI コマンドです。

```
aws location disassociate-tracker-consumer \
    --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-collection/
ExampleCollection" \
    --tracker-name "ExampleTracker"
```

トラッカーの詳細を取得します。

AWS アカウント内のトラッカーに関する詳細情報を取得するには、Amazon Location コンソール、AWS CLI、または Amazon Location API を使用します。

Console

Amazon Location コンソールを使用してトラッカーの詳細を表示するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/) 。
2. 左側のナビゲーションペインから、[トラッカー] を選択します。
3. [マイトラッカー] で、対象トラッカーの名前リンクを選択します。
4. 「情報」にトラッカーの詳細が表示されます。

API

Amazon Location Tracker API の `DescribeTracker` オペレーションを使用してください。

次の例は、 のトラッカーの詳細を取得するための API リクエストです*ExampleTracker*。

```
GET /tracking/v0/trackers/ExampleTracker
```

以下に、`DescribeTracker` のレスポンスの例を示します。

```
{
   "CreateTime": 2020-10-02T19:09:07.327Z,
   "Description": "string",
   "EventBridgeEnabled": false,
   "KmsKeyId": "1234abcd-12ab-34cd-56ef-1234567890ab",
   "PositionFiltering": "TimeBased",
   "Tags": {
      "Tag1" : "Value1"
   },
   "TrackerArn": "arn:aws:geo:us-west-2:123456789012:tracker/ExampleTracker",
   "TrackerName": "ExampleTracker",
   "UpdateTime": 2020-10-02T19:10:07.327Z
}
```

CLI

`describe-tracker`コマンドを実行します。

次の例は、 のトラッカーの詳細を取得するための AWS CLI コマンドです*ExampleTracker*。

```
aws location describe-tracker \
    --tracker-name "ExampleTracker"
```

トラッカーを削除する

AWS アカウントからトラッカーを削除するには、Amazon Location コンソール、AWS CLI、または Amazon Location API を使用します。

Console

Amazon Location コンソールを使用して既存のマップリソースを削除するには

> **Warning**
>
> このオペレーションはリソースを完全に削除します。トラッカーリソースが使用されている場合、エラーが発生する可能性があります。対象リソースがアプリケーションに使用されていないことを確認してください。

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/) 。
2. 左側のナビゲーションペインから、[トラッカー] を選択します。
3. [マイトラッカー] で、対象トラッカーを選択します。
4. [トラッカーを削除] を選択します。

API

Amazon Location Tracker API の `DeleteTracker` オペレーションを使用してください。

次の例は、トラッカー を削除する API リクエストです*ExampleTracker*。

```
DELETE /tracking/v0/trackers/ExampleTracker
```

以下に、`DeleteTracker` のレスポンスの例を示します。

```
HTTP/1.1 200
```

CLI

delete-trackerコマンドを実行します。

次の例は、トラッカー を削除する AWS CLI コマンドです*ExampleTracker*。

```
aws location delete-tracker \
    --tracker-name "ExampleTracker"
```

サンプルジオフェンシングと追跡モバイルアプリケーション

このトピックでは、モバイルアプリケーションで Amazon Location ジオフェンスとトラッカーを使用する主な機能を示すために設計されたチュートリアルについて説明します。アプリケーションは、トラッカーとジオフェンスが Lambda AWS IoT と Amazon Location の機能を組み合わせてどのように相互作用するかを示します。チュートリアルは 2 つあります。

- Android 用のサンプル追跡およびジオフェンシングアプリケーション。 GitHubhttps://github.com/aws-geospatial/amazon-location-samples-android/tree/main/tracking-with-geofence-notifications からプロジェクトファイルのクローンを作成できます。
- iOS 用のサンプル追跡およびジオフェンシングアプリケーション。 GitHubhttps://github.com/aws-geospatial/amazon-location-samples-ios/tree/main/tracking-with-geofence-notifications からプロジェクトファイルのクローンを作成できます。

Android 用のサンプル追跡およびジオフェンスアプリケーション

このトピックでは、モバイルアプリケーションで Amazon Location ジオフェンスとトラッカーを使用する主な機能を示すために設計された Android チュートリアルについて説明します。アプリケーションは、トラッカーとジオンフェンスが Lambda AWS IoT と Amazon Location の機能を組み合わせてどのように相互作用するかを示します。

トピック

- アプリケーションの Amazon Location リソースを作成する
- ジオフェンスコレクションを作成する
- トラッカーをジオフェンスコレクションにリンクする
- MQTT での AWS Lambda の使用
- サンプルアプリケーションコードを設定する

- サンプルアプリケーションの使用

アプリケーションの Amazon Location リソースを作成する

開始するには、必要な Amazon Location リソースを作成する必要があります。これらのリソースは、アプリケーションの機能や提供されたコードスニペットの実行に不可欠です。

> **Note**
>
> AWS アカウントを作成していない場合は、AWS アカウント管理ユーザーガイドの指示に従ってください。

開始するには、Amazon Cognito ID プール ID を作成する必要があります。次の手順を使用します。

1. Amazon Cognito コンソールを開き、左側のメニューからアイデンティティプールを選択し、アイデンティティプールの作成を選択します。
2. ゲストアクセスがチェックされていることを確認し、次へを押して続行します。
3. 次に、新しい IAM ロールを作成するか、既存の IAM ロールを使用します。
4. ID プール名を入力し、ID プールが、次の手順で作成するマップとトラッカーの Amazon Location (geo)リソースにアクセスできることを確認します。

次に、 AWS Amazon Location コンソールでマップを作成してスタイル設定する必要があります。次の手順を使用します。

1. Amazon Location コンソールのマップセクションに移動し、マップの作成 を選択します。
2. 新しいマップに名前 と説明 を付けます。割り当てた名前は、チュートリアルの後半で使用するため、記録します。
3. マップスタイルを選択するときは、マップデータプロバイダーを検討してください。詳細については、AWS サービス条件のセクション 82 を参照してください。
4. Amazon Location 利用規約 に同意し、マップの作成 を選択してマップ作成プロセスを完了します。

次に、Amazon Location コンソールでトラッカーを作成する必要があります。次の手順を使用します。

1. Amazon Location コンソールでマップセクションを開きます。
2. トラッカーを作成を選択します。
3. 必須フィールドに入力します。トラッカーの名前は、このチュートリアル全体で繰り返されるため、書き留めておきます。
4. 位置フィルタリング フィールドで、トラッカーリソースの使用方法に最も適したオプションを選択します。位置フィルタリングを設定しない場合、デフォルト設定は ですTimeBased。詳細については、「Amazon Location API リファレンス」の「トラッカーPositionFiltering」、「」を参照してください。
5. トラッカーの作成を選択して、トラッカーの作成を完了します。

ジオフェンスコレクションを作成する

次に、ジオフェンスコレクションを作成します。コンソール、API、または CLI のいずれかを使用できます。次の手順では、各オプションについて説明します。

- Amazon Location コンソールを使用してジオフェンスコレクションを作成します。
  1. Amazon Location コンソールのジオフェンスコレクションセクションを開きます。
  2. [ジオフェンスコレクションの作成] を選択します。
  3. コレクションの名前と説明を入力します。
  4. をターゲット Amazon CloudWatch とする EventBridge ルールでは、オプションの EventBridge ルールを作成して、ジオフェンスイベントへの対応を開始できます。これにより、Amazon Location は にイベントを発行できます Amazon CloudWatch Logs。
  5. ジオフェンスコレクションの作成を押して、コレクションの作成を完了します。
- Amazon Location API を使用してジオフェンスコレクションを作成します。

  Amazon Location Geofences API の CreateGeofenceCollectionオペレーションを使用します。APIs 次の例では、 API リクエストを使用して、 というジオフェンスコレクションを作成しますGEOCOLLECTION_NAME。

```
POST /geofencing/v0/collections
Content-type: application/json

{
   "CollectionName": "GEOCOLLECTION_NAME",
   "Description": "Geofence collection 1 for shopping center",
   "Tags": {
```

```
      "Tag1" : "Value1"
   }
}
```

- AWS CLI コマンドを使用してジオフェンスコレクションを作成します。

  `create-geofence-collection`コマンドを実行します。次の例では、 AWS CLI を使用して、というジオフェンスコレクションを作成します*GEOCOLLECTION_NAME*。 AWS CLI の使用の詳細については、AWS 「 コマンドラインインターフェイスのドキュメント」を参照してください。

```
aws location \
  create-geofence-collection \
  --collection-name "ExampleGeofenceCollection" \
  --description "Shopping center geofence collection" \
  --tags Tag1=Value1
```

トラッカーをジオフェンスコレクションにリンクする

トラッカーをジオフェンスコレクションにリンクするには、コンソール、API、または CLI のいずれかを使用できます。次の手順では、各オプションについて説明します。

Amazon Location Service コンソールを使用して、トラッカーリソースをジオフェンスコレクションにリンクします。

1. Amazon Location コンソールを開きます。
2. 左のナビゲーションペインから、[トラッカー] を選択します。
3. Device Trackers で、ターゲットトラッカーの名前リンクを選択します。
4. [リンクされたジオフェンスコレクション] で [ジオフェンスコレクションをリンク] を選択します。
5. [リンクされたジオフェンスコレクション] ウィンドウのドロップダウンメニューからジオフェンスコレクションを選択します。
6. [Link (リンク)] を選択します。
7. トラッカーリソースをリンクすると、そのリソースには[アクティブ] ステータスが割り当てられます。

Amazon Location APIs。

Amazon Location Trackers API の AsssociateTrackerConsumer オペレーションを使用してください。次の例では、Amazon リソースネーム (ARN) を使用して をジオフェンスコレクションExampleTrackerに関連付ける API リクエストを使用します。

```
        POST /tracking/v0/trackers/ExampleTracker/consumers
        Content-type: application/json
        {
           "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GOECOLLECTION_NAME"
            }
```

AWS CLI コマンドを使用してトラッカーリソースをジオフェンスコレクションにリンクします。

associate-tracker-consumer コマンドを実行します。次の例では、 AWS CLI を使用して、というジオフェンスコレクションを作成しますGOECOLLECTION_NAME。

```
aws location \
associate-tracker-consumer \
    --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GOECOLLECTION_NAME" \
    --tracker-name "ExampleTracker"
```

MQTT での AWS Lambda の使用

AWS IoT と Amazon Location 間の接続を作成するには、 EventBridge CloudWatch イベントによって転送されるメッセージを処理するための Lambda 関数が必要です。この関数は、位置データを抽出し、Amazon Location 用にフォーマットして、Amazon Location Tracker API を通じて送信します。

次の手順は、Lambda コンソールを使用してこの関数を作成する方法を示しています。

1. コンソールを開きます。
2. ナビゲーションペインで、関数 を選択します。
3. 次に、関数の作成 を選択し、作成者を最初から作成する オプションが選択されていることを確認します。
4. 関数名 を指定し、ランタイムオプションで Node.js 16.x を選択します。

5. [関数を作成] を選択します。
6. コードタブを開いてエディタにアクセスします。
7. ファイル内のプレースホルダーコードを次のように上書きしますindex.js。

```
            const AWS = require('aws-sdk')
            const iot = new AWS.Iot();
            exports.handler =  function(event) {
              console.log("event===>>>", JSON.stringify(event));
              var param = {
                endpointType: "iot:Data-ATS"
              };
              iot.describeEndpoint(param, function(err, data) {
                if (err) {
                  console.log("error===>>>", err, err.stack); // an error occurred
                } else {
                  var endp = data['endpointAddress'];
                  const iotdata = new AWS.IotData({endpoint: endp});
                  const trackerEvent = event["detail"]["EventType"];
                  const src = event["source"];
                  const time = event["time"];
                  const gfId = event["detail"]["GeofenceId"];
                  const resources = event["resources"][0];
                  const splitResources = resources.split(".");
                  const geofenceCollection = splitResources[splitResources.length -
1];
                  const coordinates = event["detail"]["Position"];

                  const deviceId = event["detail"]["DeviceId"];
                  console.log("deviceId===>>>", deviceId);
                  const msg =  {
                      "trackerEventType" : trackerEvent,
                      "source" : src,
                      "eventTime" : time,
                      "geofenceId" : gfId,
                      "coordinates": coordinates,
                      "geofenceCollection": geofenceCollection
                    };
                  const params = {
                    topic: `${deviceId}/tracker`,
                    payload: JSON.stringify(msg),
                    qos: 0
                  };
```

```
                    iotdata.publish(params, function(err, data) {
                        if (err) {
                          console.log("error===>>>", err, err.stack); // an error
occurred
                        } else {
                          console.log("Ladmbda triggered===>>>", trackerEvent);  //
successful response
                        }
                    });
                }
            });
        }
```

8. デプロイを押して、更新された関数を保存します。
9. 次に、設定タブを開きます。
10. トリガー セクションで、トリガーを追加 ボタンを押します。
11. ソースフィールドで EventBridge（ CloudWatch イベント ）を選択します。
12. 既存のルールオプションを選択します。
13. ルール名を入力します。例: AmazonLocationMonitor-GEOFENCECOLLECTION_NAME。
14. 追加ボタンを押します。
15. これにより、アクセス許可タブにリソースベースのポリシーステートメントもアタッチされます。

次に、 を使用して MQTT テストクライアントをセットアップします。次の手順 AWS IoTを使用します。

1. https://console.aws.amazon.com/iot/ を開きます。
2. 左側のナビゲーションペインで、MQTT テストクライアント を選択します。
3. MQTT 接続を設定できる MQTT テストクライアントというセクションが表示されます。
4. 必要な設定を行ったら、Connect ボタンをクリックして、指定されたパラメータを使用して MQTT ブローカーへの接続を確立します。
5. エンドポイントはチュートリアルの後半で使用するため、記録します。

   テストクライアントに接続したら、MQTT テストクライアントインターフェイスで提供されるそれぞれの入力フィールドを使用して、MQTT トピックをサブスクライブしたり、トピックにメッセージを発行したりできます。次に、 AWS IoT ポリシーを作成します。

6. 左側のメニューの「セキュリティの管理」で「セキュリティの管理」オプションを選択し、「ポリシー」をクリックします。
7. ポリシーの作成 ボタンをクリックします。
8. ポリシー名を入力します。
9. ポリシードキュメントで JSON タブを選択します。
10. 以下に示すポリシーをコピーして貼り付けます。ただし、必ず *REGION*と ですべての要素を更新してください*ACCOUNT_ID*。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:REGION:ACCOUNT_ID:client/${cognito-
identity.amazonaws.com:sub}",
        "arn:aws:iot:REGION:ACCOUNT_ID:topic/${cognito-
identity.amazonaws.com:sub}",
        "arn:aws:iot:REGION:ACCOUNT_ID:topicfilter/${cognito-
identity.amazonaws.com:sub}/*",
        "arn:aws:iot:REGION:ACCOUNT_ID:topic/${cognito-identity.amazonaws.com:sub}/
tracker"
      ],
      "Effect": "Allow"
    }
  ]
}
```

11. 作成ボタンを選択して終了します。

前の手順を完了したら、次のようにゲストロールのアクセス許可を更新します。

1. Amazon Cognito に移動し、ID プールを開きます。次に、ユーザーアクセスに進み、ゲストロールを選択します。
2. アクセス許可ポリシーをクリックして編集を有効にします。

```
{
    'Version': '2012-10-17',
    'Statement': [
        {
            'Action': [
                'geo:GetMap*',
                'geo:BatchUpdateDevicePosition',
                'geo:BatchEvaluateGeofences',
                'iot:Subscribe',
                'iot:Publish',
                'iot:Connect',
                'iot:Receive',
                'iot:AttachPrincipalPolicy',
                'iot:AttachPolicy',
                'iot:DetachPrincipalPolicy',
                'iot:DetachPolicy'
            ],
            'Resource': [
                'arn:aws:geo:us-east-1:{USER_ID}:map/{MAP_NAME}',
                'arn:aws:geo:us-east-1:{USER_ID}:tracker/{TRACKER_NAME}',
                'arn:aws:geo:us-east-1:{USER_ID}:geofence-collection/
{GEOFENCE_COLLECTION_NAME}',
                'arn:aws:iot:us-east-1:{USER_ID}:client/${cognito-
identity.amazonaws.com:sub}',
                'arn:aws:iot:us-east-1:{USER_ID}:topic/${cognito-
identity.amazonaws.com:sub}',
                'arn:aws:iot:us-east-1:{USER_ID}:topicfilter/${cognito-
identity.amazonaws.com:sub}/*',
                'arn:aws:iot:us-east-1:{USER_ID}:topic/${cognito-
identity.amazonaws.com:sub}/tracker'
            ],
            'Effect': 'Allow'
        },
        {
            'Condition': {
                'StringEquals': {
                    'cognito-identity.amazonaws.com:sub': '${cognito-
identity.amazonaws.com:sub}'
                }
            },
            'Action': [
                'iot:AttachPolicy',
                'iot:DetachPolicy',
```

```
                'iot:AttachPrincipalPolicy',
                'iot:DetachPrincipalPolicy'
            ],
            'Resource': [
                '*'
            ],
            'Effect': 'Allow'
        }
    ]
}
```

3. 上記のポリシーの変更により、アプリケーションに必要なすべての AWS リソースが適切に設定されるようになりました。

サンプルアプリケーションコードを設定する

1. https://github.com/aws-geospatial/amazon-location-samples-android/tree/main/ tracking-with-geofence-notificationsというリポジトリをローカルマシンにクローンします。
2. Android Studio でAmazonSampleSDKAppプロジェクトを開きます。
3. Android デバイスまたはエミュレータでアプリを構築して実行します。

サンプルアプリケーションの使用

サンプルを使用するには、次の手順に従います。

- を作成しますcustom.properties。

  custom.properties ファイルを設定するには、次の手順に従います。

  1. 任意のテキストエディタまたは IDE を開きます。
  2. 新しいファイルを作成します。
  3. 次に、custom.properties という名前でファイルを保存します。
  4. を次のコードサンプルcustom.propertiesで更新し、MQTT_END_POINT、POLICY_NAME、GEOFENCE_COLLECTION_NAME、 をリソース名に置き換えTOPIC_TRACKERます。

     ```
     MQTT_END_POINT=YOUR_END_POINT.us-east-1.amazonaws.com
     POLICY_NAME=YOUR_POLICY
     ```

```
GEOFENCE_COLLECTION_NAME=YOUR_GEOFENCE
TOPIC_TRACKER=YOUR_TRACKER
```

5. プロジェクトをクリーンアップして再構築します。その後、プロジェクトを実行できます。

- サインイン：

  アプリケーションにサインインするには、次のステップに従います。

  1. サインイン ボタンを押します。
  2. ID プール ID 、トラッカー名 、マップ名 を指定します。
  3. サインインをもう一度押して終了します。

- フィルターの管理：

  設定画面を開き、以下を実行します。

  1. スイッチ UI を使用してフィルターのオンとオフを切り替えます。
  2. 必要に応じて時間と距離のフィルターを更新します。

- 追跡オペレーション：

  追跡画面を開き、以下を実行します。

  - それぞれのボタンを押して、フォアグラウンド、バックグラウンド、またはバッテリーセーバーモードで追跡を開始および停止できます。

iOS 用のサンプル追跡およびジオフェンシングアプリケーション

このトピックでは、モバイルアプリケーションで Amazon Location ジオフェンスとトラッカーを使用する主な機能を示すために設計された iOS チュートリアルについて説明します。アプリケーションは、トラッカーとジオンフェンスが Lambda AWS IoT と Amazon Location の機能を組み合わせてどのように相互作用するかを示します。

トピック

- アプリケーションの Amazon Location リソースを作成する
- ジオフェンスコレクションを作成する
- トラッカーをジオフェンスコレクションにリンクする
- MQTT での AWS Lambda の使用
- サンプルアプリケーションコードのセットアップ

- サンプルアプリケーションの使用

アプリケーションの Amazon Location リソースを作成する

開始するには、必要な Amazon Location リソースを作成する必要があります。これらのリソースは、アプリケーションの機能や提供されたコードスニペットの実行に不可欠です。

> **Note**
>
> AWS アカウントを作成していない場合は、AWS アカウント管理ユーザーガイドの指示に従ってください。

開始するには、Amazon Cognito ID プール ID を作成する必要があります。次の手順を使用します。

1. Amazon Cognito コンソールを開き、左側のメニューからアイデンティティプールを選択し、アイデンティティプールの作成を選択します。
2. ゲストアクセスがチェックされていることを確認し、次へを押して続行します。
3. 次に、新しい IAM ロールを作成するか、既存の IAM ロールを使用します。
4. ID プール名を入力し、ID プールが、次の手順で作成するマップとトラッカーの Amazon Location (geo)リソースにアクセスできることを確認します。

次に、 AWS Amazon Location コンソールでマップを作成してスタイル設定する必要があります。次の手順を使用します。

1. Amazon Location コンソールのマップセクションに移動し、マップの作成 を選択します。
2. 新しいマップに名前 と説明 を付けます。割り当てた名前は、チュートリアルの後半で使用するため、記録します。
3. マップスタイルを選択するときは、マップデータプロバイダーを検討してください。詳細については、AWS サービス条件のセクション 82 を参照してください。
4. Amazon Location 利用規約 に同意し、マップの作成 を選択してマップ作成プロセスを完了します。

次に、Amazon Location コンソールでトラッカーを作成する必要があります。次の手順を使用します。

1. Amazon Location コンソールでマップセクションを開きます。
2. トラッカーを作成を選択します。
3. 必須フィールドに入力します。トラッカーの名前は、このチュートリアル全体で繰り返されるため、書き留めておきます。
4. 位置フィルタリング フィールドで、トラッカーリソースの使用方法に最も適したオプションを選択します。位置フィルタリングを設定しない場合、デフォルト設定は ですTimeBased。詳細については、「Amazon Location API リファレンス」の「 の追跡を開始する」およびPositionFiltering「」を参照してください。
5. トラッカーの作成を選択して、トラッカーの作成を完了します。

ジオフェンスコレクションを作成する

次に、ジオフェンスコレクションを作成します。コンソール、API、または CLI のいずれかを使用できます。次の手順では、各オプションについて説明します。

- Amazon Location コンソールを使用してジオフェンスコレクションを作成します。
  1. Amazon Location コンソールのジオフェンスコレクションセクションを開きます。
  2. [ジオフェンスコレクションの作成] を選択します。
  3. コレクションの名前と説明を入力します。
  4. をターゲット Amazon CloudWatch とする EventBridge ルールでは、オプションの EventBridge ルールを作成して、ジオフェンスイベントへの対応を開始できます。これにより、Amazon Location は にイベントを発行できます Amazon CloudWatch Logs。
  5. ジオフェンスコレクションの作成を押して、コレクションの作成を完了します。
- Amazon Location API を使用してジオフェンスコレクションを作成します。

  Amazon Location Geofences API の CreateGeofenceCollectionオペレーションを使用します。APIs 次の例では、 API リクエストを使用して、 というジオフェンスコレクションを作成しますGEOCOLLECTION_NAME。

```
POST /geofencing/v0/collections
Content-type: application/json

{
   "CollectionName": "GEOCOLLECTION_NAME",
   "Description": "Geofence collection 1 for shopping center",
   "Tags": {
```

```
      "Tag1" : "Value1"
   }
}
```

- AWS CLI コマンドを使用してジオフェンスコレクションを作成します。

  `create-geofence-collection`コマンドを実行します。次の例では、 AWS CLI を使用して、というジオフェンスコレクションを作成します*GEOCOLLECTION_NAME*。 AWS CLI の使用の詳細については、AWS 「 コマンドラインインターフェイスのドキュメント」を参照してください。

```
aws location \
  create-geofence-collection \
  --collection-name "ExampleGeofenceCollection" \
  --description "Shopping center geofence collection" \
  --tags Tag1=Value1
```

トラッカーをジオフェンスコレクションにリンクする

トラッカーをジオフェンスコレクションにリンクするには、コンソール、API、または CLI のいずれかを使用できます。次の手順では、各オプションについて説明します。

Amazon Location Service コンソールを使用して、トラッカーリソースをジオフェンスコレクションにリンクします。

1. Amazon Location コンソールを開きます。
2. 左のナビゲーションペインから、[トラッカー] を選択します。
3. Device Trackers で、ターゲットトラッカーの名前リンクを選択します。
4. [リンクされたジオフェンスコレクション] で [ジオフェンスコレクションをリンク] を選択します。
5. [リンクされたジオフェンスコレクション] ウィンドウのドロップダウンメニューからジオフェンスコレクションを選択します。
6. [Link (リンク)] を選択します。
7. トラッカーリソースをリンクすると、そのリソースには[アクティブ] ステータスが割り当てられます。

Amazon Location APIs。

Amazon Location Trackers API の `AsssociateTrackerConsumer` オペレーションを使用してください。次の例では、Amazon リソースネーム (ARN) を使用してジオフェンスコレクション ExampleTracker に関連付ける API リクエストを使用します。

```
POST /tracking/v0/trackers/ExampleTracker/consumers
Content-type: application/json
    {
       "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GEOCOLLECTION_NAME"
    }
```

AWS CLI コマンドを使用してトラッカーリソースをジオフェンスコレクションにリンクします。

`associate-tracker-consumer` コマンドを実行します。次の例では、AWS CLI を使用して、 というジオフェンスコレクションを作成します*GEOCOLLECTION_NAME*。

```
aws location \
    associate-tracker-consumer \
        --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GEOCOLLECTION_NAME" \
        --tracker-name "ExampleTracker"
```

MQTT での AWS Lambda の使用

AWS IoT と Amazon Location 間の接続を作成するには、 EventBridge CloudWatch イベントによって転送されるメッセージを処理するための Lambda 関数が必要です。この関数は、位置データを抽出し、Amazon Location 用にフォーマットして、Amazon Location Tracker API を通じて送信します。

次の手順は、Lambda コンソールを使用してこの関数を作成する方法を示しています。

1. コンソールを開きます。
2. ナビゲーションペインで、関数 を選択します。
3. 次に、関数の作成 を選択し、作成者を最初から作成する オプションが選択されていることを確認します。
4. 関数名 を指定し、ランタイムオプションで Node.js 16.x を選択します。
5. [関数を作成] を選択します。
6. コードタブを開いてエディタにアクセスします。
7. ファイル内のプレースホルダーコードを次のように上書きします`index.js`。

```
            const AWS = require('aws-sdk')
            const iot = new AWS.Iot();
            exports.handler =  function(event) {
              console.log("event===>>>", JSON.stringify(event));
              var param = {
                endpointType: "iot:Data-ATS"
              };
              iot.describeEndpoint(param, function(err, data) {
                if (err) {
                  console.log("error===>>>", err, err.stack); // an error occurred
                } else {
                  var endp = data['endpointAddress'];
                  const iotdata = new AWS.IotData({endpoint: endp});
                  const trackerEvent = event["detail"]["EventType"];
                  const src = event["source"];
                  const time = event["time"];
                  const gfId = event["detail"]["GeofenceId"];
                  const resources = event["resources"][0];
                  const splitResources = resources.split(".");
                  const geofenceCollection = splitResources[splitResources.length -
1];
                  const coordinates = event["detail"]["Position"];

                  const deviceId = event["detail"]["DeviceId"];
                  console.log("deviceId===>>>", deviceId);
                  const msg =  {
                      "trackerEventType" : trackerEvent,
                      "source" : src,
                      "eventTime" : time,
                      "geofenceId" : gfId,
                      "coordinates": coordinates,
                      "geofenceCollection": geofenceCollection
                    };
                  const params = {
                    topic: `${deviceId}/tracker`,
                    payload: JSON.stringify(msg),
                    qos: 0
                  };
                  iotdata.publish(params, function(err, data) {
                      if (err) {
                        console.log("error===>>>", err, err.stack); // an error
occurred
```

```
                    } else {
                      console.log("Ladmbda triggered===>>>", trackerEvent);  //
successful response
                    }
                  });
                }
              });
            }
```

8. デプロイを押して、更新された関数を保存します。
9. 次に、設定タブを開きます。
10. トリガー セクションで、トリガーを追加 ボタンを押します。
11. ソースフィールドで EventBridge（ CloudWatch イベント ）を選択します。
12. 既存のルールオプションを選択します。
13. ルール名を入力します。例: AmazonLocationMonitor-GEOFENCECOLLECTION_NAME。
14. 追加ボタンを押します。
15. これにより、アクセス許可タブにリソースベースのポリシーステートメントもアタッチされます。

次に、 AWS IoT MQTT テストクライアントをセットアップし、次の手順を使用します。

1. https://console.aws.amazon.com/iot/ を開きます。
2. 左側のナビゲーションペインで、MQTT テストクライアント を選択します。
3. MQTT 接続を設定できる MQTT テストクライアントというセクションが表示されます。
4. 必要な設定を行ったら、Connect ボタンをクリックして、指定されたパラメータを使用して MQTT ブローカーへの接続を確立します。
5. エンドポイントはチュートリアルの後半で使用するため、記録します。

   テストクライアントに接続したら、MQTT テストクライアントインターフェイスで提供されるそれぞれの入力フィールドを使用して、MQTT トピックをサブスクライブしたり、トピックにメッセージを発行したりできます。次に、 AWS IoT ポリシーを作成します。
6. 左側のメニューの「セキュリティの管理」で「セキュリティの管理」オプションを選択し、「ポリシー」をクリックします。
7. ポリシーの作成 ボタンをクリックします。
8. ポリシー名を入力します。

9. ポリシードキュメントで JSON タブを選択します。
10. 以下に示すポリシーをコピーして貼り付けます。ただし、必ず *REGION*と ですべての要素を更新してください*ACCOUNT_ID*。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:REGION:ACCOUNT_ID:client/${cognito-
identity.amazonaws.com:sub}",
        "arn:aws:iot:REGION:ACCOUNT_ID:topic/${cognito-
identity.amazonaws.com:sub}",
        "arn:aws:iot:REGION:ACCOUNT_ID:topicfilter/${cognito-
identity.amazonaws.com:sub}/*",
        "arn:aws:iot:REGION:ACCOUNT_ID:topic/${cognito-identity.amazonaws.com:sub}/
tracker"
      ],
      "Effect": "Allow"
    }
  ]
}
```

11. 作成ボタンを選択して終了します。

サンプルアプリケーションコードのセットアップ

サンプルコードを設定するには、次のツールがインストールされている必要があります。

- Git
- XCode 15.3 以降
- iOS Simulator 16 以降

この手順を使用して、サンプルアプリケーションコードを設定します。

1. https://github.com/aws-geospatial/amazon-location-samples-ios/tree/main/ という URL tracking-with-geofence-notificationsから git リポジトリをクローンします。
2. AWSLocationSampleApp.xcodeproj プロジェクトファイルを開きます。
3. パッケージ解決プロセスが完了するまで待ちます。
4. プロジェクトナビゲーションメニューで、名前ConfigTemplate.xcconfigを に変更Config.xcconfigし、次の値を入力します。

```
IDENTITY_POOL_ID = `YOUR_IDENTITY_POOL_ID`
MAP_NAME = `YOUR_MAP_NAME`
TRACKER_NAME = `YOUR_TRACKER_NAME`
WEBSOCKET_URL = `YOUR_MQTT_TEST_CLIENT_ENDPOINT`
GEOFENCE_ARN = `YOUR_GEOFENCE_COLLECTION_NAME`
```

サンプルアプリケーションの使用

サンプルコードを設定したら、iOS シミュレーターまたは物理デバイスでアプリケーションを実行できるようになりました。

1. アプリを構築して実行します。
2. アプリは、場所と通知のアクセス許可を要求します。これらを許可する必要があります。
3. Cognito 設定ボタンを押します。
4. 設定を保存します。
5. 時間、距離、精度のフィルターオプションが表示されるようになりました。必要に応じて使用してください。
6. アプリの追跡タブに移動すると、マップと追跡の開始ボタンが表示されます。
7. シミュレーターにアプリをインストールしている場合は、場所の変更をシミュレートできます。これは、ロケーションメニューオプションの 機能 で実行できます。例えば、機能 、ロケーション 、フリーウェイドライブ を選択します。
8. 追跡の開始 ボタンを押します。マップに追跡ポイントが表示されます。
9. アプリはバックグラウンドの場所も追跡しています。そのため、アプリをバックグラウンドで移動すると、バックグラウンドモードで追跡を続けるアクセス許可を求められます。
10. 追跡を停止ボタンをタップして、追跡を停止できます。

Amazon Location Service リソースにタグを付ける

Amazon Location のリソースタグを使用して、リソースを用途、所有者、環境、基準別に分類するタグを作成します。リソースにタグを付けると、リソースの管理、識別、整理、検索、フィルタリングに役立ちます。

例えば、AWS Resource Groups を使用すると、1 つまたは複数のタグ、またはタグの一部に基づいて AWS リソースのグループを作成できます。また、AWS CloudFormation スタック内での出現回数に基づいてグループを作成することもできます。リソースグループとタグエディタを使用すると、複数のサービス、リソース、リージョンで構成されるアプリケーションのデータを 1 か所にまとめて表示できます。一般的なタグ付け戦略の詳細については、「AWS ジェネラルリファレンス」を参照してください。

各タグは、ユーザー自身が定義するキーと値で構成されるラベルです。

- タグキー — タグ値を分類する一般的なラベル。例えば、「CostCenter」と入力します。
- タグ値 — タグキーカテゴリの説明 (オプション) 。例えば、「MobileAssetTrackingResourcesProd」と入力します。

このトピックでは、タグ付けの制限を確認できるので、タグの使用を開始する上で役立ちます。また、タグを使用してアクティブなタグのAWS コストを追跡する方法についても説明します。コスト配分レポートを使用します。

トピック

タグ指定の制限

タグには以下のベーシックな制限があります。

- リソースあたりのタグの最大数 - 50

- タグキーは、リソースごとにそれぞれ一意である必要があります。また、各タグキーに設定できる値は 1 つのみです。

> **Note**
>
> 既存のタグとタグキーが同じ新しいタグを追加すると、既存のタグは新しいタグで上書きされます。

- キーの最大長 - UTF-8 の 128 Unicode 文字
- 値の最大長 - UTF-8 の 256 Unicode 文字
- すべてのサービスで使用できる文字は、UTF-8 で表現できる文字、数字、およびスペースに加えて、 + - = . _ : / @ です。
- タグのキーと値は大文字と小文字が区別されます。
- aws: プレフィックスは AWS 用に限定されています。タグにこのプレフィックスが付いたタグキーがある場合、タグのキーまたは値を編集、削除することはできません。aws: プレフィックスを持つタグは、リソースあたりのタグ数の制限にはカウントされません。

リソースにタグを付けるアクセス許可を付与する

IAM ポリシーを使用して Amazon Location リソースへのアクセスを制御し、リソース作成時にタグを付けるアクセス許可を付与できます。ポリシーには、リソースを作成する権限を付与するほかに、タグ付け操作を許可するAction権限を含めることができます。

- geo:TagResource — 指定された Amazon Location リソースに 1 つまたは複数のタグを付ける権限をユーザーに付与します。

- geo:UntagResource — 指定された Amazon Location リソースから 1 つまたは複数のタグを削除する権限をユーザーに付与します。
- geo:ListTagsForResource — Amazon Location リソースに付けられたタグの一覧を表示する権限をユーザーに付与します。

以下は、ジオフェンスコレクションの作成とリソースへのタグ付けをユーザーに許可するポリシーの例です。

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AllowTaggingForGeofenceCollectionOnCreation",
            "Effect": "Allow",
            "Action": [
                "geo:CreateGeofenceCollection",
                "geo:TagResource"
            ],
            "Resource": "arn:aws:geo:region:accountID:geofence-collection/*"
    ]
}
```

Amazon Location Service リソースにタグを追加する

Amazon Location コンソール、AWS CLI、または Amazon Location API を使用して、リソース作成時にタグを追加できます。

- マップリソースを作成する
- 場所インデックスリソースの作成
- ルート計算リソースを作成する
- ジオフェンスコレクションを作成する
- トラッカーリソースを作成する

既存のリソースにタグを追加する、タグを編集、または削除するには

1. Amazon Location コンソールを開きます (https://console.aws.amazon.com/location/) 。
2. 左側のナビゲーションペインで、タグを付けるリソースを選択します。ここでは例として、[マップ]を選択します。
3. リストからリソースを選択します。
4. 次に [タグの管理] を選択して、タグの追加、編集または削除を行います。

リソースコストをタグごとに追跡します。

コスト配分のタグを使用すると、AWSコストを詳細に追跡できます。コスト配分タグを有効化すると、AWS はコスト配分レポート上のリソースコストをコスト配分タグごとに整理します。これにより、使用コストを分類して追跡できます。

有効化できるコスト配分タグには以下の 2 つのタイプがあります。

- AWS 生成タグ — これらのタグは AWS によって生成されます。AWS タグには、aws: プレフィックスを使用します。例：aws:createdBy
- ユーザー定義タグ — これらは自分で作成するカスタムタグです。ユーザー定義タグには、user: プレフィックスを使用します。例：user:CostCenter

タグタイプごとに有効化する必要があります。タグを有効化すると、AWS Cost Explorer を有効化したり、月次コスト配分レポートを表示できます。

AWS-generated tags

AWS 生成タグを有効化するには

1. 請求情報とコスト管理コンソール（https://console.aws.amazon.com/billing/）を開きます。
2. 左側のナビゲーションペインで、[コスト配分タグ] を選択します。
3. [AWS生成コスト配分タグ] タブで、有効化するタグキーを選択します。
4. [アクティブ化] を選択します。

User-defined tags

ユーザー定義タグを有効化するには

1. 請求情報とコスト管理コンソール（https://console.aws.amazon.com/billing/）を開きます。
2. 左側のナビゲーションペインで、[コスト配分タグ] を選択します。
3. [ユーザー定義のコスト配分タグ] タブで、有効化するタグキーを選択します。
4. [アクティブ化] を選択します。

タグを有効化すると、AWS はリソースの使用状況とコストに関する月次コスト配分レポートを生成します。このコスト配分レポートには、タグ付きリソースとタグなしリソースを含む、ご利用のAWSサービスのコストすべてが請求期間別に出力されます。詳細については、AWS Billing and Cost Managementユーザーガイド の [コスト配分タグの使用] をご参照ください。

タグを使用して Amazon Location Service リソースへのアクセスを制御する

AWS Identity and Access Management（IAM）ポリシーでは、タグベースの条件をサポートしています。このため、特定のタグキーと値に基づいてリソースの承認を管理できます。たとえば、IAM

ロールポリシーに、特定の環境 (開発) 、テスト、本番など) へのアクセスを制限する条件を含めることができます。

詳細については、タグに基づくリソースアクセスの制御に関するトピックを参照してください。

詳細はこちら

以下についての詳細：

- タグ付けに関するベストプラクティスの詳細については、「AWS 全般のリファレンス」の「AWS リソースのタグ付け」を参照してください。
- タグを使用してAWS リソースへのアクセスを制御する方法については、「AWS Identity and Access Management ユーザーガイド」の「タグによるAWS リソースへのアクセスの制御」を参照してください。

Amazon Location Service へのアクセスを許可する

Amazon Location Service を使用するには、Amazon Location を構成するリソースと API へのアクセス権をユーザーに付与する必要があります。リソースへのアクセスを付与するには、3 つの方法があります。

- IAM を使用する — AWS IAM Identity Center または AWS Identity and Access Management (IAM) で認証されたユーザーにアクセス権を付与するには、目的のリソースへのアクセスを許可する IAM ポリシーを作成します。IAMとアマゾン· Amazon Location の詳細については、Amazon Location Service での Identity and Access Managementを参照してください。
- API キーを使用する — 認証されていないユーザーにアクセス権を付与するには、Amazon Location Service リソースへの読み取り専用アクセスを許可する API キーを作成できます。これは、すべてのユーザーを認証したくない場合に便利です。たとえば、Web アプリケーションなどです。API キーに関する詳細については、「API キーを使用してアプリケーションへの認証されていないゲストアクセスを許可する」を参照してください。
- Amazon Cognito を使用する — API キーに代わる方法として、Amazon Cognito を使用して匿名アクセスを許可する方法があります。Amazon Cognito では、認証されていないユーザーが実行できる操作を定義するポリシーを使用して、より豊富な権限を作成することができます。Amazon Cognitoの使用に関する詳細については、Amazon Cognito を使用して、アプリケーションで認証されていないゲストアクセスを許可する方法を参照してください。

> **Note**
>
> Amazon Cognito を使用して、独自の認証プロセスを使用したり、Amazon Cognito フェデレーテッドアイデンティティを使用して複数の認証方法を組み合わせたりすることもできます。詳細については、Amazon Cognito 開発者ガイドの「Getting Started with Federated Identities」を参照してください。を参照してください。

トピック

- API キーを使用してアプリケーションへの認証されていないゲストアクセスを許可する
- Amazon Cognito を使用して、アプリケーションで認証されていないゲストアクセスを許可する方法

API キーを使用してアプリケーションへの認証されていないゲストアクセスを許可する

アプリケーションで Amazon Location Service API を呼び出すときは、通常、API 呼び出しを行う権限を持つ認証済みユーザーとしてこの呼び出しを行います。ただし、アプリケーションのすべてのユーザーを認証したくないケースも場合もあるでしょう。たとえば、会社の所在地を表示する Web アプリケーションを、ログインしているかどうかに関係なく、Web サイトを使用するすべてのユーザーが利用できるようにしたい場合があります。この場合API キーを使ってAPI 呼び出しを行うという方法もあります。

API キーは、 内の特定の Amazon Location Service リソース AWS アカウント、およびそれらのリソースに対して実行できる特定のアクションに関連付けられたキー値です。アプリケーションの API キーを使用して、それらのリソースの Amazon Location API を認証なしで呼び出すことができます。たとえば、API キーをマップリソース MyMap と`GetMap*` アクションに関連付けると、その API キーを使用するアプリケーションはそのリソースで作成されたマップを表示できるようになり、アカウントの他の使用量と同様にアカウントに請求されます。同じ API キーを使用しても、マップリソースの変更または更新の権限は付与されません。リソースの使用のみが許可されます。

> **Note**
>
> API キーはマップ、配置、ルートリソースでのみ使用でき、これらのリソースを変更または作成することはできません。アプリケーションが他のリソースにアクセスしたり、認証されていないユーザーがアクションにアクセスしたりする必要がある場合は、Amazon Cognito

を使用して API キーと一緒に、または代わりに API キーへのアクセスを提供することができます。詳細については、「Amazon Cognito を使用して、アプリケーションで認証されていないゲストアクセスを許可する方法」を参照してください。

API キーには、 AWS アカウント内の 1 つ以上のリソースへのアクセスを許可するプレーンテキストの値が含まれます。API キーをコピーしたユーザーは、同じリソースにアクセスできます。これを回避するには、API キーを作成するときに API キーの使用ができるドメインを指定することができます。これらのドメインはリファラーと呼ばれます。必要に応じて、API キーの有効期限を設定して短期間の API キーを作成することもできます。

トピック

- API キーと Amazon Cognito の比較
- API キーの作成
- API キーを使用して Amazon Location API を呼び出します。
- API キーを使用してマップをレンダリングします。
- API キーの有効期間の管理

API キーと Amazon Cognito の比較

API キーと Amazon Cognito は、似たようなシナリオでも同じような方法で使用されます。では、なぜ一方を他方よりも優先して使用するのでしょうか。以下のリストでは、その二つの違いをいくつか紹介します。

- API キーは、マップ、プレイス、ルート・リソースでのみ、また特定のアクションでのみ利用可能です。Amazon Cognito は、ほとんどの Amazon Location Service API へのアクセスを認証するために使用されます。
- API キーを使用したマップリクエストのパフォーマンスは、通常、Amazon Cognito を使用した同様のシナリオよりも高速です。認証が簡単なため、短期間に同じマップタイルを再度取得したときに、サービスとキャッシュされたリクエストへのラウンドトリップが少なくなります。
- Amazon Cognito を使用して、独自の認証プロセスを使用したり、Amazon Cognito フェデレーテッドアイデンティティを使用して複数の認証方法を組み合わせたりすることもできます。詳細については、「Amazon Cognito 開発者ガイド」の「フェデレーテッドアイデンティティの使用を開始する」を参照してください。

API キーの作成

APIキーを作成し、 AWS アカウントの1つまたは複数のリソースに関連付けることができます。

Amazon Location Service コンソール、、 AWS CLIまたは Amazon Location APIsを使用して API キーを作成できます。

Console

Amazon Location Service コンソールを使用して API キーを作成する

1. Amazon Location コンソールで、左側のメニューから API キーを選択します。
2. 新しいAPIを作成する ページで、新しいAPI を選択します。
3. API キーの作成ページで、次の情報を入力します。
   - 名前— API キーの名前、例えば `MyWebAppKey`。
   - 説明 — API キーのオプションの説明。
   - リソース — この API キーを使用してアクセスを許可する Amazon Location リソースをドロップダウンから選択します。リソースの追加を選択すると、複数のリソースを追加できます。
   - アクション — この API キーを利用して、承認したアクションを指定します。選択した各リソースタイプと一致するアクションを少なくとも 1 つ選択する必要があります。たとえば、Place リソースを選択した場合は、Places Actions の下にある選択肢を少なくとも 1 つ選択する必要があります。
   - 有効期限 — API キーの有効期限を任意で追加することができます。詳細については、「API キーの有効期間の管理」を参照してください。
   - リファラー — API キーの使用ができるドメインを 1 つ以上追加することもできます。例えば、API キーがウェブサイト`example.com`で実行されるアプリケーションを許可するものであれば、許可されるリファラーとして`*.example.com/`を置くことができます。
   - タグ — オプションで API キーにタグを追加します。
4. API キーを作成を選択して API キーを作成します。
5. API キーの詳細ページには、作成した API キーに関する情報が表示されます。API キーを表示を選択すると、Amazon Location API を呼び出すときに使用するキーバリューが表示されます。キー値は次のフォーマットになります`v1.public.a1b2c3d4...`。APIキーを使用してマップをレンダリングする方法の詳細については、API キーを使用してマップをレンダリングします。をご参照ください。

API

Amazon Location APIを使用してAPIキーを作成するには

Amazon Location API CreateKey のオペレーションを使用してください。

次の例は、有効期限*ExampleKey*のない という API キーを作成し、単一のマップリソースにアクセスする API リクエストです。

```
POST /metadata/v0/keys HTTP/1.1
Content-type: application/json

{
   "KeyName": "ExampleKey"
   "Restrictions": {
      "AllowActions": [
         "geo:GetMap*"
      ],
      "AllowResources": [
         "arn:aws:geo:region:map/mapname"
      ]
   },
   "NoExpiry": true
   }
}
```

レスポンスには、アプリケーション内のリソースにアクセスするときに使用する API キーバリューが含まれています。キー値は次のフォーマットになりますv1.public.*a1b2c3d4...*。マップのレンダリングに API キーを使用する方法については、「API キーを使用してマップをレンダリングします。」をご参照ください。

DescribeKeyAPI を使用してキーのキー値を後で検索することもできます。

AWS CLI

AWS CLI コマンドを使用して API キーを作成するには

create-keyコマンドを実行します。

次の例では、有効期限*ExampleKey*のない という API キーと、単一のマップリソースへのアクセスを作成します。

```
aws location \
```

```
  create-key \
  --key-name ExampleKey \
  --restrictions '{"AllowActions":["geo:GetMap*"],"AllowResources":
["arn:aws:geo:region:map/mapname"]}' \
  --no-expiry
```

レスポンスには、アプリケーション内のリソースにアクセスするときに使用する API キーバリューが含まれています。キー値は次のフォーマットになりますv1.public.*a1b2c3d4...*。マップのレンダリングに API キーを使用する方法については、「API キーを使用してマップをレンダリングします。」をご参照ください。create-keyへの返答は次のようにになります。

```
{
    "Key": "v1.public.a1b2c3d4...",
    "KeyArn": "arn:aws:geo:region:accountId:api-key/ExampleKey",
    "KeyName": "ExampleKey",
    "CreateTime": "2023-02-06T22:33:15.693Z"
}
```

describe-keyを使用してキーバリューを後で検索することもできます。次の例は、 という名前の API キーdescribe-keyで を呼び出す方法を示しています*ExampleKey*。

```
aws location describe-key \
    --key-name ExampleKey
```

API キーを使用して Amazon Location API を呼び出します。

API キーを作成したら、そのキーバリューを使用してアプリケーションの Amazon Location API を呼び出すことができます。

API キーをサポートする API には、API キーバリューを取る追加パラメータがあります。たとえば、GetPlace API を呼び出す場合、以下のようにキーパラメータを入力できます。

```
GET /places/v0/indexes/IndexName/places/PlaceId?key=KeyValue
```

この値を入力する場合、通常どおり AWS Sig v4 で API コールを認証する必要はありません。

JavaScript デベロッパーは、Amazon Location を使用して API キーによる API オペレーションの認証JavaScript 認証ヘルパーに役立てることができます。

モバイルデベロッパーには、次の Amazon Location モバイル認証 SDKsを使用できます。

- Amazon Location Service Mobile Authentication SDK for iOS
- Amazon Location Service Mobile Authentication SDK for Android

AWS CLI ユーザーの場合は、 `--key`パラメータも使用して`--no-sign-request`、Sig v4 で署名しないようにする必要があります。

> **Note**
>
> Amazon Location Service への呼び出しに keyと AWS Sig v4 の両方の署名を含めると、 API キーのみが使用されます。

API キーを使用してマップをレンダリングします。

API キー値を使用して、 を使用してアプリケーションでマップをレンダリングできます MapLibre。これは、直接呼び出している他の Amazon Location API で APIs キーを使用するのとは少し異なります。 がこれらの呼び出し MapLibre を行うためです。

次のサンプルコードは、GL MapLibre JS マップコントロールを使用して API キーを使用してシンプルなウェブページにマップをレンダリングする方法を示しています。このコードが正しく機能するには、*v1.public.your-api-key-value*、*us-east-1*、および *ExampleMap*文字列を に一致する値に置き換えます AWS アカウント。

```
<!-- index.html -->
<html>
  <head>
    <link href="https://unpkg.com/maplibre-gl@1.14.0/dist/maplibre-gl.css"
 rel="stylesheet" />
    <style>
      body { margin: 0; }
      #map { height: 100vh; }
    </style>
  </head>
  <body>
    <!-- Map container -->
    <div id="map" />
    <!-- JavaScript dependencies -->
    <script src="https://unpkg.com/maplibre-gl@1.14.0/dist/maplibre-gl.js"></script>
    <script>
      const apiKey = "v1.public.your-api-key-value"; // API key
```

```
      const region = "us-east-1"; // Region
      const mapName = "ExampleMap"; // Map name
      // URL for style descriptor
      const styleUrl = `https://maps.geo.${region}.amazonaws.com/maps/v0/maps/
${mapName}/style-descriptor?key=${apiKey}`;
      // Initialize the map
      const map = new maplibregl.Map({
        container: "map",
        style: styleUrl,
        center: [-123.1187, 49.2819],
        zoom: 11,
      });
      map.addControl(new maplibregl.NavigationControl(), "top-left");
    </script>
  </body>
</html>
```

API キーの有効期間の管理

無期限に機能する API キーの作成ができます。ただし、一時的な API キーを作成したり、API キーを定期的にローテーションしたり、既存の API キーを取り消したりする場合は、API キーの有効期限を使用できます。

新しい API キーを作成したり、既存の API キーを更新したりするときに、その API キーの有効期限を設定することができます。

- API キーの有効期限が切れると、キーは自動的に無効化されます。非アクティブなキーはマップリクエストには使用できなくなります。
- API キーは、非アクティブ化してから 90 日後に削除することができます。
- まだ削除していない非アクティブなキーがある場合は、有効期限を将来の時間に更新することで復元できます。
- 永久キーを作成するには、有効期限を削除することができます。
- 過去 7 日以内に使用された API キーを非アクティブ化しようとすると、変更の確認を求めるメッセージが表示されます。Amazon Location Service API または を使用している場合は AWS CLI、`ForceUpdate`パラメータを true に設定しない限り、エラーが発生します。

Amazon Cognito を使用して、アプリケーションで認証されていないゲストアクセスを許可する方法

フロントエンド SDKsとダイレクト HTTPS リクエストの両方で AWS Identity and Access Management (IAM) を直接使用する代わりに、Amazon Cognito 認証を使用できます。

以下の場合は、この認証形式を使用します。

- 認証されていないユーザー — 匿名ユーザーがいるウェブサイトがある場合は、Amazon Cognito アイデンティティプールを使用できます。詳細については、「the section called “Amazon Cognito を使用する”」セクションを参照してください。
- 独自の認証 — 独自の認証プロセスを使用する場合や、複数の認証方法を組み合わせたい場合は、Amazon Cognito フェデレーテッド IDを使用できます。詳細については、「Amazon Cognito 開発者ガイド」の「フェデレーテッドアイデンティティの使用を開始する」を参照してください。

Amazon Cognito は、ウェブおよびモバイルアプリの認証、認可、およびユーザー管理機能を提供します。アプリケーションがスコープダウンされた一時的な AWS 認証情報を取得する方法として、Amazon Cognito の認証されていない ID プールを Amazon Location で使用できます。

詳細については、「Amazon Cognito 開発者ガイド」の「ユーザープールの使用を開始する」を参照してください。

> **Note**
>
> モバイルデベロッパーの場合、Amazon Location は iOS と Android の両方用のモバイル認証 SDKs を提供しています。詳細については、次の github リポジトリを参照してください。
>
> - Amazon Location Service Mobile Authentication SDK for iOS
> - Amazon Location Service Mobile Authentication SDK for Android

Amazon Cognito アイデンティティプールを作成する

Amazon Cognito ID プールを作成して、Amazon Cognito コンソール、、または Amazon CognitoAmazon Cognito APIs を介して AWS CLI、認証されていないゲストがアプリケーションにアクセスできるようにします。

> **Important**
>
> 作成するプールは、使用している Amazon Location Service リソースと同じ AWS アカウント と AWS リージョンに存在する必要があります。

認証されていない ID ロールに関連付けられた IAM ポリシーは、以下のアクションで使用できます。

- `geo:GetMap*`
- `geo:SearchPlaceIndex*`
- `geo:GetPlace`
- `geo:CalculateRoute*`
- `geo:GetGeofence`
- `geo:ListGeofences`
- `geo:PutGeofence`
- `geo:BatchDeleteGeofence`
- `geo:BatchPutGeofence`
- `geo:BatchEvaluateGeofences`
- `geo:GetDevicePosition*`
- `geo:ListDevicePositions`
- `geo:BatchDeleteDevicePositionHistory`
- `geo:BatchGetDevicePosition`
- `geo:BatchUpdateDevicePosition`

他の Amazon Location アクションを含めても効果はなく、認証されていない ID はそれらを呼び出すことができません。

Example

Amazon Cognito コンソールを使用して、アイデンティティプールを作成するには

1. Amazon Cognito コンソールに移動します。
2. [アイデンティティプールの管理] を選択します。

3. [新しいアイデンティティプールの作成] を選択し、アイデンティティプールの名前を入力します。
4. 折りたたみ式の [認証されていない ID] セクションで、[認証されていない ID に対してアクセスを有効にする] を選択します。
5. [プールの作成] を選択します。
6. アイデンティティプールで使用する IAM ロールを選択します。
7. [詳細を表示] を展開します。
8. [認証されていない ID] に、ロール名を入力します。
9. [ポリシードキュメントの表示] セクションを展開し、[編集] を選択してポリシーを追加します。
10. リソースへのアクセスを許可するポリシーを追加します。

    マップ、場所、トラッカー、ルートのポリシー例を以下に示します。この例を独自のポリシーに使用するには、*region* と *accountID* のプレースホルダーを置き換えてください。

    Maps policy example

    次のポリシーは、 という名前のマップリソースへの読み取り専用アクセスを許可します*ExampleMap*。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MapsReadOnly",
      "Effect": "Allow",
      "Action": [
        "geo:GetMapStyleDescriptor",
        "geo:GetMapGlyphs",
        "geo:GetMapSprites",
        "geo:GetMapTile"
      ],
      "Resource": "arn:aws:geo:region:accountID:map/ExampleMap"
    }
  ]
}
```

    aws:refererと一致する IAM 条件を追加すると、リソースへのブラウザアクセスを URL または URL プレフィックスのリストに制限できます。次の例では、RasterEsriImageryと

いう名前のマップリソースへのアクセスは、example.comの Web サイトからのみ許可されます。

> **Warning**
>
> aws:refererでは、アクセスを制限することはできますが、セキュリティメカニズムではありません。一般に知られている参照子のヘッダー値を含めるのは危険です。不正な当事者は、変更されたブラウザまたはカスタムブラウザを使用して任意の aws:referer 値を提供することができます。そのため、不正な当事者による直接 AWS リクエストの防止には aws:referer を使用しないでください。このキーは、Amazon S3 に保存されているデジタルコンテンツなど、不正なサードパーティーサイトで参照されることから保護するためにのみ、お客様に提供されています。詳細については、[AWS：参照子]を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "geo:GetMap*",
            "Resource": "arn:aws:geo:us-west-2:111122223333:map/
RasterEsriImagery",
            "Condition": {
                "StringLike": {
                    "aws:referer": [
                        "https://example.com/*",
                        "https://www.example.com/*"
                    ]
                }
            }
        }
    ]
}
```

Tangram を使用してマップを表示する場合、Maps API から返されるスタイル記述子、グリフ、スプライトは使用されません。代わりに、スタイルルールと必要なアセットを含む.zip ファイルを指して設定します。次のポリシーは、 GetMapTileオペレーションExampleMapの という名前のマップリソースへの読み取り専用アクセスを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MapsReadOnly",
      "Effect": "Allow",
      "Action": [
        "geo:GetMapTile"
      ],
      "Resource": "arn:aws:geo:region:accountID:map/ExampleMap"
    }
  ]
}
```

Places policy example

次のポリシーは、テキストまたは位置で場所を検索*ExamplePlaceIndex*するために、 という名前の場所インデックスリソースへの読み取り専用アクセスを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PlacesReadOnly",
      "Effect": "Allow",
      "Action": [
          "geo:SearchPlaceIndex*",
          "geo:GetPlace"
      ],
      "Resource": "arn:aws:geo:region:accountID:place-index/ExamplePlaceIndex"
    }
  ]
}
```

`aws:referer`と一致する IAM 条件を追加すると、リソースへのブラウザアクセスを URL または URL プレフィックスのリストに制限できます。次の例では、 を除くすべての参照元ウェブサイト*ExamplePlaceIndex*から、 という名前の場所インデックスリソースへのアクセスを拒否しますexample.com。

> **Warning**
>
> aws:refererでは、アクセスを制限することはできますが、セキュリティメカニズムではありません。一般に知られている参照子のヘッダー値を含めるのは危険です。不正な当事者は、変更されたブラウザまたはカスタムブラウザを使用して任意の aws:referer 値を提供することができます。そのため、不正な当事者による直接 AWS リクエストの防止には aws:referer を使用しないでください。このキーは、Amazon S3 に保存されているデジタルコンテンツなど、不正なサードパーティーサイトで参照されることから保護するためにのみ、お客様に提供されています。詳細については、[AWS：参照子]を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "geo:*",
            "Resource": "arn:aws:geo:us-west-2:111122223333:place-
index/ExamplePlaceIndex",
            "Condition": {
                "StringLike": {
                    "aws:referer": [
                        "https://example.com/*",
                        "https://www.example.com/*"
                    ]
                }
            }
        }
    ]
}
```

Trackers policy example

次のポリシーは、デバイスの位置を更新*ExampleTracker*するために、 という名前のトラッカーリソースへのアクセスを許可します。

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "UpdateDevicePosition",
      "Effect": "Allow",
      "Action": [
        "geo:BatchUpdateDevicePosition"
      ],
      "Resource": "arn:aws:geo:region:accountID:tracker/ExampleTracker"
    }
  ]
}
```

aws:refererと一致する IAM 条件を追加すると、リソースへのブラウザアクセスを URL または URL プレフィックスのリストに制限できます。次の例では、 を除くすべての参照元ウェブサイトExampleTrackerから という名前のトラッカーリソースへのアクセスを拒否しますexample.com。

> **Warning**
>
> aws:refererでは、アクセスを制限することはできますが、セキュリティメカニズムではありません。一般に知られている参照子のヘッダー値を含めるのは危険です。不正な当事者は、変更されたブラウザまたはカスタムブラウザを使用して任意の aws:referer 値を提供することができます。そのため、不正な当事者による直接 AWS リクエストの防止には aws:referer を使用しないでください。このキーは、Amazon S3 に保存されているデジタルコンテンツなど、不正なサードパーティーサイトで参照されることから保護するためにのみ、お客様に提供されています。詳細については、[AWS：参照子]を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "geo:GetDevice*",
            "Resource": "arn:aws:geo:us-
west-2:111122223333:tracker/ExampleTracker",
            "Condition": {
                "StringLike": {
                    "aws:referer": [
```

```
                        "https://example.com/*",
                        "https://www.example.com/*"
                    ]
                }
            }
        }
    ]
}
```

Routes policy example

次のポリシーは、ルートを計算する*ExampleCalculator*ために、 という名前のルート計算リソースへのアクセスを許可します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RoutesReadOnly",
      "Effect": "Allow",
      "Action": [
        "geo:CalculateRoute"
      ],
      "Resource": "arn:aws:geo:region:accountID:route-
calculator/ExampleCalculator"
    }
  ]
}
```

aws:refererと一致する IAM 条件を追加すると、リソースへのブラウザアクセスを URL または URL プレフィックスのリストに制限できます。次の例では、 を除くすべての参照元ウェブサイト*ExampleCalculator*から という名前のルート計算ツールへのアクセスを拒否しますexample.com。

> **Warning**
>
> aws:refererでは、アクセスを制限することはできますが、セキュリティメカニズムではありません。一般に知られている参照子のヘッダー値を含めるのは危険です。不正な当事者は、変更されたブラウザまたはカスタムブラウザを使用して任意の aws:referer 値を提供することができます。そのため、不正な当事者による直接 AWS リクエストの防止には aws:referer を使用しないでください。この

キーは、Amazon S3 に保存されているデジタルコンテンツなど、不正なサードパーティーサイトで参照されることから保護するためにのみ、お客様に提供されています。詳細については、[AWS：参照子]を参照してください。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "geo:*",
            "Resource": "arn:aws:geo:us-west-2:111122223333:route-
calculator/ExampleCalculator",
            "Condition": {
                "StringLike": {
                    "aws:referer": [
                        "https://example.com/*",
                        "https://www.example.com/*"
                    ]
                }
            }
        }
    ]
}
```

> **Note**
>
> 認証されていない ID プールはセキュリティで保護されていないインターネットサイトに公開することを目的としていますが、標準の期間限定 AWS 認証情報と交換されることに注意してください。
> 認証されていないアイデンティティプールに関連付けられている IAM ロールの範囲を適切に設定することが重要です。

11. [許可] を選択してアイデンティティプールを作成します。

作成されるアイデンティティプールは次の構文に従います。*<region>*:*<GUID>*.

例：

```
us-east-1:1sample4-5678-90ef-aaaa-1234abcd56ef
```

Amazon Location 固有のその他のポリシー例については、the section called “アイデンティティベースポリシーの例”を参照してください。

での Amazon Cognito ID プールの使用 JavaScript

次の例では、作成した認証されていない ID プールを認証情報と交換し、それをマップリソース のスタイル記述子の取得に使用します*ExampleMap*。

```
const AWS = require("aws-sdk");

const credentials = new AWS.CognitoIdentityCredentials({
   IdentityPoolId: "<identity pool ID>" // for example, us-east-1:1sample4-5678-90ef-
aaaa-1234abcd56ef
});

const client = new AWS.Location({
   credentials,
   region: AWS.config.region || "<region>"
});

console.log(await client.getMapStyleDescriptor("ExampleMap").promise());
```

> **Note**
>
> 認証されていない ID から取得した認証情報の有効期限は、1 時間です。

以下は、有効期限が切れる前に認証情報を自動的に更新する関数の例です。

```
async function refreshCredentials() {
  await credentials.refreshPromise();
  // schedule the next credential refresh when they're about to expire
  setTimeout(refreshCredentials, credentials.expireTime - new Date());
}
```

Amazon Location JavaScript 認証ヘルパーを使用して、この作業を簡略化することができます。これは認証情報の取得と更新の両方の代わりになります。この例では、 AWS SDK for JavaScript v3 を使用しています。

```
import { LocationClient, GetMapStyleDescriptorCommand } from "@aws-sdk/client-
location";
import { withIdentityPoolId } from "@aws/amazon-location-utilities-auth-helper";

const identityPoolId = "<identity pool ID>"; // for example, us-
east-1:1sample4-5678-90ef-aaaa-1234abcd56ef

// Create an authentication helper instance using credentials from Cognito
const authHelper = await withIdentityPoolId(identityPoolId);

const client = new LocationClient({
  region: "<region>", // The region containing both the identity pool and tracker
 resource
  ...authHelper.getLocationClientConfig(), // Provides configuration required to make
 requests to Amazon Location
});

const input = {
  MapName: "ExampleMap",
};

const command = new GetMapStyleDescriptorCommand(input);

console.log(await client.send(command));
```

次のステップ

- ロールを変更するには、IAM コンソールにアクセスしてください。
- アイデンティティプールを管理するには、Amazon Cognito コンソールにアクセスしてください。

モニタリング Amazon Location Service

Amazon Location Service を利用する場合、利用状況とリソースをリアルタイムにモニタリングすることができます。

- Amazon CloudWatch。Amazon Location Service リソースをモニタリングし、ほぼリアルタイムで統計情報をメトリックスとして提供されます。
- AWS CloudTrail。 Amazon Location Service API へのすべての呼び出しの追跡が提供されています。

このセクションでは、これらのプロセスについて説明します。

トピック

- Amazon による Amazon Location Service のモニタリング CloudWatch
- での ログ記録とモニタリング

Amazon による Amazon Location Service のモニタリング CloudWatch

Amazon は、 AWSリソースと AWSで実行しているアプリケーションをほぼリアルタイムで CloudWatch モニタリングします。を使用して Amazon Location リソースをモニタリングすることで CloudWatch、raw データを収集し、メトリクスを意味のある統計にほぼリアルタイムで処理できます。最大 15 か月間の履歴情報を表示したり、Amazon CloudWatch コンソールでメトリクスを検索して Amazon Location リソースをより詳細に把握したりできます。特定のしきい値にアラームを設定し、これらのしきい値に達したときに通知を送信したりアクションを実行したりできます。

詳細については、「Amazon ユーザーガイド CloudWatch 」を参照してください。

トピック

- Amazon Location Service メトリクスを Amazon にエクスポート CloudWatch
- Amazon Location Service のメトリクスを見る
- Amazon Location Service メトリクスの CloudWatch アラームを作成する
- CloudWatch を使用したクォータに対する使用状況のモニタリング
- CloudWatch Amazon Location Service の メトリクスの例

Amazon Location Service メトリクスを Amazon にエクスポート CloudWatch

メトリクスは、 にエクスポートされる時系列のデータポイントです CloudWatch。ディメンションは、メトリクスを一意に識別する名前/値のペアです。詳細については、「Amazon ユーザーガイド」の CloudWatch 「メトリクスとCloudWatchディメンションの使用 CloudWatch 」を参照してください。

以下は、Amazon Location Service が `AWS/Location`名前空間 CloudWatch の にエクスポートするメトリクスです。

メトリクス	説明
`CallCount`	特定の API エンドポイントに対して行われた呼び出しの回数。

メトリクス	説明
	有効なディメンション:Amazon Location Service API 名 有効な統計: Sum 単位: カウント
ErrorCount	特定の API エンドポイントに対して行われた呼び出しによるエラー・レスポンスの数。 有効なディメンション:Amazon Location Service API 名 有効な統計: Sum 単位: カウント
SuccessCount	特定の API エンドポイントに対して成功した呼び出しの数。 有効なディメンション:Amazon Location Service API 名 有効な統計: Sum 単位: カウント
CallLatency	特定の API エンドポイントに対して呼び出しが行われたときに、オペレーションがレスポンスを処理して応答が返されるまでにかかる時間。 有効なディメンション:Amazon Location Service API 名 有効な統計: Average 単位: ミリ秒

Amazon Location Service のメトリクスを見る

Amazon Location Service メトリックスは、Amazon CloudWatchコンソールまたは Amazon CloudWatch API を使って表示することができます。

CloudWatch コンソールを使用してメトリクスを表示するには

Example

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで メトリクスを選択します。
3. すべてのメトリクス タブで、Amazon Location 名前空間を選択してください。
4. 表示するメトリクスのタイプを選択します。
5. グラフに追加するメトリクスを選択します。

詳細については、「Amazon ユーザーガイド」の「使用可能なメトリクスの表示」を参照してください。 CloudWatch

Amazon Location Service メトリクスの CloudWatch アラームを作成する

を使用して CloudWatch 、Amazon Location Service メトリクスにアラームを設定できます。例えば、 でアラームを作成して、エラー数が急増するたびに E CloudWatch メールを送信できます。

以下のトピックでは、 CloudWatch を使用してアラームを設定する方法の概要について説明します。詳細な手順については、「Amazon CloudWatch ユーザーガイド」の「アラームの使用」を参照してください。

CloudWatch コンソールを使用してアラームを設定するには

Example

1. https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインで、アラーム を選択します。
3. アラームの作成(アラームの作成) を選択します。
4. メトリクスの選択 を選択します。
5. すべてのメトリクス タブで、Amazon Location 名前空間を選択してください。
6. メトリクスカテゴリを選択してくだい。
7. アラームを作成したいメトリクスのあるその行の横にあるチェックボックスを選択してください。
8. メトリクスの選択 を選択します。
9. メトリクスに値を入力してください。
10. アラームの条件を指定してください。
11. 次へ をクリックします。

12.アラーム条件が満たされたときに通知を送信したい場合:

- アラーム状態のトリガーで、通知の送信を促すアラーム状態を選択します。
- SNS トピックの選択で 新しいトピックの作成を選択し、新しい Amazon Simple Notification Service (Amazon SNS) トピックを作成します。トピック名と通知の送信先のメールを入力します。
- 通知を送信で、通知の送信先となる追加のメールアドレスを入力します。
- 通知を追加 をクリックします。このリストは保存され、今後のアラーム用のフィールドに表示されます。

13終了したら、次へを選択します。

14.アラームの名前と説明を入力し、次へを選択します。

15.アラームの詳細を確認して、次へを選択します。

> **Note**
>
> 新しいAmazon SNSトピックを作成する場合、E メールアドレスを検証しなければ、そのアドレスで通知を受け取ることができません。メールが確認されていない場合、状態の変化によりアラームが開始されても、通知は受信されません。

CloudWatch コンソールを使用してアラームを設定する方法の詳細については、「Amazon CloudWatch ユーザーガイド」の「E メールを送信するアラームを作成する」を参照してください。

CloudWatch を使用したクォータに対する使用状況のモニタリング

Amazon CloudWatch アラームを作成して、特定のクォータの使用率が設定可能なしきい値を超えたときに通知することができます。これにより、割り当ての限界に近づいたことを認識し、コストオーバーを避けるために利用率を調整するか、必要であれば割り当ての増額を要求することができる。CloudWatch を使用してクォータをモニタリングする方法については、「Amazon CloudWatch ユーザーガイド」の「サービスクォータの視覚化とアラームの設定」を参照してください。

CloudWatch Amazon Location Service の メトリクスの例

GetMetricData API を使用して、Amazon Location のメトリクスを取得できます。

- 例えば、`CallCount`をモニタリングして、数値が低下したときのアラームを設定することができます。

SendDeviceLocationのCallCountメトリクスをモニタリングすることで、追跡対象資産の全体像を把握しやすくなります。CallCount値が下がった場合は、トラックのフリートなどの追跡対象資産が現在の位置の送信を停止したことを意味します。このアラームを設定することで、問題が発生したことを知らせるのに役立ちます。

- 別の例では、ErrorCountをモニタリングして、数値の急上昇が起こったときにアラームを設定することができます。

  デバイスの位置をジオフェンスと照合して評価するには、トラッカーをジオフェンスコレクションに関連付ける必要があります。継続的な位置情報の更新を必要とするデバイスフリートがある場合、BatchEvaluateGeofenceまたはBatchPutDevicePositionのCallCountがゼロになるのは、更新がもはや流れていないことを表示します。

以下は、 CallCountのメトリクスGetMetricDataとマップリソースの作成ErrorCountに関する の出力例です。

```
{
  "StartTime": 1518867432,
  "EndTime": 1518868032,
  "MetricDataQueries": [
    {
      "Id": "m1",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Location",
          "MetricName": "CallCount",
          "Dimensions": [
            {
              "Name": "SendDeviceLocation",
              "Value": "100"
            }
          ]
        },
        "Period": 300,
        "Stat": "SampleCount",
        "Unit": "Count"
      }
    },
    {
      "Id": "m2",
      "MetricStat": {
```

```
        "Metric": {
          "Namespace": "AWS/Location",
          "MetricName": "ErrorCount",
          "Dimensions": [
            {
              "Name": "AssociateTrackerConsumer",
              "Value": "0"
            }
          ]
        },
        "Period": 1,
        "Stat": "SampleCount",
        "Unit": "Count"
      }
    }
  ]
}
```

でのログ記録とモニタリング

AWS CloudTrail は、ユーザー、ロール、または サービスによって実行されたアクションを記録するAWSサービスです。 は、すべての API コールをイベントとして CloudTrail 記録します。で Amazon Location Service を使用して API コールを CloudTrail モニタリングできます。これには、Amazon Location Service コンソールからの呼び出しと Amazon Location Service API オペレーションへの AWS SDK 呼び出しが含まれます。

証跡を作成するときに、Amazon Location Service の CloudTrail イベントなど、S3 バケットへのイベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、コンソールのイベント履歴 で最新の CloudTrail イベントを表示できます。 で収集された情報を使用して CloudTrail、Amazon Location Service に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の詳細については CloudTrail、「 AWS CloudTrailユーザーガイド」を参照してください。

トピック

- の Amazon Location Service 情報 CloudTrail
- Amazon Location Service ログファイルエントリの理解

の Amazon Location Service 情報 CloudTrail

CloudTrail AWSアカウントを作成すると、 がアカウントで有効になります。Amazon Location Service でアクティビティが発生すると、そのアクティビティは CloudTrail イベント履歴 の他の AWSサービスイベントとともにイベントに記録されます。AWS アカウントで最近のイベントを表示、検索、ダウンロードできます。詳細については、「イベント履歴 での CloudTrail イベントの表示」を参照してください。

Amazon Location Service のイベントなどの、AWS アカウントにおけるイベントを継続的に記録するには、追跡を作成します。証跡により、 はログファイル CloudTrail を S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。証跡では、AWS パーティションのすべてのリージョンからのイベントがログに記録され、指定した S3 バケットにログファイルが配信されます。さらに、 CloudTrail ログで収集されたデータをより詳細に分析し、それに基づく対応を行うように他の AWSサービスを設定できます。

詳細については、次を参照してください:

- 証跡を作成するための概要
- CloudTrail サポートされているサービスと統合
- の Amazon SNS 通知の設定 CloudTrail
- 複数のリージョンからの CloudTrail ログファイルの受信と複数のアカウントからの CloudTrail ログファイルの受信

すべての Amazon Location Service アクションは によってログに記録 CloudTrail され、「Amazon Location Service API リファレンス」に記載されています。例えば、、、および DescribeTrackerアクションを呼び出すUpdateTrackerとCreateTracker、 CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、リクエストの生成者に関する情報が含まれます。この ID 情報は、リクエストがどのようにして送信されたかを確認するのに役立ちます:

- ルートまたは AWS Identity and Access Management (IAM) ユーザーの認証情報を使用して行われたか。
- ロールまたはフェデレーティッドユーザーの一時的なセキュリティ認証情報を使用して行われたか。
- 別の AWS サービスによって行われたか。

詳細については、「CloudTrail userIdentity 要素」を参照してください。

Amazon Location Service ログファイルエントリの理解

証跡は、指定した S3 バケットまたは Amazon CloudWatch Logs にイベントをログファイルとして配信できるようにする設定です。詳細については、「 ユーザーガイド」の CloudTrail 「ログファイルの操作AWS CloudTrail」を参照してください。

CloudTrail ログファイルには、1 つ以上のログエントリが含まれます。各イベントは任意の送信元からの単一のリクエストを表し、リクエストされたオペレーション、オペレーションの日時、リクエストパラメーターなどに関する情報を含みます。

> **Note**
>
> CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、特定の順序では表示されません。操作の順序を決定するには、`eventTime`を使用します。

次の例は、トラッカーリソースを作成する `CreateTracker`オペレーションを示す CloudTrail ログエントリを示しています。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "123456789012",
        "arn": "arn:aws:geo:us-east-1:123456789012:tracker/ExampleTracker"
        "accountId": "123456789012",
        "accessKeyId": "123456789012",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "123456789012",
                "arn": "arn:aws:geo:us-east-1:123456789012:tracker/ExampleTracker",
                "accountId": "123456789012",
                "userName": "exampleUser",
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
```

```
                "creationDate": "2020-10-22T16:36:07Z"
            }
        }
    },
    "eventTime": "2020-10-22T17:43:30Z",
    "eventSource": "geo.amazonaws.com",
    "eventName": "CreateTracker",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0/24—TEST-NET-1",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.864
 Linux/4.14.193-110.317.amzn2.x86_64 OpenJDK_64-Bit_Server_VM/11.0.8+10-LTS java/11.0.8
 kotlin/1.3.72 vendor/Amazon.com_Inc. exec-env/AWS_Lambda_java11",
    "requestParameters": {
        "TrackerName": "ExampleTracker",
        "Description": "Resource description"
    },
    "responseElements": {
        "TrackerName": "ExampleTracker",
        "Description": "Resource description"
        "TrackerArn": "arn:partition:service:region:account-id:resource-id",
        "CreateTime": "2020-10-22T17:43:30.521Z"
    },
    "requestID": "557ec619-0674-429d-8e2c-eba0d3f34413",
    "eventID": "3192bc9c-3d3d-4976-bbef-ac590fa34f2c",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012",
}
```

以下に、デバイスイベントの詳細を返す DescribeTracker アクションのログエントリを示します。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "123456789012",
        "arn": "arn:partition:service:region:account-id:resource-id",
        "accountId": "123456789012",
        "accessKeyId": "123456789012",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
```

```
                "principalId": "123456789012",
                "arn": "arn:partition:service:region:account-id:resource-id",
                "accountId": "123456789012",
                "userName": "exampleUser",
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-10-22T16:36:07Z"
            }
        }
    },
    "eventTime": "2020-10-22T17:43:33Z",
    "eventSource": "geo.amazonaws.com",
    "eventName": "DescribeTracker",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0/24—TEST-NET-1",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.864
 Linux/4.14.193-110.317.amzn2.x86_64 OpenJDK_64-Bit_Server_VM/11.0.8+10-LTS java/11.0.8
 kotlin/1.3.72 vendor/Amazon.com_Inc. exec-env/AWS_Lambda_java11",
    "requestParameters": {
        "TrackerName": "ExampleTracker"
    },
    "responseElements": null,
    "requestID": "997d5f93-cfef-429a-bbed-daab417ceab4",
    "eventID": "d9e0eebe-173c-477d-b0c9-d1d8292da103",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012",
}
```


AWS CloudFormation を使用して Amazon Location Service リソースを作成する

Amazon Location Service は AWS CloudFormation と統合されています。これは、リソースとインフラストラクチャの作成と管理の所要時間を短縮できるように AWS リソースをモデル化して設定するためのサービスです。Amazon Location リソースを含む必要なすべての AWS リソースを説明するテンプレートを作成すれば、AWS CloudFormation がユーザーに代わってこれらのリソースのプロビジョニングや設定を処理します。

AWS CloudFormation を使用すると、テンプレートを再利用して Amazon Location リソースをいつでも繰り返しセットアップできます。リソースを一度記述すると、同じリソースを複数の AWS アカウントおよびリージョンで何度でも繰り返してプロビジョニングできます。

Amazon Location と AWS CloudFormation テンプレート

Amazon Location および関連サービスのリソースをプロビジョニングしてセットアップするには、AWS CloudFormation テンプレートについて理解しておく必要があります。テンプレートは、JSON や YAML でフォーマットされたテキストファイルです。これらのテンプレートには、AWS CloudFormation スタックにプロビジョニングしたいリソースを記述します。JSON や YAML に不慣れな方は、AWS CloudFormation Designer を使えば、AWS CloudFormation テンプレートを使いこなすことができます。詳細については、「AWS CloudFormation ユーザーガイド」の「AWS CloudFormation Designer とは」を参照してください。

Amazon Location は、AWS CloudFormation で次のリソースタイプの作成をサポートします。

- AWS::Location::Map
- AWS::Location::PlaceIndex
- AWS::Location::RouteCalculator
- AWS::Location::Tracker
- AWS::Location::TrackerConsumer
- AWS::Location::GeofenceCollection

リソースの JSON テンプレートと YAML テンプレートの例を含む詳細情報については、「AWS CloudFormation ユーザーガイド」の「Amazon Location Service リソースタイプのリファレンス」を参照してください。

AWS CloudFormation の詳細はこちら

AWS CloudFormation の詳細については、以下のリソースを参照してください。

- AWS CloudFormation
- AWS CloudFormation ユーザーガイド
- AWS CloudFormation API リファレンス
- AWS CloudFormation コマンドラインインターフェイスユーザーガイド

Amazon Location Service のセキュリティ

のクラウドセキュリティが最優先事項 AWS です。お客様は AWS 、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 とユーザー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およびクラウド内のセキュリティと説明しています。

- 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を担います AWS クラウド。また、 は、安全に使用できるサービス AWS も提供します。コンプライアンスAWS プログラムコンプライアンスプログラム の一環として、サードパーティーの監査者は定期的にセキュリティの有効性をテストおよび検証。Amazon Location Service 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、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- クラウドのセキュリティ — お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、Amazon Location 使用時における責任共有モデルの適用法を理解するのに役立ちます。以下のトピックでは、セキュリティとコンプライアンスの目標を達成するように Amazon Location を設定する方法について説明します。また、Amazon Location 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

- Amazon Location Service のベストプラクティス

Amazon Location Service でのデータ保護

責任 AWS 共有モデル、Amazon Location Service でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバル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。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービス のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーのよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿されたAWS 責任共有モデルおよび GDPRのブログ記事を参照してください。

データ保護の目的で、 認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須であり TLS 1.3 がお勧めです。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、または SDK を使用して Amazon Location AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs 名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

データプライバシー

Amazon Location Service を使用すると、組織のデータを引き続き管理できます。Amazon Location は、顧客メタデータとアカウント情報を削除することで、データプロバイダーに送信されるすべてのクエリを匿名化します。

Amazon Location は、追跡やジオフェンシングにデータプロバイダーを使用しません。つまり、機密データは AWS アカウントに残ります。これにより、施設、資産、人員の所在地などの機密性の高い位置情報を第三者から保護し、ユーザーのプライバシーを保護し、アプリケーションのセキュリティリスクを軽減できます。

詳細については、「AWS データプライバシーのよくある質問」を参照してください。

Amazon Location でのデータ保持

以下の特性は、Amazon Location がサービスのデータを収集して保存する方法に関係します。

- Amazon Location Service Trackers — Trackers API を使用してエンティティの位置を追跡すると、その座標を保存できます。デバイスの位置情報は、サービスによって削除されるまで 30 日間保存されます。
- Amazon Location Service Geofences — Geofences API を使用して対象地域を定義すると、サービスは指定したジオメトリを保存します。これらは明示的に削除する必要があります。

> **Note**
>
> AWS アカウントを削除すると、アカウント内のすべてのリソースが削除されます。詳細については、「AWS データプライバシーのよくある質問」を参照してください。

Amazon Location Service の保管中のデータ暗号化

Amazon Location Service は、デフォルトで暗号化を提供し、 AWS 所有の暗号化キーを使用して保管中の顧客の機密データを保護します。

- AWS 所有キー — Amazon Location は、デフォルトでこれらのキーを使用して、個人を特定できるデータを自動的に暗号化します。 AWS 所有キーを表示、管理、使用したり、その使用を監査したりすることはできません。ただし、データを暗号化するキーを保護するためのアクションの実施やプログラムの変更を行う必要はありません。詳細については、AWS Key Management Service デベロッパーガイドの「AWS 所有キー」を参照してください。

保管中のデータをデフォルトで暗号化することで、機密データの保護におけるオーバーヘッドと複雑な作業を減らすのに役立ちます。同時に、セキュリティを重視したアプリケーションを構築して、暗号化のコンプライアンスと規制の厳格な要件を満たすことができます。

この暗号化レイヤーを無効にしたり、代替の暗号化タイプを選択したりすることはできませんが、トラッカーおよびジオフェンスコレクションリソースを作成するときにカスタマーマネージドキーを選択することで、既存の AWS 所有の暗号化キーに 2 つ目の暗号化レイヤーを追加できます。

- カスタマーマネージドキー — Amazon Location は、ユーザーが作成、所有、管理する対称カスタマーマネージドキーの使用をサポートし、既存の AWS 所有暗号化に 2 番目の暗号化レイヤーを追加します。この暗号化層はユーザーが完全に制御できるため、次のようなタスクを実行できます。
  - キーポリシーの策定と維持
  - IAM ポリシーとグラントの策定と維持
  - キーポリシーの有効化と無効化
  - キー暗号化マテリアルのローテーション
  - タグの追加
  - キーエイリアスの作成
  - キー削除のスケジュール設定

  詳細については、AWS Key Management Service デベロッパーガイドの「カスタマーマネージドキー」を参照してください。

次の表は、Amazon Location が個人を特定できるデータをどのように暗号化するかをまとめたものです。

データ型	AWS 所有キーの暗号化	カスタマーマネージドキーの暗号化 (オプション)
`Position` デバイス位置の詳細 を含むポイントジオメトリ。	有効	有効
`PositionProperties` 位置の更新に関連付けられたキーと値のペアのセット。	有効	有効

データ型	AWS 所有キーの暗号化	カスタマーマネージドキーの暗号化 (オプション)
`GeofenceGeometry` ジオフェンスされたエリアを表すポリゴンジオフェンスジオメトリ。	有効	有効
`DeviceId` デバイス位置更新をトラッカーリソースにアップロードする際に指定されたデバイス識別子。	有効	サポートされていません
`GeofenceId` ジオフェンスジオメトリ、または特定のジオフェンスコレクション内のジオフェンスのバッチを格納するときに指定される識別子。	有効	サポートされていません

> **Note**
>
> Amazon Location では、個人を特定できるデータを無料で保護するために、 AWS 所有キーを使用した保管時の暗号化が自動的に有効になります。
> ただし、カスタマーマネージドキーの使用には AWS KMS 料金が適用されます。料金の詳細については、「AWS Key Management Service 料金」を参照してください。

の詳細については AWS KMS、「 とは」を参照してください AWS Key Management Service。

Amazon Location Service が で許可を使用する方法 AWS KMS

Amazon AppIntegrations には、カスタマー管理キーを使用するためのグラントが必要です。

カスタマーマネージドキーで暗号化されたトラッカーリソースまたはジオフェンスコレクションを作成すると、Amazon Location は にCreateGrantリクエストを送信してユーザーに代わって許可を作成します AWS KMS。の許可 AWS KMS は、顧客アカウントの KMS キーへのアクセス権を Amazon Location に付与するために使用されます。

このグラントは、Amazon Location が、以下の内部オペレーションでカスタマーマネージドキーを使用するために必要です。

- トラッカーまたはジオフェンスコレクションの作成時に入力された対称カスタマーマネージド KMS キー ID が有効であることを確認するDescribeKeyリクエストを AWS KMS に送信します。
- カスタマーマネージドキーで暗号化されたデータキーを生成する AWS KMS には、にGenerateDataKeyWithoutPlaintextリクエストを送信します。
- Decrypt リクエストを AWS KMS に送信して、暗号化されたデータキーを復号し、データの暗号化に使用できます。

任意のタイミングで、許可に対するアクセス権を取り消したり、カスタマーマネージドキーに対するサービスからのアクセス権を削除したりできます。これを行うと、Amazon Location はカスタマーマネージドキーによって暗号化されたすべてのデータにアクセスできなくなり、そのデータに依存しているオペレーションが影響を受けます。例えば、Amazon Location がアクセスできない暗号化されたトラッカーから デバイスの位置を取得 しようとすると、操作によって `AccessDeniedException` エラーが返されます。

カスタマーマネージドキーを作成する

対称カスタマーマネージドキーを作成するには AWS Management Console、、、または AWS KMS APIsを使用します。

対称カスタマーマネージドキーを作成するには

AWS Key Management Service デベロッパーガイド にある 対称カスタマーマネージドキーの作成 ステップに従います。

キーポリシー

キーポリシーは、カスタマーマネージドキーへのアクセスを制御します。すべてのカスタマーマネージドキーには、キーポリシーが 1 つだけ必要です。このポリシーには、そのキーを使用できるユーザーとその使用方法を決定するステートメントが含まれています。カスタマーマネージドキーを作成する際に、キーポリシーを指定することができます。詳細については、AWS Key Management

Service デベロッパーガイドの「カスタマーマネージドキーへのアクセスの管理」を参照してください。

Amazon Location リソースでカスタマーマネージドキーを使用するには、キーポリシーで次の API オペレーションを許可する必要があります。

- kms:CreateGrant - カスタマーマネージドキーに許可を追加します。この権限は、指定された KMS キーへのアクセスを制御します。これにより、必要なグラントオペレーションに対し Amazon Location がアクセスできるようにします。詳細については、AWS Key Management Service デベロッパーガイド の「許可の使用」を参照してください。

  これにより、Amazon Location で次のことが可能になります。

  - GenerateDataKeyWithoutPlainText を呼び出して、暗号化されたデータキーを生成して保存します。データキーは暗号化にすぐには使用されないからです。
  - Decrypt を呼び出して、保存された暗号化データキーを使用して暗号化データにアクセスします。
  - 退職するプリンシパルを設定して、RetireGrant にサービスがそれを許可するようにします。
- kms:DescribeKey — カスタマーマネージドキーの詳細を提供し、サービスがキーを検証できるようにします。

Amazon Location に追加できるポリシーステートメントの例を以下に示します。

```
"Statement" : [
  {
    "Sid" : "Allow access to principals authorized to use Amazon Location",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "*"
    },
    "Action" : [
      "kms:DescribeKey",
      "kms:CreateGrant"
    ],
    "Resource" : "*",
    "Condition" : {
      "StringEquals" : {
        "kms:ViaService" : "geo.region.amazonaws.com",
        "kms:CallerAccount" : "111122223333"
      }
  },
```

```
    {
      "Sid": "Allow access for key administrators",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
       },
      "Action" : [
        "kms:*"
       ],
      "Resource": "arn:aws:kms:region:111122223333:key/key_ID"
    },
    {
      "Sid" : "Allow read-only access to key metadata to the account",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "arn:aws:iam::111122223333:root"
      },
      "Action" : [
        "kms:Describe*",
        "kms:Get*",
        "kms:List*",
        "kms:RevokeGrant"
      ],
      "Resource" : "*"
    }
  ]
```

ポリシーでの許可の指定に関する詳細については、「AWS Key Management Service デベロッパーガイド」を参照してください。

キーアクセスのトラブルシューティング については、AWS Key Management Service デベロッパーガイド を参照してください。

Amazon Location のカスタマーマネージドキーの指定

カスタマーマネージドキーは、以下のリソースのセカンドレイヤー暗号化として指定できます。

- トラッカーリソース
- ジオフェンスコレクション

リソースを作成する場合、KMS ID を入力してデータキーを指定できます。Amazon Location は、リソースに保存されている識別可能な個人データを暗号化するために使用します。

- KMS ID — AWS KMS カスタマーマネージドキーのキー識別子。キー ID、キー ARN、エイリアス名、またはエイリアス ARN を入力します。

Amazon Location Service の暗号化コンテキスト

暗号化コンテキストは、データに関する追加のコンテキスト情報が含まれたキーバリューペアのオプションのセットです。

AWS KMS は、認証された暗号化 をサポートするために、暗号化コンテキストを追加の認証データとして使用します。データを暗号化するリクエストに暗号化コンテキストを含めると、 は暗号化コンテキストを暗号化されたデータに AWS KMS バインドします。データを復号化するには、そのリクエストに (暗号化時と) 同じ暗号化コンテキストを含めます。

Amazon Location Service の暗号化コンテキスト

Amazon Location は、すべての暗号化オペレーションで同じ AWS KMS 暗号化コンテキストを使用します。キーは `aws:geo:arn`で、値はリソース Amazon リソースネーム (ARN) です。

Example

```
"encryptionContext": {
    "aws:geo:arn": "arn:aws:geo:us-west-2:111122223333:geofence-collection/SAMPLE-
GeofenceCollection"
}
```

モニタリングに暗号化コンテキストを使用する

対称カスタマーマネージドキーを使用してトラッカーまたはジオフェンスコレクションを暗号化する場合は、監査レコードとログで暗号化コンテキストを使用して、カスタマーマネージドキーがどのように使用されているかを特定することもできます。暗号化コンテキストは、 AWS CloudTrail または Amazon CloudWatch Logs によって生成されたログにも表示されます。

暗号化コンテキストを使用してカスタマーマネージドキーへのアクセスを制御する

`conditions` が対称カスタマーマネージドキーへのアクセスを制御するための条件として、キーポリシーと IAM ポリシー内の暗号化コンテキストを使用することもできます。付与する際に、暗号化コンテキストの制約を使用することもできます。

Amazon Location は、権限で暗号化コンテキスト制約を使用して、アカウントまたはリージョン内のカスタマーマネージドキーへのアクセスを制御します。権限の制約では、権限によって許可されるオペレーションで指定された暗号化コンテキストを使用する必要があります。

Example

次に、特定の暗号化コンテキストのカスタマーマネージドキーへのアクセスを付与するキーポリシーステートメントの例を示します。このポリシーステートメントの条件では、権限に暗号化コンテキストを指定する暗号化コンテキスト制約が必要です。

```
{
    "Sid": "Enable DescribeKey",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/ExampleReadOnlyRole"
     },
     "Action": "kms:DescribeKey",
     "Resource": "*"
},
{
     "Sid": "Enable CreateGrant",
     "Effect": "Allow",
     "Principal": {
         "AWS": "arn:aws:iam::111122223333:role/ExampleReadOnlyRole"
     },
     "Action": "kms:CreateGrant",
     "Resource": "*",
     "Condition": {
         "StringEquals": {
             "kms:EncryptionContext:aws:geo:arn": "arn:aws:geo:us-
west-2:111122223333:tracker/SAMPLE-Tracker"
          }
     }
}
```

Amazon Location Service の暗号化キーを監視する

Amazon Location Service リソースで AWS KMS カスタマーマネージドキーを使用する場合、 AWS CloudTrailまたは Amazon CloudWatch Logs を使用して、Amazon Location が に送信するリクエストを追跡できます AWS KMS。

次の例はCreateGrant、Amazon Location によって呼び出された KMS オペレーションをモニタリングDescribeKeyして、カスタマーマネージドキーによって暗号化されたデータにアクセスするための Decrypt、、、および GenerateDataKeyWithoutPlainTextの AWS CloudTrail イベントです。

CreateGrant

AWS KMS カスタマーマネージドキーを使用してトラッカーまたはジオフェンスコレクションリソースを暗号化すると、Amazon Location はユーザーに代わって AWS アカウントの KMS キーにアクセスするCreateGrantリクエストを送信します。Amazon Location が作成する権限は、 AWS KMS　カスタマーマネージドキーに関連付けられたリソースに固有のものです。さらに、Amazon Location は、リソースを削除するときに付与を削除する　RetireGrant　オペレーションを使用します。

以下のイベント例では CreateGrant オペレーションを記録しています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-04-22T17:02:00Z"
            }
        },
        "invokedBy": "geo.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "retiringPrincipal": "geo.region.amazonaws.com",
```

```
        "operations": [
            "GenerateDataKeyWithoutPlaintext",
            "Decrypt",
            "DescribeKey"
        ],
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "granteePrincipal": "geo.region.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE"
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

GenerateDataKeyWithoutPlainText

トラッカーまたはジオフェンスコレクションリソースの AWS KMS カスタマーマネージドキーを有効にすると、Amazon Location は一意のテーブルキーを作成します。リソースの AWS KMSカスタマーマネージドキー AWS KMS を指定するGenerateDataKeyWithoutPlainTextリクエストを に送信します。

以下のイベント例では GenerateDataKeyWithoutPlainText オペレーションを記録しています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
```

```
        "type": "AWSService",
        "invokedBy": "geo.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "encryptionContext": {
            "aws:geo:arn": "arn:aws:geo:us-west-2:111122223333:geofence-collection/
SAMPLE-GeofenceCollection"
        },
        "keySpec": "AES_256",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "57f5dbee-16da-413e-979f-2c4c6663475e"
}
```


Decrypt

暗号化されたトラッカーまたはジオフェンスコレクションにアクセスすると、Amazon Location は `Decrypt` オペレーションを呼び出し、保存されている暗号化されたデータキーを使用して暗号化されたデータにアクセスします。

以下のイベント例では Decrypt オペレーションを記録しています。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "geo.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:10:51Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "encryptionContext": {
            "aws:geo:arn": "arn:aws:geo:us-west-2:111122223333:geofence-collection/
SAMPLE-GeofenceCollection"
        },
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "dc129381-1d94-49bd-b522-f56a3482d088"
}
```

DescribeKey

Amazon Location は DescribeKey オペレーションを使用して、トラッカーまたはジオフェンスコレクションに関連付けられている AWS KMS カスタマーマネージドキーがアカウントとリージョンに存在するかどうかを確認します。

以下のイベント例では DescribeKey オペレーションを記録しています。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-04-22T17:02:00Z"
            }
        },
        "invokedBy": "geo.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "keyId": "00dd0db0-0000-0000-ac00-b0c000SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
```

```
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```


詳細はこちら

次のリソースは、保管時のデータ暗号化についての詳細を説明しています。

- AWS Key Management Service 基本概念の詳細については、「AWS Key Management Service デベロッパーガイド」を参照してください。
- のセキュリティのベストプラクティスの詳細については、「 AWS Key Management ServiceAWS Key Management Service デベロッパーガイド」を参照してください。

Amazon Location Service の転送中のデータの暗号化

Amazon Location は、Transport Layer Security (TLS) 1.2 暗号化プロトコルを使用してすべてのネットワーク間データを自動的に暗号化することにより、サービスに出入りする転送中のデータを保護します。Amazon Location Service API に送信されるダイレクト HTTPS リクエストは、AWS 署名バージョン 4 アルゴリズム を使用して署名され、安全な接続を確立します。

Amazon Location Service での 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、誰が Amazon Location リソースの使用を認証 (サインイン) および承認 (アクセス許可を付与する) できるかを制御します。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、Amazon Location 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 Amazon Location Service を使用する場合は、管理者から必要なアクセス許可と認証情報が与えられます。さらに多くの Amazon Location 機能を使用して作業を行うには、追加のアクセス許可が必要になることがあります。アクセスの管理方法を理解しておくと、管理者に適切な許可をリクエストするうえで役立ちます。Amazon Location の機能にアクセスできない場合は、「Amazon Location Service Identity and Access のトラブルシューティング」を参照してください。

サービス管理者 - 社内の Amazon Location リソースを担当している場合は、通常、Amazon Location へのフルアクセスがあります。サービスのユーザーがどの Amazon Location 機能や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 Amazon Location と IAM を併用する方法の詳細については、「Amazon Location Service と IAM の連携」を参照してください。

IAM 管理者 - 管理者は、Amazon Location へのアクセス権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 で使用可能な、Amazon Location アイデンティティベースのポリシーの例を確認するには、「Amazon Location Service のアイデンティティベースのポリシーの例」を参照してください。

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（ にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center（ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook の認証情報は、フェデレーティッド ID の例です。フェデレーティッド 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 AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 ユーザーガイド」の「 へのサインイン AWS アカウント方法AWS サインイン 」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。推奨される方法を使用してリクエストを自分で署名する方法の詳細については、IAM ユーザーガイドの API AWS リクエストの署名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、 AWS では、多要素認証 (MFA) を使用してアカウントのセキュリティを向上させる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「Multi-factor authentication」(多要素認証) および「IAM ユーザーガイド」の「AWSでの多要素認証 (MFA) の使用」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 AWS のサービス およびリソースへの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に、一時的な認証情報を使用して にアクセスするための ID プロバイダーとのフェデレーションの使用を要求 AWS のサービス 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、Identity Center ディレクトリのユーザー、または ID ソースを通じて提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID がにアクセスすると 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て、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、ロールでは一時的な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 (ロールではなく) の作成が適している場合」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。ロール を切り替える AWS Management Console ことで、 で IAM ロールを一時的に引き受けることができます。ロールを引き受けるには、 または AWS API AWS CLI オペレーションを呼び出すか、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「IAM ロールの使用」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詳細については、「IAM ユーザーガイド」の「Creating a role for a third-party Identity Provider」(サードパーティーアイデンティティプロバイダー向けロールの作成) 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 ユーザーガイド」の「アクセス許可セット」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) ポリシーをリソースに直接アタッチできます。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 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 でアクションを実行する場合 AWS、ユーザーは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を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストリクエストリクエストと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 のサービスにアクセス許可を委任するロールの作成」を参照してください。
- サービスにリンクされたロール – サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

- Amazon 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 リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。 AWS ロールを EC2 インスタンスに割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時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。

IAM ロールと IAM ユーザーのどちらを使用するかについては、IAM ユーザーガイドの(IAM ユーザーではなく) IAM ロールをいつ作成したら良いのか?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 AWS には、ポリシーを作成し、 AWS ID またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 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 AWS JSON 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んなリソースにどんな条件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継ぐことができます。

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 ポリシーの作成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 ユーザーガイドのマネージドポリシーとインラインポリシーの比較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ます。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースのポリシーによって 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) に対する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、 AWS アカウント ビジネスが所有する複数の をグループ化して一元管理するサービスです。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「SCP の仕組み」を参照してください。
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もあります。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、IAM ユーザーガイドのセッションポリシーを参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関与する場合に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 Location Service と IAM の連携

IAM を使用して Amazon Location へのアクセスを管理する前に、Amazon Location で使用できる IAM 機能について理解しておく必要があります。

Amazon Location Service で利用できる IAM 機能

IAM 機能	Amazon Location サポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	なし
ポリシーアクション	あり
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	なし
ABAC (ポリシー内のタグ)	はい
一時的な認証情報	あり
プリンシパル権限	いいえ
サービスロール	いいえ
サービスリンクロール	なし

Amazon Location 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。

Amazon Location 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ポリシーをサポートする	あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、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 ポリシーの作成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。

Amazon Location のアイデンティティベースのポリシーの例

Amazon Location のアイデンティティベースポリシーの例を確認するには、「Amazon Location Service のアイデンティティベースのポリシーの例」を参照してください。

Amazon Location 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 にある場合 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。

Amazon Location のポリシーアクション

ポリシーアクションに対するサポート	あり

管理者は AWS JSON 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んなリソースにどんな条件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、関連付けられた 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権限を付与するポリシーで使用されます。

Amazon Location アクションのリストを確認するには、「サービス認証リファレンス」の「Amazon Location Service で定義されるアクション」を参照してください。

Amazon Location のポリシーアクションは、アクションの前にプレフィックスを使用します。

```
geo
```

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": [
      "geo:action1",
      "geo:action2"
         ]
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、Get という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "geo:Get*"
```

Amazon Location のアイデンティティベースポリシーの例を確認するには、「Amazon Location Service のアイデンティティベースのポリシーの例」を参照してください。

Amazon Location のポリシーリソース

ポリシーリソースに対するサポート	あり

管理者は AWS JSON 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どのような条件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": "*"
```

Amazon Location リソースのタイプとその ARN 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 Location Service で定義されるリソース」を参照してください。どのアクションで各リソースの ARN を指定できるかについては、「Amazon Location Service」定義されるアクションを参照してください。

Amazon Location のアイデンティティベースポリシーの例を確認するには、「Amazon Location Service のアイデンティティベースのポリシーの例」を参照してください。

Amazon Location のポリシー条件キー

サービス固有のポリシー条件キーのサポート	あり

管理者は AWS JSON 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んなリソースにどんな条件でアクションを実行できるかということです。

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

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

条件を指定する際にプレースホルダー変数も使用できます。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。

Amazon Location での条件キーの一覧については、「サービス認証リファレンス」の「Amazon Location Service 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 Location Service で定義されるアクション」を参照してください。

Amazon Location では、ポリシーステートメント内の特定のジオフェンスまたはデバイスへのアクセスを許可または拒否できる条件キーをサポートしています。次の条件キーが使用可能です。

- ジオフェンス アクションで使用する `geo:GeofenceIds`。タイプは `ArrayOfString` です。
- トラッカーアクションで使用する `geo:DeviceIds`。タイプは `ArrayOfString` です。

IAM ポリシーの `geo:GeofenceIds` では、次のアクションを使用できます。

- BatchDeleteGeofences
- BatchPutGeofences
- GetGeofence
- PutGeofence

IAM ポリシーの geo:DeviceIds では、次のアクションを使用できます。

- BatchDeleteDevicePositionHistory
- BatchGetDevicePosition
- BatchUpdateDevicePosition
- GetDevicePosition
- GetDevicePositionHistory

> **Note**
>
> これらの条件キー
> はBatchEvaluateGeofences、ListGeofences、ListDevicePositionまたはアクションでは使用できません。

Amazon Location のアイデンティティベースポリシーの例を確認するには、「Amazon Location Service のアイデンティティベースのポリシーの例」を参照してください。

Amazon Location での ACL

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コントロールします。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。

ABAC と Amazon Location

ABAC のサポート (ポリシー内のタグ)	はい

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アクセス許可戦略です。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 とは?を参照してください。ABAC をセットアップするステップ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に基づく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。

Amazon Location リソースのタグ付けの詳細については、「Amazon Location Service リソースにタグを付ける」を参照してください。

リソースのタグに基づいてリソースへのアクセスを制限するためのアイデンティティベースポリシーの例を表示するには、「タグに基づいて リソースへのアクセスを制御する」を参照してください。

Amazon Location 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 は、一時的な認証情報を使用してサインインすると機能 AWS のサービス しません。一時的な認証情報 AWS のサービス を使用する などの詳細については、IAM ユーザーガイドのAWS のサービス「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、一時的な認証情報を使用します。例えば、会社の Single Sign-On (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 ユーザーガイドのロールへの切り替え (コンソール)を参照してください。

一時的な認証情報は、 AWS CLI または 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 . AWS recommends にアクセスできます AWS。これは、長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成することを推奨しています。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。

Amazon Location 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) をサポート	なし

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を 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と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。

Amazon Location のサービスロール

サービスロールのサポート	いいえ

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 のサービスにアクセス許可を委任するロールの作成」を参照してください。

> **Warning**
>
> サービスロールの許可を変更すると、Amazon Location の機能が破損する可能性があります。Amazon Location 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。

Amazon Location のサービスリンクロール

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。

サービスリンク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。表の中から、[Service-linked role] (サービスにリンクされたロール) 列に Yes と記載されたサービスを見つけます。サービスリンク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。

認証されていないユーザーに対する Amazon Location Service の仕組み

ウェブやモバイルアプリケーションでのマップ表示など、Amazon Location Service を使用する多くのシナリオでは、IAM でサインインしていないユーザーにアクセスを許可する必要があります。このような認証されていないシナリオでは、2 つのオプションがあります。

- API キーを使用する — 認証されていないユーザーにアクセス権を付与するには、Amazon Location Service リソースへの読み取り専用アクセスを許可する API キーを作成できます。これは、すべてのユーザーを認証したくない場合に便利です。例えば、Web アプリケーションなどです。API キーに関する詳細については、「API キーを使用してアプリケーションへの認証されていないゲストアクセスを許可する」を参照してください。
- Amazon Cognito を使用する — API キーに代わる方法として、Amazon Cognito を使用して匿名アクセスを許可する方法があります。Amazon Cognito では、認証されていないユーザーが実行できる操作を定義する IAM ポリシーを使用して、より詳細な認証を作成できます。Amazon Cognito の使用方法の詳細については、「Amazon Cognito を使用して、アプリケーションで認証されていないゲストアクセスを許可する方法」を参照してください。

認証されていないユーザーにアクセス権を与える方法の概要については、「Amazon Location Service へのアクセスを許可する」を参照してください。

Amazon Location Service 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 Amazon Location リソースを作成または変更するアクセス許可がありません。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受けることができます。

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 ポリシーの作成」を参照してください。

Amazon Location が定義するアクションとリソースタイプ (リソースタイプごとの ARN の形式を含む) の詳細については、サービス認証リファレンス の「Amazon Location Service のアクション、リソース、および条件キー」を参照してください。

トピック

- ポリシーのベストプラクティス
- Amazon Location コンソールの使用
- 自分の権限の表示をユーザーに許可する
- ポリシーでの Amazon Location Service リソースの使用
- デバイスの位置を更新する権限
- トラッカーリソースの読み取り専用ポリシー
- ジオフェンスの作成に関するポリシー
- ジオフェンスの読み取り専用ポリシー
- マップリソースをレンダリングする権限
- 検索操作を許可する権限
- ルート計算ツールの読み取り専用ポリシー
- 条件キーに基づいてリソースアクセスを制御する
- タグに基づいて リソースへのアクセスを制御する

ポリシーのベストプラクティス

ID ベースのポリシーは、ユーザーのアカウント内で誰かが Amazon Location 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、 などの特定の を介してサービスアクションが使用される場合に 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 policy elements: Condition] (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 ポリシーの検証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 保護 API アクセスの設定を参照してください。

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

Amazon Location コンソールの使用

Amazon Location Service コンソールにアクセスするには、許可の最小限のセットが必要です。これらのアクセス許可により、 の Amazon Location リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーとロールが Amazon Location コンソールを使用できるようにするには、次のポリシーをエンティティにアタッチします。詳細については、「IAM ユーザーガイド」の「ユーザーへの許可の追加」を参照してください。

以下のポリシーでは、Amazon Location Service コンソールにアクセスして、 AWS アカウント内の Amazon Location リソースの詳細を作成、削除、一覧表示、表示できるようにします。

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

または、読み取り専用アクセス権限を付与して、読み取り専用アクセスを容易にすることもできます。読み取り専用権限では、ユーザーがリソースの作成や削除などの書き込み操作を試みると、エラーメッセージが表示されます。例については、「the section called “トラッカーの読み取り専用ポリシー”」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザー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": "*"
        }
    ]
}
```

ポリシーでの Amazon Location Service リソースの使用

Amazon Location Service リソースに以下のプレフィックスを使用します。

Amazon Location リソースプレフィックス

リソース	リソースプレフィックス
マップリソース	`map`
プレースリソース	`place-index`
ルートリソース	`route-calculator`
トラッキングリソース	`tracker`
ジオフェンスコレクションのリソース	`geofence-collection`

次の ARN 構文を使用します。

```
arn:Partition:geo:Region:Account:ResourcePrefix/ResourceName
```

ARN の形式の詳細については、「Amazon リソースネーム (ARNs AWS 「サービス名前空間」を参照してください。

例

- 次の ARN を使用して、指定されたマップリソースへのアクセスを許可します。

  ```
  "Resource": "arn:aws:geo:us-west-2:account-id:map/map-resource-name"
  ```

- 特定のアカウントに属するすべての `map` リソースへのアクセスを指定するには、ワイルドカード (*) を使用します。

  ```
  "Resource": "arn:aws:geo:us-west-2:account-id:map/*"
  ```

- リソースの作成を含む、一部の Amazon Location アクションは、特定のリソースで実行できません。このような場合は、ワイルドカード *を使用する必要があります。

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

Amazon Location リソースのタイプとその ARN 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 Location Service で定義されるリソース」を参照してください。どのアクショ

ンで各リソースの ARN を指定できるかについては、「Amazon Location Service」 定義されるアクションを参照してください。

デバイスの位置を更新する権限

複数のトラッカーのデバイス位置を更新するには、1 つ以上のトラッカーリソースへのアクセス権をユーザーに付与する必要があります。また、ユーザーがデバイスの位置を一括更新できるようにしたいとします。

この例では、*Tracker1* リソースと *Tracker2* リソースへのアクセスを許可することに加えて、次のポリシーにより、*Tracker1* リソースと *Tracker2* リソースに対する geo:BatchUpdateDevicePosition アクションを使用する権限も許可されます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UpdateDevicePositions",
      "Effect": "Allow",
      "Action": [
        "geo:BatchUpdateDevicePosition"
      ],
      "Resource": [
        "arn:aws:geo:us-west-2:account-id:tracker/Tracker1",
        "arn:aws:geo:us-west-2:account-id:tracker/Tracker2"
      ]
    }
  ]
}
```

ユーザーが特定のデバイスのデバイス位置のみを更新できるように制限したい場合は、そのデバイス ID に条件キーを追加できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UpdateDevicePositions",
      "Effect": "Allow",
      "Action": [
        "geo:BatchUpdateDevicePosition"
      ],
```

```
      "Resource": [
        "arn:aws:geo:us-west-2:account-id:tracker/Tracker1",
        "arn:aws:geo:us-west-2:account-id:tracker/Tracker2"
      ],
      "Condition":{
        "ForAllValues:StringLike":{
          "geo:DeviceIds":[
            "deviceId"
          ]
        }
      }
    }
  ]
}
```

トラッカーリソースの読み取り専用ポリシー

AWS アカウント内のすべてのトラッカーリソースの読み取り専用ポリシーを作成するには、すべてのトラッカーリソースへのアクセスを許可する必要があります。また、複数の端末の端末位置の取得、単一の端末からの端末位置の取得、位置履歴の取得を可能にするアクションへのアクセス権をユーザーに付与する必要があります。

この例では、次のポリシーは次のアクションにアクセス権限を付与します。

- geo:BatchGetDevicePosition を使用して複数のデバイスの位置を取得します。
- geo:GetDevicePosition を使用して 1 つのデバイスの位置を取得します。
- geo:GetDevicePositionHistory を使用してデバイスの位置履歴を取得します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetDevicePositions",
      "Effect": "Allow",
      "Action": [
        "geo:BatchGetDevicePosition",
        "geo:GetDevicePosition",
        "geo:GetDevicePositionHistory"
      ],
      "Resource": "arn:aws:geo:us-west-2:account-id:tracker/*"
    }
```

```
  ]
}
```

ジオフェンスの作成に関するポリシー

ユーザーにジオフェンスの作成を許可するポリシーを作成するには、ユーザーがジオフェンスコレクションに 1 つ以上のジオフェンスを作成できるようにする特定のアクションへのアクセスを許可する必要があります。

以下のポリシーは、*Collection* での以下のアクションに対する権限を付与します。

- geo:BatchPutGeofence を使用して複数のジオフェンスを作成します。
- geo:PutGeofence を使用して 1 つのジオフェンスを作成します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateGeofences",
      "Effect": "Allow",
      "Action": [
        "geo:BatchPutGeofence",
        "geo:PutGeofence"
      ],
      "Resource": "arn:aws:geo:us-west-2:account-id:geofence-collection/Collection"
    }
  ]
}
```

ジオフェンスの読み取り専用ポリシー

AWS アカウントのジオフェンスコレクションに保存されているジオフェンスの読み取り専用ポリシーを作成するには、ジオフェンスを保存しているジオフェンスコレクションから読み取るアクションへのアクセスを許可する必要があります。

以下のポリシーは、*Collection* での以下のアクションに対する権限を付与します。

- geo:ListGeofences は、指定されたジオフェンスコレクション内のジオフェンスを一覧表示します。
- geo:GetGeofence は、ジオフェンスコレクションからジオフェンスを取得します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetGeofences",
      "Effect": "Allow",
      "Action": [
        "geo:ListGeofences",
        "geo:GetGeofence"
      ],
      "Resource": "arn:aws:geo:us-west-2:account-id:geofence-collection/Collection"
    }
  ]
}
```

マップリソースをレンダリングする権限

マップをレンダリングするための十分な権限を付与するには、マップタイル、スプライト、グリフ、スタイル記述子へのアクセスを許可する必要があります。

- geo:GetMapTile は、マップ上のフィーチャを選択的にレンダリングするために使用するマップタイルを取得します。
- geo:GetMapSprites は、PNG スプライトシートと、その中のオフセットを記述した対応する JSON ドキュメントを取得します。
- geo:GetMapGlyphs は、テキストの表示に使用されるグリフを取得します。
- geo:GetMapStyleDescriptor は、レンダリングルールを含むマップのスタイル記述子を取得します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetTiles",
      "Effect": "Allow",
      "Action": [
        "geo:GetMapTile",
        "geo:GetMapSprites",
        "geo:GetMapGlyphs",
        "geo:GetMapStyleDescriptor"
      ],
```

```
      "Resource": "arn:aws:geo:us-west-2:account-id:map/Map"
    }
  ]
}
```

検索操作を許可する権限

検索オペレーションを許可するポリシーを作成するには、まず AWS アカウント内の場所インデックスリソースへのアクセスを許可する必要があります。また、ユーザーがジオコーディングでテキストを使用して検索したり、リバースジオコーディングで位置を使用して検索したりできるようにするアクションへのアクセス権を付与する必要があります。

この例では、 へのアクセスを許可するだけでなく*PlaceIndex*、次のポリシーは、次のアクションへのアクセス許可も付与します。

- geo:SearchPlaceIndexForPosition は、特定の位置の近くの場所や興味のあるポイントを検索できます。
- geo:SearchPlaceIndexForText は、自由形式のテキストを使用して、住所、名前、都市、または地域を検索できます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Search",
      "Effect": "Allow",
      "Action": [
        "geo:SearchPlaceIndexForPosition",
        "geo:SearchPlaceIndexForText"
      ],
      "Resource": "arn:aws:geo:us-west-2:account-id:place-index/PlaceIndex"
    }
  ]
}
```

ルート計算ツールの読み取り専用ポリシー

ユーザーがルート計算リソースにアクセスしてルートを計算できるようにする読み取り専用ポリシーを作成できます。

この例では、 へのアクセスを許可するだけでなく*ExampleCalculator*、次のポリシーによって次のオペレーションへのアクセス許可が付与されます。

- geo:CalculateRoute は、出発位置、目的地位置、および経由地の位置のリストを考慮してルートを計算します。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RoutesReadOnly",
      "Effect": "Allow",
      "Action": [
        "geo:CalculateRoute"
      ],
      "Resource": "arn:aws:geo:us-west-2:accountID:route-calculator/ExampleCalculator"
    }
  ]
}
```

条件キーに基づいてリソースアクセスを制御する

ジオフェンスまたはデバイスポジションを使用するためのアクセスを許可する IAM ポリシーを作成する場合、条件演算子 を使用して、ユーザーがアクセスできるジオフェンスまたはデバイスをより正確に制御できます。そのためには、ポリシーの Condition 要素にジオフェンス ID またはデバイス ID を含める必要があります。

次のポリシー例は、ユーザーが特定のデバイスのデバイス位置を更新することを許可するポリシーの作成方法を示しています。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UpdateDevicePositions",
      "Effect": "Allow",
      "Action": [
        "geo:BatchUpdateDevicePosition"
      ],
      "Resource": [
        "arn:aws:geo:us-west-2:account-id:tracker/Tracker"
```

```
      ],
      "Condition":{
        "ForAllValues:StringLike":{
          "geo:DeviceIds":[
            "deviceId"
          ]
        }
      }
    }
  ]
}
```

タグに基づいて リソースへのアクセスを制御する

Amazon Location リソースを使用するアクセス権を付与する IAM ポリシーを作成する場合、属性ベースのアクセス制御 を使用して、ユーザーが変更、使用、または削除できるリソースをより適切に制御できます。そのためには、ポリシーの Condition 要素に タグ 情報を含めて、リソースタグに基づいてアクセスを制御します。

次のポリシー例は、ユーザーにジオフェンスを作成することを許可するポリシーの作成方法を示しています。これにより、*Collection* というジオフェンスコレクションに 1 つ以上のジオフェンスを作成する次のアクションを実行できる権限が付与されます。

- geo:BatchPutGeofence を使用して複数のジオフェンスを作成します。
- geo:PutGeofence を使用して 1 つのジオフェンスを作成します。

ただし、このポリシーでは、*Collection* タグ、Owner、にそのユーザーのユーザー名の値がある場合にのみ、 Condition 要素を使用してアクセス許可を付与します。

- 例えば、richard-roe という名前のユーザーが Amazon Location *Collection* を表示しようとすると、その *Collection* には Owner=richard-roe または owner=richard-roe のタグが付けられている必要があります。それ以外の場合、ユーザーはアクセスを拒否されます。

> **Note**
>
> 条件キー名では大文字と小文字が区別されないため、条件タグキー Owner は Owner と owner の両方に一致します。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。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateGeofencesIfOwner",
      "Effect": "Allow",
      "Action": [
        "geo:BatchPutGeofence",
        "geo:PutGeofence"
      ],
      "Resource": "arn:aws:geo:us-west-2:account-id:geofence-collection/Collection",
      "Condition": {
                "StringEquals": {"geo:ResourceTag/Owner": "${aws:username}"}
      }
    }
  ]
}
```

タグに基づいて AWS リソースにアクセスする権限を定義する方法のチュートリアルについては、AWS Identity and Access Management ユーザーガイド を参照してください。

Amazon Location Service Identity and Access のトラブルシューティング

Amazon Location と IAM の使用に伴って発生する可能性がある一般的な問題の診断や修復には、次の情報を利用してください。

トピック

- Amazon Location でアクションを実行する認可がありません
- iam を実行する権限がありません。PassRole
- 自分の 以外のユーザーに Amazon Location リソース AWS アカウント へのアクセスを許可したい

Amazon Location でアクションを実行する認可がありません

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

次のエラー例は、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geo:*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:
 geo:GetWidget on resource: my-example-widget
```

この場合、geo:*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。サインイン認証情報を提供した担当者が管理者です。

iam を実行する権限がありません。PassRole

iam:PassRole 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、ポリシーを更新して Amazon Location にロールを渡せるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

次の例のエラーは、marymajor という IAM ユーザーがコンソールを使用して Amazon Location でアクションを実行しようとする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可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。サインイン認証情報を提供した担当者が管理者です。

自分の 以外のユーザーに Amazon Location リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- Amazon Location がこれらの機能をサポートしているかどうかを確認するには、「Amazon Location Service と IAM の連携」を参照してください。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティー に提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイドの外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いについては、IAM ユーザーガイドの「IAM でのクロスアカウントリソースアクセス」を参照してください。

Amazon Location Service でのインシデントへの対応

AWSでは、セキュリティが最優先事項です。 AWS クラウド責任共有モデル の一環として、 は、セキュリティを最も重視する組織の要件を満たすデータセンターとネットワークアーキテクチャ AWS を管理します。お客様は AWS 、クラウドでセキュリティを維持する責任を共有します。つまり、ユーザーがアクセスできる AWS ツールや機能から実装するセキュリティを制御できます。

クラウド上で稼働するアプリケーションの目標を満たすセキュリティベースラインを確立することで、対応可能な逸脱を検出できます。セキュリティインシデント対応は複雑なトピックになる可能性があるため、インシデント対応 (IR) と選択肢が企業目標に与える影響をよりよく理解できるように、AWS Security Incident Response Guide、AWS Security Best Practices ホワイトペーパー、および AWS Cloud Adoption Framework (AWS CAF) のリソースを確認することをお勧めします。

Amazon Location Service でのログ記録とモニタリング

ログ記録とモニタリングはインシデントへの対応の重要な部分です。これにより、逸脱を検出するためのセキュリティベースラインを確立し、調査して対応することができます。Amazon Location Service のログ記録とモニタリングを実装することで、プロジェクトとリソースの信頼性、可用性、およびパフォーマンスを維持できます。

AWS には、インシデント対応のためのデータのログ記録と収集に役立つツールがいくつか用意されています。

AWS CloudTrail

Amazon Location Service は、ユーザー AWS CloudTrail、ロール、またはサービスによって実行されたアクションを記録する AWS サービスである と統合されます。これには、Amazon Location Service コンソールからのアクションと Amazon Location API オペレーションへのプログラムによる呼び出しが含まれます。これらのアクション記録はイベントと呼ばれます。詳細については、「 を使用した Amazon Location Service のログ記録とモニタリング AWS CloudTrail」を参照してください。

Amazon CloudWatch

Amazon を使用して CloudWatch 、Amazon Location Service アカウントに関連するメトリクスを収集および分析できます。 CloudWatch アラームを有効にして、メトリクスが特定の条件を満たし、指定されたしきい値に達した場合に通知できます。アラームを作成すると、CloudWatch は定義した Amazon Simple Notification Service に通知を送信します。詳細については、「Amazon による Amazon Location Service のモニタリング CloudWatch」を参照してください。

AWS Health ダッシュボード

AWS Health ダッシュボード を使用して、Amazon Location Service ステータスを確認できます。 AWS 環境に影響を与える可能性のあるイベントや問題に関する履歴データをモニタリングして表示することもできます。詳細については、「AWS Health ユーザーガイド」を参照してください。

Amazon Location Service のコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの範囲内にあるかどうかを確認するには、コンプライアンスプログラムAWS のサービス による対象範囲内のコンプライアンスプログラムを参照し、関心のあるコンプライアンスプログラムを選択します。一般的な情報については、AWS 「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「 でのレポートのダウンロード AWS Artifact」の」を参照してください。

を使用する際のお客様のコンプライアンス責任 AWS のサービス は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 は、コンプライアンスに役立つ以下のリソース AWS を提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境 AWS を にデプロイする手順について説明します。
- アマゾン ウェブ サービスにおける HIPAA セキュリティとコンプライアンスのアーキテクチャ — このホワイトペーパーでは、企業が AWS を使用して HIPAA 対象アプリケーションを作成する方法について説明します。

> **Note**
>
> すべて AWS のサービス HIPAA の対象となるわけではありません。詳細については、HIPAA 対応サービスのリファレンスを参照してください。

- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- AWS カスタマーコンプライアンスガイド – コンプライアンスの観点から責任共有モデルを理解します。このガイドでは、ガイダンスを保護し AWS のサービス 、複数のフレームワーク (米国国立標準技術研究所 (NIST)、Payment Card Industry Security Standards Council (PCI)、国際標準化機構 (ISO) を含む) のセキュリティコントロールにマッピングするためのベストプラクティスをまとめています。
- 「 デベロッパーガイド」の「 ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – これにより AWS のサービス 、 内のセキュリティ状態を包括的に確認できます AWS。Security Hub では、セキュリティコントロールを使用して AWS リソースを評価し、セキュリティ業界標準とベストプラクティスに対するコンプライアンスをチェックします。サポートされているサービスとコントロールのリストについては、Security Hub のコントロールリファレンスを参照してください。
- Amazon GuardDuty – これにより AWS アカウント、疑わしいアクティビティや悪意のあるアクティビティがないか環境を監視することで、、、ワークロード、コンテナ、データに対する潜在的な脅威 AWS のサービス を検出します。 GuardDuty は、特定のコンプライアンスフレームワークで義務付けられている侵入検知要件を満たすことで、PCI DSS などのさまざまなコンプライアンス要件への対応に役立ちます。
- AWS Audit Manager – これにより AWS のサービス 、 AWS 使用状況を継続的に監査し、リスクの管理方法と規制や業界標準への準拠を簡素化できます。

Amazon Location Service の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。 は、低レイテンシー、高スループット、および高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティーゾーン AWS リージョン を提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

AWS リージョン およびアベイラビリティーゾーンの詳細については、AWS 「 グローバルインフラストラクチャ」を参照してください。

Amazon Location には、 AWS グローバルインフラストラクチャに加えて、データの耐障害性とバックアップのニーズをサポートするのに役立つ機能がいくつか用意されています。

Amazon Location Service でのインフラストラクチャセキュリティ

マネージドサービスである Amazon Location Service は、 AWS グローバルネットワークセキュリティで保護されています。 AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS 「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して AWS 環境を設計するには、「 Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

が AWS 公開した API コールを使用して、ネットワーク経由で Amazon Location にアクセスします。クライアントは以下をサポートする必要があります:

- Transport Layer Security (TLS)。TLS 1.2 は必須で TLS 1.3 がお勧めです。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは、Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストには、アクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的なセキュリティ認証情報を生成し、リクエストに署名することもできます。

Amazon Location での設定と脆弱性の分析

設定と IT コントロールは、 AWS とお客様の間で共有される責任です。詳細については、 AWS 「 責任共有モデル」を参照してください。

サービス間の混乱した代理の防止

混乱した代理問題とは、アクションを実行する許可を持たないエンティティが、より高い特権を持つエンティティにそのアクションの実行を強制できるというセキュリティ問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、1 つのサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスは、本来ならアクセスすることが許可されるべきではない方法でその許可を使用して、別のお客様のリソースに対する処理を実行するように操作される場合があります。これを防ぐため、 AWS では、アカウント内のリソースへのアクセス権が付与されたサービスプリンシパルですべてのサービスのデータを保護するために役立つツールを提供しています。

Amazon Location Service は、ユーザーに代わって他の サービスへの呼び出し AWS サービスとして機能しないため、この場合、これらの保護を追加する必要はありません。混乱した代理について詳しくは、AWS Identity and Access Management ユーザーガイドの「混乱した代理の問題」を参照してください。

Amazon Location Service のセキュリティベストプラクティス

Amazon Location Service には、独自のセキュリティポリシーを策定および実装する際に考慮すべき、さまざまなセキュリティ機能が用意されています。以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションを説明するものではありません。これらのベストプラクティスはお客様の環境に必ずしも適切または十分でない可能性があるので、処方箋ではなく、あくまで有用な考慮事項とお考えください。

Amazon Location Service の探知用セキュリティのベストプラクティス

以下は、セキュリティ問題の検出に役立つ Amazon Location Service でのベストプラクティスです。

AWS モニタリングツールの実装

モニタリングはインシデントへの対応に不可欠であり、Amazon Location Service のリソースとソリューションの信頼性とセキュリティを維持します。AWS で利用できる複数のツールとサー

ビスのモニタリングツールを実装して、リソースと他の AWS のサービスをモニタリングできます。

例えば、Amazon CloudWatch では Amazon Location Service のメトリクスをモニタリングし、メトリクスが設定した特定の条件を満たし、定義したしきい値に達した場合に通知するアラームを設定できます。アラームを作成するときに、Amazon Simple Notification Service を使用してアラートの通知 CloudWatch を送信するように を設定できます。詳細については、「the section called “ログ記録とモニタリング”」を参照してください。

AWS ログ記録ツールを有効にする

Amazon Location Service は、ユーザー、ロール、または AWS のサービスによって実行されたアクションの記録を提供します。異常な API アクティビティを検出するアクションに関するデータを収集 AWS CloudTrail する などのログ記録ツールを実装できます。

証跡を作成するときに、イベントをログに記録する CloudTrail ように を設定できます。イベントは、Amazon Location に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などのリソースオペレーションの記録で、その他のデータが含まれます。詳細については、「 AWS CloudTrail ユーザーガイド」の「証跡のデータイベントのログ記録」を参照してください。

Amazon Location Service の予防的セキュリティベストプラクティス

以下は、セキュリティ問題の防止に役立つ Amazon Location Service でのベストプラクティスです。

安全な接続を使用

`https://` で始まる接続など、転送時に機密情報を安全に保つには、常に暗号化された接続を使用してください。

リソースへの最小特権アクセスの実装

Amazon Location リソースにカスタムポリシーを作成する場合は、タスクの実行に必要なアクセス許可のみを付与します。最小限の許可セットから開始し、必要に応じて追加許可を付与することをお勧めします。最小特権アクセスの実装は、エラーや悪意のある攻撃から生じる可能性のあるリスクと影響を軽減するために不可欠です。詳細については、「the section called “Identity and Access Management”」を参照してください。

グローバルにユニークな ID をデバイス ID として使用

デバイス ID には以下の規則を使用してください。

- デバイス ID は一意であることが必要です。
- デバイス ID は他のシステムの外部キーとして使用される可能性があるため、秘密にしないでください。
- デバイス ID には、電話デバイス ID やメールアドレスなどの個人を特定できる情報 (PII) を含めないでください。
- デバイス ID は予測可能であってはなりません。UUID のような不透明な識別子が推奨されます。

デバイスの位置プロパティには PII を含めないでください

デバイスの更新を送信する場合 (を使用するなどDevicePositionUpdate)、電話番号や E メールアドレスなどの個人を特定できる情報 (PII) を に含めないでくださいPositionProperties。

Amazon Location Service のベストプラクティス

このトピックでは、Amazon Location Service を使用するために、ベストプラクティスについて説明します。これらのベストプラクティスは Amazon Location Service 最大限に活用するのに役立ちますが、完全なソリューションではありません。ご使用の環境に該当するレコメンデーションのみに従うようにしてください。

トピック

セキュリティ

セキュリティリスクを管理または回避するには、次のベストプラクティスを検討してください。

- ID フェデレーションと IAM ロールを使用して、Amazon Location リソースへのアクセスを管理、制御、制限します。詳細については、「IAM ユーザーガイド」の「IAM のベストプラクティス」を参照してください。
- 最小特権の原則に従い、Amazon Location Service リソースへの必要最小限のアクセス権限のみを付与してください。詳細については、「the section called “ポリシーを使用したアクセスの管理”」を参照してください。

- ウェブアプリケーションで使用される Amazon Location Service リソースについては、`aws:referer` IAM 条件を使用してアクセスを制限し、許可リストに含まれているサイト以外のサイトによる使用を制限します。
- モニタリングおよびログ記録ツールを使用して、リソースのアクセスと使用状況を追跡します。詳細については、the section called “ログ記録とモニタリング”「」および「 AWS CloudTrail ユーザーガイド」の「証跡のデータイベントのログ記録」を参照してください。
- `https://` で始まる安全な接続を使用してセキュリティを強化し、サーバーとブラウザの間でデータが送信されている間の攻撃からユーザーを保護します。

探知用セキュリティと予防セキュリティのベストプラクティスの詳細については、「the section called “セキュリティに関するベストプラクティス”」を参照してください。

リソース管理

Amazon Location Service でロケーションリソースを効果的に管理するには、次のベストプラクティスを検討してください。

- 予想されるユーザーベースの中心となるリージョナルエンドポイントを使用して、ユーザーエクスペリエンスを向上させましょう。リージョンエンドポイントの詳細については、「Amazon Location のリージョンとエンドポイント」を参照してください。
- マップリソースやプレースインデックスリソースなど、データプロバイダーを使用するリソースについては、必ず特定のデータプロバイダーの利用規約に従ってください。詳細については、「データプロバイダー」を参照してください。
- マップ、Place Index 、ルートの構成ごとに 1 つのリソースを用意して、リソースの作成を最小限に抑えます。リージョン内では、通常、データプロバイダーまたはマップスタイルごとに必要なリソースは 1 つだけです。ほとんどのアプリケーションは既存のリソースを使用し、実行時にはリソースを作成しません。
- マップリソースやルート計算ツールなど、1 つのアプリケーションで異なるリソースを使用する場合は、データが一致するように各リソースで同じデータプロバイダーを使用してください。例えば、ルートカリキュレータで作成したルートジオメトリが、マップリソースを使用して描画されたマップ上の道路と一致するとします。

請求情報とコスト管理

コストと請求を管理するには、次のベストプラクティスを検討してください。

- Amazon などのモニタリングツールを使用して CloudWatch、リソースの使用状況を追跡します。使用量が指定した制限を超えそうになったときに通知するアラートを設定できます。詳細については、「Amazon CloudWatch ユーザーガイド」の「推定 AWS 料金を監視する請求アラームの作成」を参照してください。

クォータと使用量

使用量のデフォルト制限を設定するクォータ AWS アカウント を含めます。使用量が上限に近づいたときに通知するアラームを設定したり、必要なときにクォータの引き上げをリクエストしたりできます。クォータの使用方法については、次のトピックを参照してください。

- Amazon Location Service のサービスクォータ
- CloudWatch を使用したクォータに対する使用状況のモニタリング
- 「Amazon ユーザーガイド」の「サービスクォータの視覚化」と「アラームの設定 CloudWatch 」。

制限を超えそうになったときに事前に警告するアラームを作成できます。Amazon Location AWS リージョン を使用する各 で、クォータごとにアラームを設定することをお勧めします。例えば、`SearchPlaceIndexForText` オペレーションの使用状況をモニタリングし、現在のクォータの 80% を超えたらアラームを作成できます。

クォータに関するアラーム警告が表示されたら、何をすべきかを決める必要があります。顧客ベースが拡大したため、追加のリソースを使用している可能性があります。その場合は、そのリージョンでの API コールの割り当てを 50% 増やすなど、割り当ての増額をリクエストするとよいでしょう。あるいは、サービスにエラーがあり、Amazon Location に対して不必要な呼び出しを何度も行うことになる場合もあります。その場合は、サービスの問題を解決したいと思うでしょう。

ドキュメント履歴

以下の表は、Amazon Location Service ドキュメント内容を示します。更新に関する通知については、RSS フィードにサブスクライブできます。

変更	説明	日付
Amazon Location Service が用の新しい SDK をリリース JavaScript	Amazon Location アプリケーションをウェブフロントエンドで簡単に開発できるように、Amazon Location は AWS SDK for JavaScript v3 をサポートする新しいオープンソース SDK を追加し、認証を簡素化し、GeoJSON を使用します。詳細については、「Amazon Location SDK」を参照してください。	2023 年 7 月 6 日
Amazon Location Service が API キーの一般提供を開始	Amazon Location は、場所とルートのサポートを追加し、API キー機能の一般提供を発表しました。詳細については、「Using API keys」を参照してください。	2023 年 7 月 6 日
Amazon Location Service が位置更新用の Amazon EventBridge イベントを追加	Amazon Location では、トラッカーの位置更新イベントを に送信するためのサポートが追加されました EventBridge。トラッカーのイベントを有効にする方法などの詳細については、「 を使用したイベントへの対応 EventBridge」を参照してください。	2023 年 7 月 6 日

Amazon Location で、ジオフェンスにメタデータを追加	Amazon Location API を使用して、メタデータのプロパティをジオフェンスに追加できるようになりました。これらはジオフェンスとともに保存され、Amazon のジオフェンスに関連するイベントに含まれます EventBridge。詳細については、「ジオフェンスの描画」と「イベントルールの作成」を参照してください。	2023 年 6 月 15 日
Amazon Location で、場所のカテゴリを追加	Amazon Location に、場所検索結果のカテゴリが追加され、結果をカテゴリ別にフィルタリングできるようになりました。詳細については、「カテゴリとフィルタリング」を参照してください。	2023 年 6 月 15 日
Amazon Location で、政治的な見解を追加	Amazon Location に、特定のマップスタイル向けに政治的な見解が追加されました。詳細については、「政治的な見解」を参照してください。	2023 年 5 月 23 日
Amazon Location で、新しいデモサイトとサンプルサイトを発表	Amazon Location は、Amazon Location のデモとサンプルにアクセスできる新しいウェブサイトを発表しました。詳細については、「Amazon Location デモサイト」をご参照ください。	2023 年 5 月 3 日

Amazon Location が でより長いルートを導入 CalculateRouteMatrix	Amazon Location では、HERE データプロバイダーで作成されたルートマトリックスルートについて、長さに制限のないルートが許可されるようになりました。詳細については、「より長いルート計画」を参照してください。	2023 年 4 月 24 日
Amazon Location のドキュメントに、データプロバイダーごとの機能の違いを追加	Amazon Location のドキュメントが更新され、マップ、場所検索、ルーティングにおける各データプロバイダーの違いについて情報が追加されました。詳細については、「データプロバイダー別の機能」を参照してください。	2023 年 3 月 30 日
Amazon Location Open Data マップの一般利用状況	の OpenStreetMap夏地図に基づく Amazon Location Service データプロバイダーとスタイルの一般提供。詳細については、「Open Data」を参照してください。	2023 年 3 月 7 日
Amazon Location で、プレビュー版の新しい認証方法を追加	Amazon Location Service で、匿名ユーザーによる新しい認証方法としてプレビュー版の API キーが追加されました。詳細については、「API キーを使用して認証されていないゲストによるアプリケーションへのアクセスを許可する」を参照してください。	2023 年 2 月 23 日

Amazon Location のドキュメントを、最新の IAM ベストプラクティスに合わせて更新	Amazon Location Service のドキュメントに、最新の AWS Identity and Access Management ベストプラクティスに合わせて更新されました。詳細については、「Amazon Location Service のセキュリティ」を参照してください。	2023 年 1 月 26 日
Amazon Location Service が をデータプロバイダー GrabMaps として を南米に追加	Amazon Location では、が、南米のデータプロバイダー GrabMaps として導入されました。詳細については、「」を参照してくださいGrabMaps。	2023 年 1 月 10 日
Amazon Location Service Open Data 新マップのプレビュー版を追加	の新しい Amazon Location データプロバイダーとスタイルが、 OpenStreetMapの夏地図に基づいてパブリックプレビューに追加されました。詳細については、プレビューを参照してください。	2022 年 12 月 15 日
新しい HERE 衛星画像スタイル	HERE をデータプロバイダーとして使用するマップ用に、HERE 衛星画像スタイルと HERE ハイブリッドマップスタイルの 2 つの新しいマップスタイルが追加されました。詳細については、「HERE マップスタイル」を参照してください。	2022 年 10 月 25 日

住所の単位	Amazon Location Service で、「米国、エニータウン市、メインストリート 123 番地、アパートメント 3B」といった住所内の単位がサポートされるようになりました。	2022 年 9 月 20 日
ID で場所を取得	Amazon Location Service に、`GetPlace` オペレーションを使用して、`SearchPlaceIndexForSuggestions` オペレーションが提示した正確な位置を検索する機能が追加されました。詳細については、「オートコンプリートの使用」を参照してください。	2022 年 9 月 20 日
IAM ポリシーで追加の条件キー	Amazon Location Service では、IAM ポリシーで特定のジオフェンスまたはデバイスへのアクセスを設定できる追加の条件キーがサポートされるようになりました。「条件キー」を参照してください。	2022 年 8 月 23 日
円形ジオフェンス	Amazon Location Service では、中心点と半径で定義される円形ジオフェンスがサポートされるようになりました。これにより、デバイスが特定の場所から特定の距離内にあるときにイベントを取得できます。「円形ジオフェンスの追加」を参照してください。	2022 年 8 月 11 日

総合 API リファレンス	Amazon Location Service は、サブサービスごとに個別のガイドを、1 つの API リファレンスガイドにまとめて提供を開始しました。API の詳細については、「Amazon Location API」を参照してください。	2022 年 7 月 7 日
Service Quotas の統合	Amazon Location に、Service Quotas を統合しました。これにより、AWS Management Console または AWS CLI を使用してクオータを表示および管理できるようになりました。	2022 年 7 月 6 日
概念の章の更新	「Amazon Location の概念」の章が更新され、Amazon Location のユーザー向けの詳細情報が追加されました。	2022 年 4 月 22 日
新しい Android クイックスタートチュートリアル	Kotlin を使用した Android 開発用の新しい「クイックスタートチュートリアル」が追加されました。これにより、開発者がすぐに使い始めることができるようになりました。	2022 年 4 月 15 日
新しい HERE マップスタイル	HERE をデータプロバイダーとして使用するマップ用に 2 つの新しいマップスタイルが追加されました。詳細については、「HERE マップスタイル」を参照してください。	2022 年 3 月 15 日

ドキュメントを再構成し、コード例とチュートリアルを追加	この開発者ガイドに新しい「クイックスタート」章や「コード例」章などを追加し、トピックが見つけやすいよう再構成しました。	2022 年 2 月 25 日
トラッカー用の精度ベースの位置フィルター	トラッカーリソースを作成する際、精度ベースのフィルターを使用できるようになりました。	2021 年 12 月 7 日
場所インデックスのオートコンプリート	場所索引を検索する際、オートコンプリートを使用できるようになりました。	2021 年 12 月 6 日
マップを使用するための新しいAmplify チュートリアル	AWS Amplify を使用してWebアプリケーションでマップを表示する方法を示す新しいチュートリアルが追加されました。このチュートリアルは、「Amazon Location Service で Amplify ライブラリを使用する」でご覧いただけます。	2021 年 11 月 24 日
場所クエリエクステンション	Amazon Location Service では、ジオコーディングまたはリバースジオコーディング時に結果の優先言語を設定できるようになり、タイムゾーンやその他の情報が結果に追加されるようになりました。ジオコーディングとリバースジオコーディングの詳細については、「ジオコーディング、リバースジオコーディング、検索」を参照してください。	2021 年 11 月 16 日

トラッカー用の位置フィルター	Amazon Location Service では、コスト管理に役立つ新しい位置フィルター機能がトラッカーに追加されています。この機能は、更新がジオフェンスに対して保存または評価される前に、デバイスの位置更新を一部の除外します。位置フィルターの詳細については、「トラッカー」を参照してください。	2021 年 10 月 5 日
Update のオペレーション	Amazon Location Service API リファレンス に、UpdateMap 、、UpdatePlaceIndex、UpdateRouteCalculator UpdateGeofenceCollectionおよび のオペレーションが追加されましたUpdateTracker。	2021 年 7 月 19 日
チュートリアルの更新：Amazon Aurora PostgreSQL ユーザー定義関数	Amazon Location で Amazon Aurora PostgreSQL ユーザー定義関数を使用して地理空間データを検証、整理、エンリッチする方法を示す新しいチュートリアルが追加されました。	2021 年 7 月 19 日

AWS CloudFormation リソース	Amazon Location では、AWS CloudFormation リソース、、、、、および でのリソースタイプの作成がサポートされるようになりましたAWS::Location::GeofenceCollection 。 AWS::Location::Map AWS::Location::PlaceIndex AWS::Location::RouteCalculator AWS::Location::Tracker AWS::Location::TrackerConsumer	2021 年 6 月 7 日
リソースのタグ付け	Amazon Location リソースにタグを追加できるようになりました。タグを、リソースの管理、識別、整理、検索、フィルタリングに役立てることができるようになりました。	2021 年 6 月 1 日
一般提供	Amazon Location Service 開発者向けドキュメントの一般提供リリース：「リージョンおよびエンドポイント」、「Service Quotas」が更新されました。	2021 年 6 月 1 日

Esri Imagery	Amazon Location で、Esri Imagery と呼ばれるEsri マップスタイルの使用がサポートされるようになりました。詳細については、Esri Web サイトの「Esri World Imagery」を参照してください。	2021 年 6 月 1 日
ルートの計算	Amazon Location ルート計算ツールを使用して、選択したデータプロバイダーからの up-to-date 道路網とライブトラフィック情報に基づいてルートを計算し、移動時間を推定できるようになりました。	2021 年 6 月 1 日
AWS KMS 顧客管理キーによる保管中のデータの暗号化	Amazon Location では、お客様が作成、所有、管理する対称型顧客管理キーの使用がサポートされるようになりました。これにより、AWS の既存の暗号化に加えて暗号化レイヤーが追加されます。	2021 年 6 月 1 日
公開プレビューリリース	公開プレビュードキュメントのイニシャルリリース。	2020 年 12 月 16 日
チュートリアルの更新：マップの表示	MapLibre for Android および iOS を使用してマップを表示するためのチュートリアルが更新され、 MapLibre ネイティブ SDK を使用するようになりました。	2020 年 3 月 17 日

AWS 用語集

AWS の最新の用語については、「AWS の用語集リファレンス」の「AWS 用語集」を参照してください。

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。