



Developer Guide

Amazon Location Service



Amazon Location Service: Developer Guide

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is Amazon Location Service	1
Key features	1
Key benefits	4
Access high-quality geospatial data	4
Integrate seamlessly	5
Own your data	5
Reduce operating costs	5
Rapid development	6
Supported regions	6
Pricing model	9
Terms of use and data attribution	9
Data quality and coverage	11
Route Coverage definitions	11
Place Address definitions	11
Place Points-of-Interest (POI) definitions	11
Country/area data quality	12
Get started	28
Set up an account for the first time	28
Set up authentication for the first time	32
Choose an API	33
Create your first Maps and Places application	35
Create your first Geofences and Trackers application	51
iOS	51
Android	60
Authentication	70
Choose an authentication method	70
Use API keys	71
Create an API key	72
Call an API	74
Best practices	79
Use Amazon Cognito	79
Use Amazon Cognito and Amazon Location Service	80
Create an Amazon Cognito identity pool	83
Use the identity pool in web	84

Use IAM	85
Audience	86
Authenticating with identities	86
Managing access using policies	90
How Amazon Location Service works with IAM	92
How Amazon Location Service works with unauthenticated users	100
Identity-based policy examples	100
Troubleshooting	112
Maps	115
Features	115
Common use cases	116
Standalone Map APIs	117
Displaying Map	118
Map concepts	118
Maps terminology	119
Localization and internationalization	120
Map styles	123
Map styles overview	123
AWS map styles	123
Standard map style	124
Monochrome map style	125
Hybrid map style	125
Satellite map style	126
Standard map style	127
Monochrome map style	136
Hybrid map style	140
Map APIs	142
Maps	142
Dynamic maps	145
Static maps	159
How to	164
Dynamic maps	164
Static maps	209
Manage costs and usage	246
Best practices	247
Pricing	249

Quotas and usage	249
Places	252
What is a Place?	252
Features	1
Use cases	152
Places APIs overview	254
Places concepts	255
Places terminology	255
Querying and biasing	257
Filtering	258
Localization and internationalization	294
Contacts and opening hours	296
Additional features	298
IntendedUse	299
Places APIs	300
Places	300
Geocode	304
Reverse Geocode	307
Autocomplete	310
GetPlace	312
Search Nearby	314
Search Text	317
Suggest	319
How to	322
Use Geocode	323
Use ReverseGeocode	339
Use Autocomplete	351
Use Get Place	362
Use SearchNearby	367
Use SearchText	378
Use Suggest	388
Manage costs and usage	400
Best practices	400
Places pricing	401
Places quota and usage	402
Routes	405

Features	405
Use cases	408
APIs	409
Routes concepts	410
Terminology	411
Where (origin, destination, waypoint, and traces)	412
When (departure and arrival)	420
How (travel mode, avoidance, and exclusion)	422
Traffic awareness	429
Optimize route and waypoint sequence	430
Driver schedule and notices	431
Route APIs	434
Calculate routes	442
Calculate route matrix	448
Calculate isolines	449
Optimize waypoints	452
Snap to Roads	454
How to	455
Use CalculateRoutes	456
Calculate isolines	462
Calculate a route matrix	467
Optimize waypoints	477
Use SnapToRoads	485
Manage costs and usage	489
Best Practices	490
Routes pricing	492
Routes Quota and Usage	494
Geofences	498
Use cases	499
Geofence concepts	500
Common terminology	500
Get started	501
How to	502
Evaluate device positions against geofences	503
React to events with EventBridge	504
Manage resources	510

Manage costs and usage	517
Pricing	518
Quotas and usage	518
Trackers	521
Features	522
Use cases	524
Tracker concepts	525
Common terminology	525
Get started	527
Create a tracker	527
Create an unauthenticated identity pool	533
Update your tracker with a device position	535
Get a device's location history	537
List your device positions	538
How to	541
Verify positions	541
Link to a geofence collection	543
Track using AWS IoT and MQTT	545
Manage trackers	553
Manage costs and usage	557
Pricing	558
Quotas and usage	559
Developer tools	562
SDKs and frameworks	562
Developer tutorials	563
SDKs by language	591
Map Rendering SDK by language	595
API and CLI reference	597
API	597
CLI	597
Examples and Learning Resources	600
Demos	600
Samples	600
GitHub	601
Integrate with AWS	602
Resource Management	603

Manage quotas	604
Manage resources with Tags	622
Manage billing and costs	626
Create resources with AWS CloudFormation	626
Monitoring and Auditing	627
Monitor with Amazon CloudWatch	627
Monitor and log with AWS CloudTrail	635
Best practices	645
Resource management	645
Billing and cost management	646
Quotas and usage	646
Security	647
Data protection	648
Data privacy	649
Data retention	649
Data at rest encryption	649
Data in transit encryption	662
Incident response	662
Logging and Monitoring	663
Compliance validation	663
Resilience	665
Infrastructure security	665
Configuration and vulnerability analysis	665
Confused deputy prevention	666
Best practices	666
Security	666
Document history	669
AWS Glossary	670

What is Amazon Location Service

Amazon Location Service is a fully-managed service that makes it easy to add location functionality to your applications. Amazon Location uses high-quality geospatial data to provide capabilities such as maps, places, routes, trackers, and geofences. In addition to these capabilities, Amazon Location helps you maintain control of your data for privacy and security. It enables you to access location data simply with developer tools and move your applications to production faster with included monitoring and management capabilities.

This video provides an overview of the Amazon Location service.

[Introduction to Amazon Location Service](#)

With Amazon Location Service, you can add powerful location data and functionality to your applications. This includes capabilities such as maps, places (search and geocoding), routes, geofences, and trackers. Amazon Location provides location-based services (LBS) with high-quality data with global coverage. With its affordable data, geofences and trackers features, and built-in metrics for health monitoring, you can build sophisticated location-enabled applications.

With Amazon Location, you are in control of your organization's data. It anonymizes queries sent to data providers by removing customer metadata and account information. Moreover, sensitive tracking and geofences location information, such as facility, asset, and personnel locations, never leaves your AWS account. This approach helps protect sensitive information from third parties, safeguard user privacy, and reduce security risks for your application. Amazon Location also ensures that neither Amazon nor third parties have rights to sell your data or use it for advertising.

With Amazon Location Service, you can easily and seamlessly integrated your application with other AWS services, streamlining your development workflow and speeding up deployment. It works seamlessly with AWS CloudTrail, Amazon CloudWatch, Amazon EventBridge, and AWS Identity and Access Management. With this integration, you can easily manage resources and cost, monitor health metrics, and automatically respond to events, thanks to its built-in monitoring, security, and compliance features.

Key features of Amazon Location

Amazon Location offers a comprehensive set of features to enhance your location-based applications and services. This page provides an overview of the key capabilities available, including Maps, Places, Routes, and Geofences and Trackers. Leveraging these features, you can

build rich, location-aware experiences tailored to your specific use cases, whether it's delivering real-time location intelligence, enabling location-based services, or optimizing logistics and transportation operations.

Amazon Location provides the following features:

Maps

Amazon Location Service Maps lets you visualize location information and is the foundation of many location-based service capabilities. Amazon Location Service offers both dynamic and static maps.

Dynamic Maps allow you to create interactive maps using map tiles, with the option to use pre-built map styles such as standard, monochrome, hybrid, and satellite. You can stitch the Dynamic Maps content (Tiles, Styles, Glyphs, and Sprites) together using a map rendering engine, such as MapLibre.

Static Maps allow you to create pre-rendered, non-interactive map images that display a fixed geographical area to be embedded in applications without complex renderers.

For more information, see [Maps](#).

Places

Amazon Location Service Places lets you integrate search and geocode functionality into your application. You can:

- Use geocode (Forward and Reverse) to convert a place, such as an admin area, street, or address into geographic coordinates and vice versa.
- Use places search (Text and Nearby) to search for points of interest and get information on contact, access points, and opening hours.
- Use autocomplete or suggestions to autofill or predict an address or place based on user input. You can also use Get Place to get place details by place ID.

For more information, see [Places](#).

Routes

Amazon Location Service Routes lets you find routes, service area, and optimize and analyze routes. You can do the following:

- Estimate travel time and distance based on an up-to-date road network and live traffic information.
- Create a service area or isoline based on distance and time thresholds of your business need.
- Calculate a matrix (time and distance) for multiple origins and destinations and use the same for route planning.
- Align GPS traces to the nearest road segment, to improve accuracy of vehicle tracking and route visualization.
- Find the most efficient order to travel to multiple destinations, also known as solving the travelling salesman problem.

For more information, see [Routes](#).

Geofences

Amazon Location Service Geofences lets you give your application the ability to detect and act when a device enters or exits a defined geographical boundary known as a geofence. Automatically send an entry or exit event to Amazon EventBridge when a geofence breach is detected. This lets you initiate downstream actions, such as sending a notification to a target.

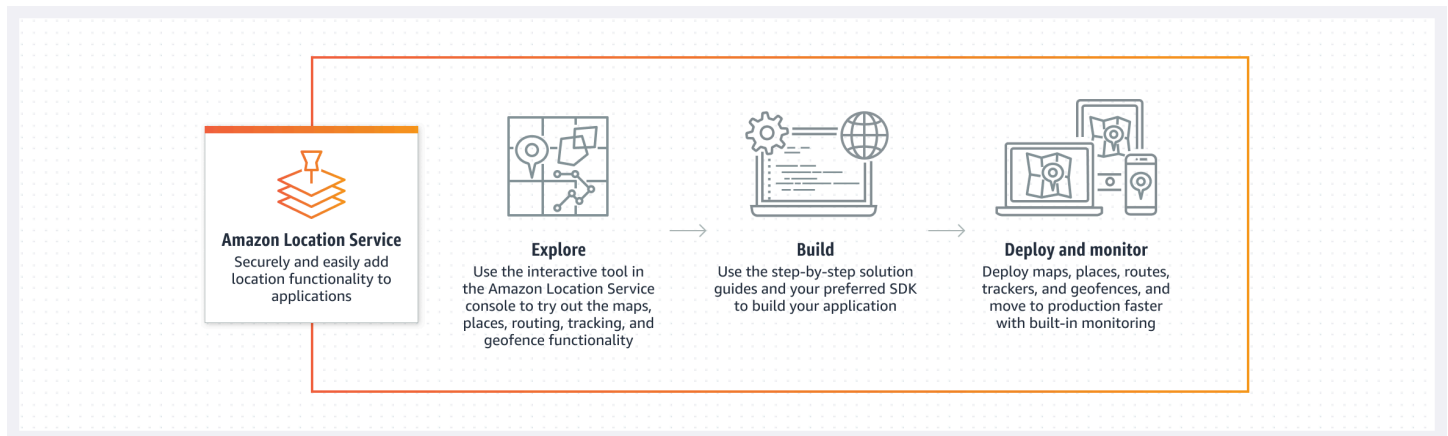
For more information, see [Geofences](#).

Trackers

Amazon Location Service Trackers let you retrieve the current and historical location of devices that are running your tracking-enabled application. You can also link trackers with Amazon Location Service geofences to evaluate location updates from your devices against your geofences automatically. Trackers can help you reduce costs by filtering position updates that haven't moved before storing or evaluating them against geofences.

For more information, see [Trackers](#).

Key benefits of Amazon Location Service



This section provides an overview of the key benefits of Amazon Location Service. The main benefits are:

- **Access high-quality geospatial data** - Integrate reliable, complete, and up-to-date data into applications to make informed decisions, deliver precise insights, and optimize operations.
- **Integrate seamlessly** - Quickly deploy and accelerate application development with seamless integration with AWS services. Amazon Location seamlessly integrates with various AWS services, enabling faster development, deployment, monitoring, and security compliance.
- **Own your data** - Safeguard sensitive customer information, protect user privacy and reduce your application's overall security risks. The service provides robust data protection measures, including data anonymization, encryption, and integration with AWS security services.
- **Reduce operating costs** - It offers high-quality geospatial data with a pay-as-you-go model, requiring no upfront commitment or minimum fee.
- **Rapid development** - The service allows for easy integration of geospatial data into applications, accelerating development cycles and streamlining integration with other AWS services.

Access high-quality geospatial data

Amazon Location Service offers a comprehensive solution for integrating geospatial capabilities into applications, focusing on four key benefits: seamless AWS integration, data ownership and privacy, cost reduction, and rapid development. By leveraging AWS infrastructure and security measures, the service enables you to quickly implement location-based features while maintaining control over your data and reducing operational costs.

Integrate seamlessly

With Amazon Location Service, you quickly deploy and accelerate application development with seamless integrations with AWS services. Amazon Location seamlessly integrates with a suite of AWS services to streamline development workflows, accelerate deployment, and provide robust monitoring, security, and compliance. It works with AWS CloudFormation for consistent resource provisioning, Amazon CloudWatch for tracking metrics, AWS CloudTrail for logging account activity, and Amazon EventBridge for event-driven architectures utilizing AWS Lambda. Amazon Location also leverages AWS Identity and Access Management (IAM) for secure access controls and resource tagging for efficient organization and management. This comprehensive integration enables faster development cycles, operational visibility, and adherence to security best practices.

For more information, see [Integrate with AWS](#).

Own your data

With Amazon Location Service, you protect user privacy, shield your sensitive information, and reduce your application's security risks. Amazon Location provides robust data control, rights, and secure access measures to ensure your organization's data remains private and secure. It anonymizes queries, encrypts sensitive location data at rest and in transit, and integrates with AWS Key Management Service (AWS KMS) so you can use your own encryption keys. You retain full rights to your data, with no third-party access for advertising or resale. Amazon Location also seamlessly integrates with AWS Identity and Access Management and Amazon Cognito for secure authentication and access controls, leveraging the proven security services at AWS to safeguard your data and applications.

For more information, see [Security](#).

Reduce operating costs

With Amazon Location Service, you access cost-effective, high-quality geospatial data from trusted data providers. Amazon Location requires no up-front commitment, and no minimum fee. You are only charged for what you use. Location data is billed based on each request your application makes to the service.

See the following case studies:

- [Halodoc, a digital health system, reduced 88% cost using Amazon Location Service](#)

- [Geo.me, a web services provider, reduced 90% geospatial cost using Amazon Location Service](#)

Rapid development

Amazon Location Service enables developers to quickly integrate geospatial data and functionality into their applications. By providing fully managed APIs, Amazon Location eliminates the need to build and maintain complex location data infrastructure. The service also seamlessly integrates with AWS services like AWS CloudFormation, CloudWatch, CloudTrail, and EventBridge, allowing developers to leverage these tools to build efficient, event-driven architectures. This integration streamlines the development process, accelerating the prototyping, iteration, and deployment of location-based experiences. With Amazon Location, organizations can rapidly enhance their offerings with geospatial features, reducing time-to-market and allowing developers to focus on core product development.

...Since it was so easy to get started and implement Amazon Location Service, our application was up and running in production in 2 weeks. Our ability to develop at this speed ensured readiness for the state department of transportation review date. We've done the review with the state and they approved our solution, we are now rolling it out to multiple company divisions for their upcoming state regulatory needs...

[Charles Evans, Chief Technology Officer at Command Alkon](#)

Amazon Location supported regions

Note

This topic documents the service quotas for the latest version of Amazon Location. For information on previous versions, refer to the [previous documentation](#).

Amazon Location Service is available across multiple AWS regions globally. This page lists the regions where the service is currently deployed and operational. It provides information to help you determine the most suitable region for your applications and workloads. The guide covers any potential variations or limitations in feature availability across regions. This information is crucial for making informed decisions regarding deployment locations, ensuring compliance with data residency requirements, and optimizing performance based on the proximity to end-users or operational centers.

Amazon Location is available in the following AWS Regions, using the listed endpoints.

For more information, see [AWS service endpoints](#).

Region name	Region code	Endpoint	Protocol	Available version
US East (Ohio)	us-east-2	geo.us-east-2.amazonaws.com	HTTPS	Current and Previous
US East (N. Virginia)	us-east-1	geo.us-east-1.amazonaws.com	HTTPS	Current and Previous
US West (Oregon)	us-west-2	geo.us-west-2.amazonaws.com	HTTPS	Current and Previous
Asia Pacific (Mumbai)	ap-south-1	geo.ap-south-1.amazonaws.com	HTTPS	Current and Previous
Asia Pacific (Singapore)	ap-southeast-1	geo.ap-southeast-1.amazonaws.com	HTTPS	Previous
Asia Pacific (Sydney)	ap-southeast-2	geo.ap-southeast-2.amazonaws.com	HTTPS	Current and Previous
Asia Pacific (Tokyo)	ap-northeast-1	geo.ap-northeast-1.amazonaws.com	HTTPS	Current and Previous

Region name	Region code	Endpoint	Protocol	Available version
Canada (Central)	ca-central-1	geo.ca-central-1.amazonaws.com	HTTPS	Current and Previous
Europe (Frankfurt)	eu-central-1	geo.eu-central-1.amazonaws.com	HTTPS	Current and Previous
Europe (Ireland)	eu-west-1	geo.eu-west-1.amazonaws.com	HTTPS	Current and Previous
Europe (London)	eu-west-2	geo.eu-west-2.amazonaws.com	HTTPS	Current and Previous
Europe (Spain)	eu-south-2	geo.eu-south-2.amazonaws.com	HTTPS	Current and Previous
Europe (Stockholm)	eu-north-1	geo.eu-north-1.amazonaws.com	HTTPS	Current and Previous
South America (São Paulo)	sa-east-1	geo.sa-east-1.amazonaws.com	HTTPS	Current and Previous
AWS GovCloud (US-West)	us-gov-west-1	geo.us-gov-west-1.amazonaws.com geo-fips.us-gov-west-1.amazonaws.com	HTTPS	Previous

Pricing model

Amazon Location Service requires no up-front commitment and no minimum fee. You're only charged for what you use. Amazon Location offers a free trial for the first three months of usage (usage quota apply). Location data is billed based on each request your application makes to the service. Beyond the Amazon Location Service free trial, you pay for the requests your application makes to the service. For more pricing information, visit our [pricing page](#).

Pricing Buckets

Each API may offer multiple features at different price points and the price may vary based on features you requested through request parameters.

For example, for `CalculateRoutes`, you'll be charged at **Core** price bucket when using `Car`, `Pedestrian`, or `Truck` mode. You'll be charged at **Advanced** price bucket when using other travel modes, such as `Scooter` mode. You'll be charged at **Premium** price bucket when you request for Toll Cost calculation.

The specific pricing bucket you'll be charged for will be returned in the `PricingBucket` response field. To review the pricing for each API and feature, see the Amazon Location Service [pricing page](#) for details on API feature, request parameters, and corresponding pricing. For more information, see the following topics:

- [the section called "Pricing"](#)
- [the section called "Places pricing"](#)
- [the section called "Routes pricing"](#)
- [the section called "Pricing"](#)
- [the section called "Pricing"](#)

Terms of use and data attribution

Before you use Amazon Location Service, be sure you can comply with all applicable legal requirements, including license terms applicable to the use of the provider.

- For more information about the AWS requirements, see Section 82 of the [AWS Service Terms](#).
- For more information about the SLAs, see [Amazon Location SLA](#).

Amazon Location Service leverages data from multiple sources, as shown below:

- HERE Technologies provides AWS customers access to base maps, geocoding, reverse geocoding, and street-level address data. Maps provided by HERE are constantly validated based on an average of 5 million daily updates to deliver over 900 mapping attributes and maintain accurate information.

[Learn more about HERE.](#)

- GrabMaps provides AWS customers access to various styles of map tiles, geocoding capabilities, reverse geocoding capabilities, and routing options. Today, GrabMaps powers more than 800 billion searches, suggestions, trips and trends each month in eight Southeast Asian countries.

[Learn more about GrabMaps.](#)

- Esri provides AWS customers access to various styles of map tiles, geocoding capabilities, reverse geocoding capabilities, and street-level address data. Today, Esri's technology is deployed in more than 350,000 organizations including 90 of the Fortune 100 companies.

[Learn more about Esri.](#)

- AWS provides customers access to global base maps built from the Daylight distribution of OpenStreetMap (OSM) and other open data sources. Map styles are professionally designed to support different applications and use cases, including logistics and delivery, and data visualization in web and mobile environments. With a community of over a million map makers, OSM is an open data project where contributors update more than 500,000 features per day.

[Learn more about Open Data.](#)

When using data from Amazon Location Service you shall provide End-Users of your products and/or services with notice, in a reasonably conspicuous manner (e.g., by inserting a reference in the End-User Terms), a link to this page, which highlights our data sources, and their sources for attribution. In addition, by default, attribution is shown on the bottom right corner of the map when using Amazon Location Service to render dynamic maps.

- [Esri's data source attributions](#)
- [GrabMaps' data source attributions](#)
- [HERE's data source attributions](#)

Data quality and coverage

Amazon Location Service has extensive global coverage of map data. All Amazon Location Map data quality is categorized as Comprehensive, Good, or Fair per country. Map data is categorized separately for Routing, Place Addresses, and Place Points-of-Interest (POI) data.

Route Coverage definitions

- **Comprehensive** - Complete coverage of the road network in a country. Providing the ability to calculate optimal routes between all addresses within a country.
- **Good** - Coverage of most roads within a country. Providing the ability to calculate routes between all addresses within a country.
- **Fair** - Coverage of roads in major cities and major roads between cities.

Place Address definitions

- **Comprehensive** - Complete coverage of all addresses within a country with full address details.
- **Good** - Coverage of addresses in major and select cities and streets.
- **Fair** - Coverage of major cities.
- **Unsupported** - No support for addresses within a country.

Place Points-of-Interest (POI) definitions

- **Comprehensive** - Coverage of the businesses to support an optimal search experience.
- **Good** - Coverage of businesses on a best effort basis.
- **Fair** - Only basic and high-profile landmark POI information available.
- **Unsupported** - No support for POIs within a country.

Country/area data quality

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
Afghanistan	AFG	AF	Fair	Fair	Unsupported
Albania	ALB	AL	Comprehensive	Comprehensive	Good
Algeria	DZA	DZ	Comprehensive	Comprehensive	Unsupported
American Samoa	ASM	AS	Good	Good	Unsupported
Andorra	AND	AD	Comprehensive	Comprehensive	Comprehensive
Angola	AGO	AO	Comprehensive	Comprehensive	Unsupported
Anguilla	AIA	AI	Fair	Good	Unsupported
Antarctica	ATA	AQ	Fair	Fair	Unsupported
Antigua And Barbuda	ATG	AG	Fair	Fair	Unsupported
Argentina	ARG	AR	Comprehensive	Comprehensive	Comprehensive
Armenia	ARM	AM	Good	Good	Unsupported
Aruba	ABW	AW	Good	Good	Unsupported
Australia	AUS	AU	Comprehensive	Comprehensive	Comprehensive
Austria	AUT	AT	Comprehensive	Comprehensive	Comprehensive
Azerbaijan	AZE	AZ	Comprehensive	Comprehensive	Unsupported

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
Bahamas	BHS	BS	Comprehensive	Comprehensive	Good
Bahrain	BHR	BH	Comprehensive	Comprehensive	Good
Bangladesh	BGD	BD	Comprehensive	Good	Unsupported
Barbados	BRB	BB	Good	Good	Unsupported
Belarus	BLR	BY	Comprehensive	Comprehensive	Comprehensive
Belgium	BEL	BE	Comprehensive	Comprehensive	Comprehensive
Belize	BLZ	BZ	Good	Good	Unsupported
Benin	BEN	BJ	Good	Good	Unsupported
Bermuda	BMU	BM	Good	Good	Unsupported
Bhutan	BTN	BT	Fair	Fair	Unsupported
Bolivia	BOL	BO	Comprehensive	Comprehensive	Unsupported
Bosnia And Herzegovina	BIH	BA	Comprehensive	Comprehensive	Comprehensive
Botswana	BWA	BW	Comprehensive	Comprehensive	Good
Brazil	BRA	BR	Comprehensive	Comprehensive	Comprehensive
British Indian Ocean Territory	IOT	IO	Fair	Fair	Unsupported

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
British Virgin Islands	VGB	VG	Fair	Fair	Unsupported
Brunei Darussala m	BRN	BN	Comprehensive	Comprehensive	Fair
Bulgaria	BGR	BG	Comprehensive	Comprehensive	Comprehensive
Burkina Faso	BFA	BF	Good	Good	Unsupported
Burundi	BDI	BI	Good	Good	Unsupported
Cambodia	KHM	KH	Comprehensive	Good	Fair
Cameroon	CMR	CM	Comprehensive	Comprehensive	Unsupported
Canada	CAN	CA	Comprehensive	Comprehensive	Comprehensive
Cape Verde	CPV	CV	Good	Good	Unsupported
Cayman Islands	CYM	KY	Comprehensive	Comprehensive	Good
Central African Republic	CAF	CF	Good	Good	Unsupported
Chad	TCD	TD	Good	Good	Unsupported
Chile	CHL	CL	Comprehensive	Comprehensive	Comprehensive
China	CHN	CN	Fair	Fair	Unsupported

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
Colombia	COL	CO	Comprehensive	Comprehensive	Fair
Comoros	COM	KM	Good	Good	Unsupported
Congo	COG	CG	Good	Good	Unsupported
Congo, The Democrati c Republic of	COD	CD	Good	Good	Unsupported
Cook Islands	COK	CK	Fair	Fair	Unsupported
Costa Rica	CRI	CR	Comprehensive	Comprehensive	Good
Croatia	CRV	HR	Comprehensive	Comprehensive	Comprehensive
Cuba	CUB	CU	Good	Good	Unsupported
Curacao	CUW	CW	Fair	Good	Unsupported
Cyprus	CYP	CY	Comprehensive	Comprehensive	Unsupported
Czechia	CZE	CZ	Comprehensive	Comprehensive	Comprehensive
Denmark	DNK	DK	Comprehensive	Comprehensive	Comprehensive
Djibouti	DJI	DJ	Good	Good	Unsupported
Dominica	DMA	DM	Fair	Fair	Unsupported
Dominican Republic	DOM	DO	Comprehensive	Comprehensive	Fair

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
Ecuador	ECU	EC	Good	Good	Unsupported
Egypt	EGY	EG	Comprehensive	Comprehensive	Fair
El Salvador	CLV	SV	Good	Good	Unsupported
Equatoria l Guinea	GNQ	GQ	Good	Good	Unsupported
Eritrea	ERI	ER	Good	Good	Unsupported
Estonia	EST	EE	Comprehensive	Comprehensive	Comprehensive
Eswatini	SWZ	SZ	Comprehensive	Comprehensive	Good
Ethiopia	ETH	ET	Comprehensive	Comprehensive	Unsupported
Falkland Islands	FLK	FK	Comprehensive	Comprehensive	Unsupported
Faroe Islands	FRO	FO	Comprehensive	Comprehensive	Unsupported
Fiji	FJI	FJ	Good	Good	Unsupported
Finland	FIN	FI	Comprehensive	Comprehensive	Comprehensive
France	FRA	FR	Comprehensive	Comprehensive	Comprehensive
French Guiana	GUF	GF	Comprehensive	Comprehensive	Good
French Polynesia	PYF	PF	Good	Good	Unsupported
Gabon	GAB	GA	Good	Good	Unsupported

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
Gambia	GMB	GM	Good	Good	Unsupported
Georgia	GEO	GE	Good	Good	Unsupported
Germany	DEU	DE	Comprehensive	Comprehensive	Comprehensive
Ghana	GHA	GH	Comprehensive	Comprehensive	Unsupported
Gibraltar	GIB	GI	Comprehensive	Comprehensive	Comprehensive
Greece	GRC	GR	Comprehensive	Comprehensive	Comprehensive
Greenland	GRL	GL	Fair	Fair	Unsupported
Grenada	GRD	GD	Fair	Fair	Unsupported
Guadeloupe	GLP	GP	Comprehensive	Comprehensive	Fair
Guam	GUM	GU	Good	Good	Good
Guatemala	GTM	GT	Comprehensive	Comprehensive	Unsupported
Guernsey	GGY	GG	Comprehensive	Comprehensive	Comprehensive
Guinea	GIN	GN	Good	Good	Unsupported
Guinea-Bissau	GNB	GW	Good	Good	Unsupported
Guyana	GUY	GY	Good	Good	Unsupported
Haiti	HTI	HT	Good	Good	Unsupported
Honduras	HND	HN	Good	Good	Unsupported
Hong Kong	HKG	HK	Comprehensive	Comprehensive	Fair

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
Hungary	GUN	HU	Comprehensive	Comprehensive	Comprehensive
Iceland	ISL	IS	Comprehensive	Comprehensive	Comprehensive
India	IND	IN	Comprehensive	Comprehensive	Good
Indonesia	IDN	ID	Comprehensive	Comprehensive	Good
Iran	IRN	IR	Fair	Fair	Unsupported
Iraq	IRQ	IQ	Good	Good	Unsupported
Ireland	IRL	IE	Comprehensive	Comprehensive	Comprehensive
Isle of Man	IMN	IM	Comprehensive	Comprehensive	Good
Israel	ISR	IL	Comprehensive	Comprehensive	Fair
Italy	ITA	IT	Comprehensive	Comprehensive	Comprehensive
Ivory Coast	CIV	CI	Comprehensive	Comprehensive	Good
Jamaica	JAM	JM	Good	Good	Unsupported
Japan	JPN	JP	Comprehensive	Comprehensive	Comprehensive
Jersey	JEY	JE	Comprehensive	Comprehensive	Good
Jordan	JOR	JO	Comprehensive	Comprehensive	Fair
Kazakhstan	KAZ	KZ	Comprehensive	Comprehensive	Fair
Kenya	KEN	KE	Comprehensive	Comprehensive	Fair
Kiribati	KIR	KI	Fair	Fair	Unsupported

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
Kosovo	KOS	XK	Comprehensive	Comprehensive	Unsupported
Kuwait	KWT	KW	Comprehensive	Comprehensive	Fair
Kyrgyzstan	KGZ	KG	Good	Good	Unsupported
Laos	LAO	LA	Fair	Fair	Unsupported
Latvia	LVA	LV	Comprehensive	Comprehensive	Comprehensive
Lebanon	LBN	LB	Comprehensive	Comprehensive	Fair
Lesotho	LSO	LS	Comprehensive	Comprehensive	Good
Liberia	LBR	LR	Good	Good	Unsupported
Libya	LBY	LY	Good	Good	Unsupported
Liechtenstein	LIE	LI	Comprehensive	Comprehensive	Comprehensive
Lithuania	LTU	LT	Comprehensive	Comprehensive	Comprehensive
Luxembourg	LUX	LU	Comprehensive	Comprehensive	Comprehensive
Macau	MAC	MO	Comprehensive	Comprehensive	Good
Madagascar	MDG	MG	Good	Good	Unsupported
Malawi	MWI	MW	Comprehensive	Comprehensive	Unsupported
Malaysia	MYS	MY	Comprehensive	Comprehensive	Comprehensive
Maldives	MDV	MV	Good	Good	Unsupported

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
Mali	MLI	ML	Good	Good	Unsupported
Malta	MLT	MT	Comprehensive	Comprehensive	Comprehensive
Marshall Islands	MHL	MH	Fair	Fair	Unsupported
Martiniqu e	MTQ	MQ	Comprehensive	Comprehensive	Unsupported
Mauritani a	MRT	MR	Good	Good	Unsupported
Mauritius	MUS	MU	Comprehensive	Comprehensive	Unsupported
Mayotte	MYT	YT	Comprehensive	Comprehensive	Good
Mexico	MEX	MX	Comprehensive	Comprehensive	Fair
Micronesi a, Federated States of	FSM	FM	Fair	Fair	Unsupported
Moldova	MDA	MD	Comprehensive	Comprehensive	Comprehensive
Monaco	MCO	MC	Comprehensive	Comprehensive	Comprehensive
Mongolia	MNG	MN	Good	Good	Fair
Montenegr o	MNE	ME	Comprehensive	Comprehensive	Comprehensive
Montserra t	MSR	MS	Fair	Fair	Unsupported
Morocco	MAR	MA	Comprehensive	Comprehensive	Fair

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
Mozambique	MOZ	MZ	Comprehensive	Comprehensive	Fair
Myanmar	MMR	MM	Comprehensive	Fair	Unsupported
Namibia	NAM	NA	Comprehensive	Comprehensive	Fair
Nauru	NRU	NR	Fair	Fair	Unsupported
Nepal	NPL	NP	Good	Good	Unsupported
Netherlands	NLD	NL	Comprehensive	Comprehensive	Comprehensive
New Caledonia	NCL	NC	Good	Good	Unsupported
New Zealand	NZL	NZ	Comprehensive	Comprehensive	Comprehensive
Nicaragua	NIC	NI	Good	Good	Unsupported
Niger	NER	NE	Good	Good	Unsupported
Nigeria	NGA	NG	Comprehensive	Comprehensive	Unsupported
Niue	NIU	NU	Fair	Fair	Unsupported
Norfolk Island	NFK	NF	Comprehensive	Comprehensive	Unsupported
North Korea	PRK	KP	Fair	Fair	Unsupported
North Macedonia	MKD	MK	Comprehensive	Comprehensive	Good

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
Northern Mariana Islands	MNP	MP	Good	Good	Unsupported
Norway	NOR	NO	Comprehensive	Comprehensive	Comprehensive
Oman	OMN	OM	Comprehensive	Comprehensive	Fair
Pakistan	PAK	PK	Fair	Fair	Unsupported
Palau	PLW	PW	Good	Good	Unsupported
Panama	PAN	PA	Comprehensive	Comprehensive	Fair
Papua New Guinea	PNG	PG	Good	Good	Unsupported
Paraguay	PRY	PY	Comprehensive	Comprehensive	Good
Peru	PER	PE	Comprehensive	Comprehensive	Good
Philippin es	PHL	PH	Comprehensive	Comprehensive	Fair
Pitcairn	PCN	PN	Fair	Fair	Unsupported
Poland	POL	PL	Comprehensive	Comprehensive	Comprehensive
Portugal	PRT	PT	Comprehensive	Comprehensive	Comprehensive
Puerto Rico	PRI	PR	Comprehensive	Comprehensive	Fair
Qatar	QAT	QA	Comprehensive	Comprehensive	Fair
Reunion	REU	RE	Comprehensive	Comprehensive	Unsupported

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
Romania	ROU	RO	Comprehensive	Comprehensive	Good
Russia	RUS	RU	Comprehensive	Comprehensive	Good
Rwanda	RWA	RW	Good	Good	Good
Saint Barthélem y	BLM	BL	Comprehensive	Comprehensive	Unsupported
Saint Helena	SHN	SH	Good	Good	Unsupported
Saint Kitts And Nevis	KNA	KN	Good	Good	Unsupported
Saint Lucia	LCA	LC	Fair	Fair	Unsupported
Saint Martin	MAF	MF	Fair	Fair	Unsupported
Saint Pierre And Miquelon	SPM	PM	Fair	Fair	Unsupported
Saint Vincent And The Grenadine s	VCT	VC	Good	Good	Unsupported
Samoa	WSM	WS	Fair	Fair	Unsupported

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
San Marino	SMR	SM	Comprehensive	Comprehensive	Comprehensive
Sao Tome And Principe	STP	ST	Good	Good	Unsupported
Saudi Arabia	SAU	SA	Comprehensive	Comprehensive	Fair
Senegal	SEN	SN	Comprehensive	Comprehensive	Unsupported
Serbia	SRB	RS	Comprehensive	Comprehensive	Comprehensive
Seychelles	SYC	SC	Comprehensive	Comprehensive	Unsupported
Sierra Leone	SLE	SL	Good	Good	Unsupported
Singapore	SGP	SG	Comprehensive	Comprehensive	Comprehensive
Sint Eustatius, Saba, Bonaire	BES	BQ	Fair	Fair	Unsupported
Sint Maarten	SXM	SX	Fair	Fair	Unsupported
Slovakia	SVK	SK	Comprehensive	Comprehensive	Comprehensive
Slovenia	SVN	SI	Comprehensive	Comprehensive	Comprehensive
Solomon Islands	SLB	SB	Fair	Fair	Unsupported

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
Somalia	SOM	SO	Good	Good	Unsupported
South Africa	ZAF	ZA	Comprehensive	Comprehensive	Comprehensive
South Korea	KOR	KR	Fair	Fair	N/A
South Sudan	SSD	SS	Good	Good	Unsupported
Spain	ESP	ES	Comprehensive	Comprehensive	Comprehensive
Sri Lanka	LKA	LK	Comprehensive	Comprehensive	Unsupported
Sudan	SDN	SD	Good	Good	Unsupported
Suriname	SUR	SR	Good	Good	Unsupported
Svalbard and Jan Mayen Islands	SJM	SJ	Comprehensive	Comprehensive	Unsupported
Sweden	SWE	SE	Comprehensive	Comprehensive	Comprehensive
Switzerla nd	CHE	CH	Comprehensive	Comprehensive	Comprehensive
Syria	SYR	SY	Good	Good	Unsupported
Taiwan	TWN	TW	Comprehensive	Comprehensive	Good
Tajikistan	TJK	TJ	Fair	Fair	Unsupported
Tanzania	TZA	TZ	Comprehensive	Comprehensive	Unsupported
Thailand	THA	TH	Comprehensive	Comprehensive	Fair

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
Timor- Leste	TLS	TL	Good	Good	Unsupported
Togo	TGO	TG	Good	Good	Unsupported
Tokelau	TKL	TK	Fair	Fair	Unsupported
Tonga	TON	TO	Good	Good	Unsupported
Trinidad And Tobago	TTO	TT	Good	Good	Unsupported
Tunisia	TUN	TN	Comprehensive	Comprehensive	Unsupported
Turkey	TUR	TR	Comprehensive	Comprehensive	Good
Turkmenis- tan	TKM	TM	Fair	Fair	Unsupported
Turks And Caicos Islands	TCA	TC	Fair	Fair	Unsupported
Tuvalu	TUV	TV	Fair	Fair	Unsupported
Uganda	UGA	UG	Comprehensive	Comprehensive	Unsupported
Ukraine	UKR	UA	Comprehensive	Comprehensive	Good
United Arab Emirates	ARE	AE	Comprehensive	Comprehensive	Fair
United Kingdom	GBR	GB	Comprehensive	Comprehensive	Comprehensive

Country/ Area	ISO A3	ISO A2	Route coverage	Place Address coverage	Place POI coverage
United States of America	USA	US	Comprehensive	Comprehensive	Comprehensive
Uruguay	URY	UY	Comprehensive	Comprehensive	Comprehensive
US Virgin Islands	VIR	VI	Comprehensive	Comprehensive	Fair
Uzbekistan	UZB	UZ	Good	Good	Unsupported
Vanuatu	VUT	VU	Good	Good	Unsupported
Vatican City	VAT	VA	Comprehensive	Comprehensive	Comprehensive
Venezuela	VEN	VE	Comprehensive	Comprehensive	Fair
Vietnam	VNM	VN	Comprehensive	Comprehensive	Comprehensive
Wallis And Futuna Islands	WLF	WF	Good	Good	Unsupported
Western Sahara	ESH	EH	Comprehensive	Comprehensive	Unsupported
Yemen	YEM	YE	Good	Good	Unsupported
Zambia	ZMB	ZM	Comprehensive	Comprehensive	Unsupported
Zimbabwe	ZWE	ZW	Comprehensive	Comprehensive	Unsupported

Get started with Amazon Location Service

This topic helps you get started with Amazon Location Service. Follow these steps to create your first application and understand how to choose the appropriate Amazon Location API based on common use cases.

1. Set up your AWS account and access.

If you don't already have an account, you'll be prompted to create one. With the AWS free tier, you can get three months of free tier for Amazon Location Service.

If you already have an account, you need to provide access to Amazon Location Service.

Continue with the [the section called "Set up an account for the first time"](#) topic.

2. Understand and set up authentication.

To use Amazon Location, a user must be granted access to the resources and APIs that make up Amazon Location. API Key, Amazon Cognito, and AWS Identity and Access Management (IAM) are three options to grant access to your resources and actions (APIs).

Continue with the [Authentication](#) topic.

3. Create your first location application.

See [the section called "Create your first Maps and Places application"](#) to build your first "Hello World" application.

4. Choose the right API.

Amazon Location offers a rich set of APIs across Places, Routes, Maps, and Geofences and Trackers to solve a variety of business use cases.

To learn more about how to choose an Amazon Location API, see [the section called "Choose an API"](#) for more information.

Set up your account

This section describes what you need to do to use Amazon Location Service. You must have an AWS account and have set up access to Amazon Location for users that want to use it.

I'm new to AWS

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open <https://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform [tasks that require root user access](#).

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to <https://aws.amazon.com/> and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the [AWS Management Console](#) as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

For help signing in by using root user, see [Signing in as the root user](#) in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

For instructions, see [Enable a virtual MFA device for your AWS account root user \(console\)](#) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

For instructions, see [Enabling AWS IAM Identity Center](#) in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

For a tutorial about using the IAM Identity Center directory as your identity source, see [Configure user access with the default IAM Identity Center directory](#) in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

For help signing in using an IAM Identity Center user, see [Signing in to the AWS access portal](#) in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

For instructions, see [Create a permission set](#) in the *AWS IAM Identity Center User Guide*.

2. Assign users to a group, and then assign single sign-on access to the group.

For instructions, see [Add groups](#) in the *AWS IAM Identity Center User Guide*.

I already have an AWS account

Grant access to Amazon Location Service

Your non-admin users have no permissions by default. Before they can access Amazon Location, you must grant permission by attaching an IAM policy with specific permissions. Make sure to follow the principle of least privilege when granting access to resources.

Note

For information about giving unauthenticated users access to Amazon Location Service functionality (for example, in a web-based application), see [Authentication](#).

The following example policy gives a user permission to access all Amazon Location operations. For more examples, see [the section called "Identity-based policy examples"](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "geo:*"
        "geo-maps:*"
        "geo-places:*"
        "geo-routes:*"
      ],
      "Resource": "*",
      "Effect": "Allow"
    }
  ]
}
```

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

Create a permission set. Follow the instructions in [Create a permission set](#) in the *AWS IAM Identity Center User Guide*.

- Users managed in IAM through an identity provider:

Create a role for identity federation. Follow the instructions in [Create a role for a third-party identity provider \(federation\)](#) in the *IAM User Guide*.

- IAM users:

- Create a role that your user can assume. Follow the instructions in [Create a role for an IAM user](#) in the *IAM User Guide*.

- (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in [Adding permissions to a user \(console\)](#) in the *IAM User Guide*.

When creating applications that use Amazon Location Service, you may need some users to have unauthenticated access. For these use cases, see [Enabling unauthenticated access using Amazon Cognito](#).

Use the Amazon Location Service console to authenticate

Note

To learn more about authentication, see [Authentication](#).

To use Amazon Location Service, a user must be granted access to the resources and APIs that make up Amazon Location. By default, the Amazon Location APIs require authentication to use. You can use either Amazon Cognito or API keys to provide authentication and authorization for anonymous users.

In the [the section called “Create your first Maps and Places application”](#) tutorial, the application has anonymous usage, which means your users aren't required to sign in. In the tutorial, you create API keys for use in the sample application.

Follow the procedures below to create your first API key.

1. In the [Amazon Location console](#) and choose **API keys** from the left menu.
2. On the **API keys** page, choose **Create API key**.
3. On the **Create API key** page, fill in the following information:
 - **Name** – A name for your API key, such as MyHelloWorldApp.
 - **Description** – An optional description for your API key.
 - **Actions** – Specify the actions you want to authorize with this API key. You must select at least **geo-maps:Get*** and **geo-places:Search***.
 - **Expiration time** – Optionally, add an expiration date and time for your API key. For more information, see [the section called “Best practices”](#).

- **Referers** – Optionally, add one or more domains where you can use the API key. For example, if the API key is to allow an application running on the website `example.com`, then you could put `*.example.com/` as an allowed referrer.
- **Tags** – Optionally, add tags to the API key.

Important

We recommend that you protect your API key usage by setting either an expiration time or a referrer, if not both.

4. Choose **Create API key** to create the API key.
5. On the detail page for the API key, you can see information about the API key that you have created.

Choose **Show API key** and copy the key value to use later in the [the section called “Create your first Maps and Places application”](#) tutorial. The key value will have the format `v1.public.a1b2c3d4....`

Choose the right API

This topic helps you choose an Amazon Location Service API based on common use cases that you may want to solve with location-based data and services.

Maps

Maps provide access to both dynamic and static map types for a variety of applications. For more information, See [Maps](#).

- **Dynamic Maps:** Interactive maps that can be customized in real time, allowing users to pan, zoom, and overlay data. For more information, See [the section called “Dynamic maps”](#).
- **Static Maps:** Static images of maps that display specific locations or routes without interactive elements, suitable for applications with limited interactivity. For more information, See [the section called “Static maps”](#).

Routes

Routes provide capabilities for calculating optimized paths between locations. These features support applications requiring logistics planning, distance calculations, and route optimization.

Users can also snap location points to roads for improved accuracy. For more information, See [Routes](#).

- **CalculateIsolines:** Generates isolines based on travel time or distance, useful for defining service areas or reachability zones. For more information, See [the section called "Calculate isolines"](#).
- **CalculateRouteMatrix:** Provides a matrix of distances and travel times between multiple origins and destinations, supporting logistics and trip planning. For more information, See [the section called "Calculate route matrix"](#).
- **CalculateRoutes:** Calculates optimized routes for point-to-point or multi-stop navigation, including customizable routing preferences. For more information, See [the section called "Calculate routes"](#).
- **OptimizeWaypoints:** Optimizes the order of waypoints for the most efficient travel route, minimizing distance or time. For more information, See [the section called "Optimize waypoints"](#).
- **SnapToRoads:** Aligns coordinates to the nearest road paths, enhancing GPS accuracy by snapping points to known roads. For more information, See [the section called "Snap to Roads"](#).

Places

Places enable applications to search, find, and retrieve details about points of interest, addresses, and specific locations. These capabilities enhance location-based services by providing context and improving user experience in search functions. For more information, See [Places](#).

- **Geocode:** Converts addresses or place names into geographic coordinates (longitude, latitude), supporting applications that require address-to-location transformation for mapping and spatial analysis. For more information, see [the section called "Reverse Geocode"](#).
- **Reverse Geocode:** Converts geographic coordinates to the nearest address or place name, providing context for a location. For more information, See [the section called "Reverse Geocode"](#).
- **Autocomplete:** Suggests potential completions for user-entered text, improving efficiency in search input. For more information, See [the section called "Autocomplete"](#).
- **GetPlace:** Retrieves detailed information about a specified place, including attributes like address, contact details, and opening hours. For more information, See [the section called "GetPlace"](#).

- **SearchNearby:** Finds places within a specified radius of a given geographic point, suitable for "near me" searches. For more information, See [the section called "Search Nearby"](#).
- **SearchText:** Allows text-based searching for places or points of interest based on a keyword or phrase, ideal for finding locations by name or description. For more information, See [the section called "Search Text"](#).
- **Suggest:** Provides search term suggestions as users type, enhancing search relevance and user experience. For more information, See [the section called "Suggest"](#).

Geofences

Geofencing allows applications to define geographical boundaries and monitor entry or exit events within these regions. Features include creating, updating, and deleting geofences, as well as configuring notifications or triggers for location-based actions when tracked devices cross geofence boundaries. Ideal for proximity-based notifications, security monitoring, and asset tracking within predefined areas. For more information, See [Geofences](#).

Trackers

Tracking enables real-time monitoring of device or asset locations over time. Features include adding tracked devices, updating their location data, and retrieving historical position data. Trackers are useful for managing fleets, monitoring personnel, and ensuring the security of valuable assets by providing up-to-date location data and movement patterns. For more information, See [Trackers](#).

Create your first Amazon Location Maps and Places application

In this section, you will create your first application with Maps and Places.

Prerequisite:

If you already created an API key in the [the section called "Set up authentication for the first time"](#) steps, let's get started.

If you haven't created an API key yet, follow [the section called "Set up authentication for the first time"](#) before continuing to build the application. If you have any questions, see [the section called "Use API keys"](#) and [the section called "Supported regions"](#) for more information.

Web

Here's a step-by-step tutorial for creating an Amazon Location Service map application with MapLibre GL JS. This guide will walk you through setting up the map, adding styling options, and enabling place search functionality.

Set up the initial page

In this section, we will set up the initial page and folder structure.

Add required libraries and stylesheets

Create an `index.html` file. To render the map, you need MapLibre GL JS and MapLibre GL Geocoder. You will add the MapLibre and Geocoder stylesheets and JavaScript scripts.

Copy and paste the following code into your `index.html` file.

```
<!DOCTYPE html>
<html lang="en">
<head>

  <title>Amazon Location Service - Getting Started with First Map App</title>
  <meta charset='utf-8'>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="description" content="Interactive map application using Amazon Location
Service">

  <!--Link to MapLibre CSS and JavaScript library for map rendering and visualization
-->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/maplibre-gl@4.7.1/dist/
maplibre-gl.css" />
  <script src="https://cdn.jsdelivr.net/npm/maplibre-gl@4.7.1/dist/maplibre-gl.js"></
script>

  <!--Link to MapLibre Geocoder CSS and JavaScript library for place search and
geocoding -->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@maplibre/maplibre-gl-
geocoder@1.7.0/dist/maplibre-gl-geocoder.css" />
  <script src="https://cdn.jsdelivr.net/npm/@maplibre/maplibre-gl-geocoder@1.7.0/
dist/maplibre-gl-geocoder.js"></script>

  <!--Link to amazon-location JavaScript librarie -->
  <script src="https://cdn.jsdelivr.net/npm/@aws/amazon-location-utilities-auth-
helper@1"></script>
```

```
<script src="https://cdn.jsdelivr.net/npm/@aws/amazon-location-client@1.2"></script>

<!-- Link to the first Amazon Location Map App's CSS and JavaScript -->
<script src="utils.js"></script>
<link rel="stylesheet" href="style.css"/>

</head>
<body>
  <main>

  </main>
  <script>
    // Step 1: Setup API Key and AWS Region
    // Step 2.1 Add maps to application
    // Step 2.2 initialize the map
    // Step 3: Add places features to application
    // Step 3.1: Get GeoPlaces instance. It will be used for addion search box and
map click functionality
    // Step 3.2: Add search box to the map
    // Step 3.3.: Setup map click functionality
    // Add functions
  </script>
</body>
</html>
```

Create the Map container

Under the `<body>` element of the HTML file, create a `<div>` element in your HTML to hold the map. You can style this `<div>` in your CSS to set dimensions as needed for your application. You must download the CSS file, `style.css`, from our GitHub repository. This will help you focus on business logic.

Save the `style.css` and `index.html` files in the same folder.

Download the `style.css` file from [GitHub](#).

```
<main role="main" aria-label="Map Container">
  <div id="map"></div>
</main>
```

Add API key and AWS Region details

Add the API key you created in [the section called "Use API keys"](#) to this file, along with the AWS Region where the key was created.

```
<!DOCTYPE html>
<html lang="en">
  .....
  .....
  <body>
    <main role="main" aria-label="Map Container">
      <div id="map"></div>
    </main>
    <script>
      // Step 1: Setup API Key and AWS Region
      const API_KEY = "Your_API_Key";
      const AWS_REGION = "Region_where_you_created_API_Key";
      // Step 2: Add maps to application
      // Step 2.1 initialize the map
      // Step 2.2 Add navigation controls to the map
      // Step 3: Add places feature to application
      // Step 3.1: Get GeoPlaces instance. It will be used for addion search box
and map click functionality
      // Step 3.2: Add search box to the map
      // Step 3.3.: Setup map click functionality
    </script>
  </body>
</html>
```

Add Map to your application

In this section, we will add Map capabilities to the application. Before you start, your files should be in this folder structure.

If have not already done so, please download the `style.css` file from [GitHub](#).

```
|---FirstApp [Folder]
|----- index.html [File]
|----- style.css [File]
```

Create a Function to Initialize the Map

To set up your map, create the following function, `initializeMap(...)`, after the line `//Add functions`.

Choose an initial center location and zoom level. In this example, we set the map center to Vancouver, Canada, with a zoom level of 10. Add navigation controls for easy zooming.

```
/**
 * Initializes the map with the specified style and color scheme.
 */
function initializeMap(mapStyle = "Standard", colorScheme = "Dark") {
    const styleUrl = `https://maps.geo.${AWS_REGION}.amazonaws.com/v2/styles/
${mapStyle}/descriptor?key=${API_KEY}&color-scheme=${colorScheme}`;
    const map = new maplibregl.Map({
        container: 'map',           // The ID of the map container
        style: styleUrl,           // The style URL for the map
        center: [-123.116226, 49.246292], // Starting center coordinates
        zoom: 10,                 // Initial zoom level
        validateStyle: false      // Disable style validation
    });
    return map;                  // Return the initialized map
}
```

Initialize the Map

Call `initializeMap(...)` to initialize the map. Optionally, you can initialize it with your preferred style and color scheme after the `initializeMap` function. For more style options, see [the section called “Map styles”](#).

```
// Step 1: Setup API Key and AWS Region
const API_KEY = "Your_API_Key";
const AWS_REGION = "Region_where_you_created_API_Key";

// Step 2.1 Add maps to application
// Step 2.2 initialize the map
const map = initializeMap("Standard","Light");

// Step 3: Add places features to application
```

Open `index.html` in a browser to see the map in action.

Add Navigation Control

Optionally, you can add navigation controls (zoom and rotation) to the map. This should be done after calling `initializeMap(...)`.

```
// Step 2.1 initialize the map
const map = initializeMap("Standard","Light");

// Step 2.2 Add navigation controls to the map
map.addControl(new maplibregl.NavigationControl());

// Step 3: Add places features to application
```

Review the Map Code

Congratulations! Your first app is ready to use a map. Open `index.html` in a browser. Make sure `style.css` is in the same folder as `index.html`.

Your final HTML should look like this:

```
<!DOCTYPE html>
<html lang="en">
<head>

  <title>Amazon Location Service - Getting Started with First Map App</title>
  <meta charset='utf-8'>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="description" content="Interactive map application using Amazon
Location Service">

  <!-- Link to MapLibre CSS and JavaScript library for map rendering and
visualization -->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/maplibre-gl@4.7.1/
dist/maplibre-gl.css" />
  <script src="https://cdn.jsdelivr.net/npm/maplibre-gl@4.7.1/dist/maplibre-
gl.js"></script>
```

```

        <!-- Link to MapLibre Geocoder CSS and JavaScript library for place search
and geocoding -->
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@maplibre/
maplibre-gl-geocoder@1.7.0/dist/maplibre-gl-geocoder.css" />
        <script src="https://cdn.jsdelivr.net/npm/@maplibre/maplibre-gl-
geocoder@1.7.0/dist/maplibre-gl-geocoder.js"></script>

        <!-- Link to amazon-location JavaScript library -->
        <script src="https://cdn.jsdelivr.net/npm/@aws/amazon-location-utilities-
auth-helper@1"></script>
        <script src="https://cdn.jsdelivr.net/npm/@aws/amazon-location-
client@1.2"></script>

        <!-- Link to the first Amazon Location Map App's CSS and JavaScript -->
        <script src="utils.js"></script>
        <link rel="stylesheet" href="style.css"/>
</head>

<body>
    <main role="main" aria-label="Map Container">
        <div id="map"></div>
    </main>
    <script>
        const API_KEY = "Your_API_Key";
        const AWS_REGION = "Region_where_you_created_API_Key";

        function initializeMap(mapStyle, colorScheme) {
            const styleUrl = `https://maps.geo.${AWS_REGION}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?key=${API_KEY}&color-scheme=${colorScheme}`;

            const map = new maplibregl.Map({
                container: 'map',                // ID of the HTML element for
the map
                style: styleUrl,                // URL for the map style
                center: [-123.116226, 49.246292], // Initial map center
[longitude, latitude]
                zoom: 10                        // Initial zoom level
            });
            map.addControl(new maplibregl.NavigationControl());
            return map;
        }

        const map = initializeMap("Standard", "Light");

```

```
    </script>
  </body>
</html>
```

Add Places to your application

In this section, we will set up add places capabilities to the application. Download the JavaScript file from GitHub, [utils.js](#).

Before you start, your files should be in this folder structure:

```
|---FirstApp [Folder]
|----- index.html [File]
|----- style.css [File]
|----- utils.js [File]
```

Create Function to Create GeoPlaces

To add search functionality, initialize the GeoPlaces class using AuthHelper and AmazonLocationClient. Add the following getGeoPlaces(map) function before the </script> tag in index.html.

```
/**
 * Gets a GeoPlaces instance for Places operations.
 */
function getGeoPlaces(map) {
  const authHelper = amazonLocationClient.withAPIKey(API_KEY, AWS_REGION);
  // Authenticate using the API key and AWS region
  const locationClient = new
amazonLocationClient.GeoPlacesClient(authHelper.getClientConfig()); // Create a
GeoPlaces client
  const geoPlaces = new GeoPlaces(locationClient, map);
  // Create GeoPlaces instance
  return geoPlaces;
  // Return the GeoPlaces instance
}
```


Create Function to Add Search Box to the Application

Add the following `addSearchBox(map, geoPlaces)`, `renderPopup(feature)`, and `createPopup(feature)` functions before the `</script>` tag in `index.html` to complete the search functionality setup.

```
/**
 * Adds search box to the map.
 */
function addSearchBox(map, geoPlaces) {
  const searchBox = new MaplibreGeocoder(geoPlaces, {
    maplibregl,
    showResultsWhileTyping: true,           // Show results while
typing
    debounceSearch: 300,                   // Debounce search
requests
    limit: 30,                             // Limit number of
results
    popuprender: renderPopup,             // Function to render
popup
    reverseGeocode: true,                 // Enable reverse
geocoding
    zoom: 14,                             // Zoom level on
result selection
    placeholder: "Search text or nearby (lat,long)" // Placeholder text
for search box.
  });

  // Add the search box to the map
  map.addControl(searchBox, 'top-left');

  // Event listener for when a search result is selected
  searchBox.on('result', async (event) => {
    const { id, result_type } = event.result; // Get
result ID and type
    if (result_type === "Place") { // Check
if the result is a place
      const placeResults = await geoPlaces.searchByPlaceId(id); // Fetch
details for the selected place
      if (placeResults.features.length) {
        createPopup(placeResults.features[0]).addTo(map); // Create
and add popup for the place
      }
    }
  });
}
```

```

        }
    }
    });
}

/**
 * Renders the popup content for a given feature.
 */
function renderPopup(feature) {
    return `
        <div class="popup-content">
            <span class="${feature.place_type.toLowerCase()} badge">
                ${feature.place_type}</span><br>
                ${feature.place_name}
            </div>`;
}

/**
 * Creates a popup for a given feature and sets its position.
 */
function createPopup(feature) {
    return new maplibregl.Popup({ offset: 30 }) // Create a new popup
        .setLngLat(feature.geometry.coordinates) // Set the popup position
        .setHTML(renderPopup(feature)); // Set the popup content
}

```

Add Search Box to the Application

Create a GeoPlaces object by calling `getGeoPlaces(map)` as defined in Section 3.1 and then call `addSearchBox(map, geoPlaces)` to add the search box to the application.

```

// Step 2: Add maps to application
// Step 2.1 initialize the map
const map = initializeMap("Standard","Light");
// Step 2.2 Add navigation controls to the map
map.addControl(new maplibregl.NavigationControl());

// Step 3: Add places feature to application
// Step 3.1: Get GeoPlaces instance. It will be used for adding search box and
map click functionality
const geoPlaces = getGeoPlaces(map);

```

```
// Step 3.2: Add search box to the map
addSearchBox(map, geoPlaces);
```

Your place search is ready to use. Open `index.html` in a browser to see it in action.

Add Function to Show Popup on User Click on the Map

Create a function `addMapClick(map, geoPlaces)` to display a popup when the user clicks on the map. Add this function just before the `</script>` tag.

```
/**
 * Sets up reverse geocoding on map click events.
 */
function addMapClick(map, geoPlaces) {
  map.on('click', async ({ lngLat }) => { // Listen for
click events on the map
    const response = await geoPlaces.reverseGeocode({ query: [lngLat.lng,
lngLat.lat], limit: 1, click: true }); // Perform reverse geocoding

    if (response.features.length) { // If there are
results
      const clickMarker = new maplibregl.Marker({ color: "orange" }); //
Create a marker
      const feature = response.features[0]; // Get the
clicked feature
      const clickedPopup = createPopup(feature); // Create popup
for the clicked feature
      clickMarker.setLngLat(feature.geometry.coordinates) // Set marker
position
        .setPopup(clickedPopup) // Attach popup
to marker
        .addTo(map); // Add marker
to the map

      clickedPopup.on('close', () => clickMarker.remove()).addTo(map); //
Remove marker when popup is closed
    }
  });
}
```

Call Function to Add Map Click Feature

To enable the map click action, call `addMapClick(map, geoPlaces)` after the line containing `addSearchBox(map, geoPlaces)`.

```
// Step 3: Add places feature to application
// Step 3.1: Get GeoPlaces instance. It will be used for adding search box and
map click functionality
const geoPlaces = getGeoPlaces(map);
// Step 3.2: Add search box to the map
addSearchBox(map, geoPlaces);
// Step 3.3: Setup map click functionality
addMapClick(map, geoPlaces);
```

Review Maps and Places application

Congratulations! Your first application is ready to use Maps and Places. Open `index.html` in a browser. Make sure `style.css` and `utils.js` are in the same folder with `index.html`.

Your final HTML should look like this:

```
<!DOCTYPE html>
<html lang="en">
<head>

  <title>Amazon Location Service - Getting Started with First Map App</title>
  <meta charset='utf-8'>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="description" content="Interactive map application using Amazon Location
Service">

  <!--Link to MapLibre CSS and JavaScript library for map rendering and visualization
-->
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/maplibre-gl@4.7.1/dist/
maplibre-gl.css" />
  <script src="https://cdn.jsdelivr.net/npm/maplibre-gl@4.7.1/dist/maplibre-gl.js"></
script>

  <!--Link to MapLibre Geocoder CSS and JavaScript library for place search and
geocoding -->
```

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@maplibre/maplibre-gl-geocoder@1.7.0/dist/maplibre-gl-geocoder.css" />
<script src="https://cdn.jsdelivr.net/npm/@maplibre/maplibre-gl-geocoder@1.7.0/dist/maplibre-gl-geocoder.js"></script>

<!--Link to amazon-location JavaScript librarie -->
<script src="https://cdn.jsdelivr.net/npm/@aws/amazon-location-utilities-auth-helper@1"></script>
<script src="https://cdn.jsdelivr.net/npm/@aws/amazon-location-client@1.2"></script>

<!-- Link to the first Amazon Location Map App's CSS and JavaScript -->
<script src="utils.js"></script>
<link rel="stylesheet" href="style.css"/>

</head>
<body>
  <main role="main" aria-label="Map Container">
    <div id="map"></div>
  </main>
  <script>
    // Step 1: Setup API Key and AWS Region
    const API_KEY = "Your_API_Key";
    const AWS_REGION = "Region_where_you_created_API_Key";

    // Step 2: Add maps to application
    // Step 2.1 initialize the map
    const map = initializeMap("Standard","Light");
    // Step 2.2 Add navigation controls to the map
    map.addControl(new maplibregl.NavigationControl());

    // Step 3: Add places feature to application
    // Step 3.1: Get GeoPlaces instance. It will be used for addion search box and
    map click functionality
    const geoPlaces = getGeoPlaces(map);
    // Step 3.2: Add search box to the map
    addSearchBox(map, geoPlaces);
    // Step 3.3.: Setup map click functionality
    addMapClick(map, geoPlaces);
```

```

/**
 * Functions to add maps and places feature.
 */

/**
 * Initializes the map with the specified style and color scheme.
 */
function initializeMap(mapStyle = "Standard", colorScheme = "Dark") {
    const styleUrl = `https://maps.geo.${AWS_REGION}.amazonaws.com/v2/styles/
    ${mapStyle}/descriptor?key=${API_KEY}&color-scheme=${colorScheme}`;
    const map = new maplibregl.Map({
        container: 'map', // The ID of the map container
        style: styleUrl, // The style URL for the map
        center: [-123.116226, 49.246292], // Starting center coordinates
        zoom: 10, // Initial zoom level
        validateStyle: false // Disable style validation
    });
    return map; // Return the initialized map
}

/**
 * Gets a GeoPlaces instance for Places operations.
 */
function getGeoPlaces(map) {
    const authHelper = amazonLocationClient.withAPIKey(API_KEY, AWS_REGION);
    // Authenticate using the API key and AWS region
    const locationClient = new
amazonLocationClient.GeoPlacesClient(authHelper.getClientConfig()); // Create a
GeoPlaces client
    const geoPlaces = new GeoPlaces(locationClient, map);
    // Create GeoPlaces instance
    return geoPlaces;
    // Return the GeoPlaces instance
}

/**
 * Adds search box to the map.
 */

function addSearchBox(map, geoPlaces) {
    const searchBox = new MaplibreGeocoder(geoPlaces, {
        maplibregl,
        showResultsWhileTyping: true, // Show results while
typing

```

```

        debounceSearch: 300, // Debounce search
requests
        limit: 30, // Limit number of
results
        popuprender: renderPopup, // Function to render
popup
        reverseGeocode: true, // Enable reverse
geocoding
        zoom: 14, // Zoom level on
result selection
        placeholder: "Search text or nearby (lat,long)" // Place holder text
for search box.
    });

    // Add the search box to the map
    map.addControl(searchBox, 'top-left');

    // Event listener for when a search result is selected
    searchBox.on('result', async (event) => {
        const { id, result_type } = event.result; // Get
result ID and type
        if (result_type === "Place") { // Check
if the result is a place
            const placeResults = await geoPlaces.searchByPlaceId(id); // Fetch
details for the selected place
            if (placeResults.features.length) {
                createPopup(placeResults.features[0]).addTo(map); // Create
and add popup for the place
            }
        }
    });
}

/**
 * Renders the popup content for a given feature.
 */
function renderPopup(feature) {
    return `
        <div class="popup-content">
            <span class="${feature.place_type.toLowerCase()} badge">
${feature.place_type}</span><br>
                ${feature.place_name}
            </div>`;
}

```

```
/**
 * Creates a popup for a given feature and sets its position.
 */
function createPopup(feature) {
    return new maplibregl.Popup({ offset: 30 }) // Create a new popup
        .setLngLat(feature.geometry.coordinates) // Set the popup position
        .setHTML(renderPopup(feature)); // Set the popup content
}

/**
 * Sets up reverse geocoding on map click events.
 */
function addMapClick(map, geoPlaces) {
    map.on('click', async ({ lngLat }) => { // Listen for
click events on the map
        const response = await geoPlaces.reverseGeocode({ query: [lngLat.lng,
lngLat.lat], limit: 1, click:true }); // Perform reverse geocoding

        if (response.features.length) { // If there are
results
            const clickMarker = new maplibregl.Marker({ color: "orange" }); //
Create a marker
            const feature = response.features[0]; // Get the
clicked feature
            const clickedPopup = createPopup(feature); // Create popup
for the clicked feature
            clickMarker.setLngLat(feature.geometry.coordinates) // Set marker
position
                .setPopup(clickedPopup) // Attach popup
to marker
                .addTo(map); // Add marker
to the map

            clickedPopup.on('close', () => clickMarker.remove()).addTo(map); //
Remove marker when popup is closed
        }
    });
}

</script>
</body>
</html>
```


Explore more

You have completed the quick start tutorial, and should have an idea of how Amazon Location Service is used to build applications. To get more out of Amazon Location, you can check out the following resources:

- **Query suggestion details** - Consider extending the `GeoPlaces` class or using a similar approach to `ReverseGeocode` to get more details about results returned by the Suggestion API.
- **Choose the right API for your business needs** - To determine the best Amazon Location API for your requirements, check out this resource: [the section called "Choose an API"](#).
- **Check out Amazon Location "how-to" guides** - Visit the [Amazon Location Service Developer Guide](#) for tutorials and further resources.
- **Documentation and product information** - For complete documentation, visit the [Amazon Location Service Developer Guide](#) . To learn more about the product, go to the [Amazon Location Service Product](#) page.

Create your first Geofences and Trackers application

In this section, you'll create an application that demonstrates the key features of using the Amazon Location Geofences and Trackers. The applications demonstrate how a tracker and geofence interact using a combination of Lambda, AWS IoT, and Amazon Location features. Choose iOS or Android platform to get started.

Before you start to build your application, follow the procedures in [the section called "Set up authentication for the first time"](#) to grant appropriate access.

Continue with the procedures for your preferred operating system:

- [Create an iOS application](#)
- [Create an Android application](#)

Create an iOS application

Follow these procedures to build an iOS application using Amazon Location Service.

Clone the project files from [GitHub](#).

Create Amazon Location resources for your app

You can generate Amazon Location Service resources once your AWS account is ready. These resources will be essential for executing the provided code snippets.

Note

If you haven't created an AWS account yet, please [create an AWS account](#).

To begin you will need to create a Amazon Cognito Identity Pool Id, use the following procedure:

1. In the AWS console, navigate to the Amazon Cognito service and then select **Identity pools** from the left side menu and select **Create Identity pool**.
2. Make sure **Guest Access** is checked, and press **Next** to continue.
3. Next create a new IAM role or Use an existing IAM role.
4. Enter an Identity pool name, and make sure Identity Pool has access to Amazon Location (geo) resources for the map and tracker you will be creating in the next procedure.
- 5.

Now you need to create and style a map in the AWS Amazon Location console, use the following procedure:

1. Navigate to the [Maps section](#) in the Amazon Location console and select **Create Map** to preview available map styles.
2. Give the new map resource a **Name** and **Description**. Record the name you assign to the map resource, as it is used later in the tutorial.
3. When choosing a map style, consider the map data provider. Refer to section 82 of the [AWS service terms](#) for more details.
4. Accept the [Amazon Location Terms and Conditions](#), then select **Create Map**. After map has been created, you can interact with the map by zooming in, out, or panning in any direction.

To create a tracker using the Amazon Location console

1. Open the [Amazon Location Service console](#).
2. In the left navigation pane, choose **Trackers**.

3. Choose **Create tracker**.
4. Fill in the all the required fields.
5. Under **Position filtering**, choose the option that best fits how you intend to use your tracker resource. If you do not set Position filtering, the default setting is TimeBased. For more information, see Trackers in this guide, and PositionFiltering in the Amazon Location Service Trackers API Reference.
6. Choose **Create tracker** to finish.

Create a Geofence Collection

When creating a geofence collection you can use either the console, API or CLI. The following procedures walk you through each option.

Create a geofence collection using the Amazon Location console:

1. Open the Amazon Location Service console at <https://console.aws.amazon.com/location/>.
2. In the left navigation pane, choose Geofence Collections.
3. Choose Create geofence collection.
4. Provide a name and description for the collection.
5. Under EventBridge rule with CloudWatch as a target, you can create an optional EventBridge rule to get started reacting to geofence events. This enables Amazon Location to publish events to Amazon CloudWatch Logs.
6. Choose Create geofence collection.

Create a geofence collection using the Amazon Location APIs:

Use the CreateGeofenceCollection operation from the Amazon Location Geofences APIs. The following example uses an API request to create a geofence collection called *GOECOLLECTION_NAME*.

```
POST /geofencing/v0/collections
Content-type: application/json
{
  "CollectionName": "GOECOLLECTION_NAME",
  "Description": "Geofence collection 1 for shopping center",
  "Tags": {
    "Tag1" : "Value1"
```

```
}  
}
```

Create a geofence collection using AWS CLI commands:

Use the `create-geofence-collection` command. The following example uses an AWS CLI to create a geofence collection called *GOECOLLECTION_NAME*.

```
aws location \ create-geofence-collection \  
  --collection-name "GOECOLLECTION_NAME" \  
  --description "Shopping center geofence collection" \  
  --tags Tag1=Value1
```

Link a tracker to a geofence collection

To link a tracker to a geofence collection you can use either the console, API, or CLI. The following procedures walk you through each option.

Link a tracker resource to a geofence collection using the Amazon Location Service console:

1. Open the Amazon Location console.
2. In the left navigation pane, choose **Trackers**.
3. Under **Device Trackers**, select the name link of the target tracker.
4. Under **Linked Geofence Collections**, choose **Link Geofence Collection**.
5. In the **Linked Geofence Collection window**, select a geofence collection from the dropdown menu.
6. Choose **Link**.
7. After you link the tracker resource, it will be assigned an Active status.

Link a tracker resource to a geofence collection using the Amazon Location APIs:

Use the `AssociateTrackerConsumer` operation from the Amazon Location Trackers APIs. The following example uses an API request that associates `ExampleTracker` with a geofence collection using its Amazon Resource Name (ARN).

```
POST /tracking/v0/trackers/ExampleTracker/consumers  
Content-type: application/json  
{
```

```
    "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GOECOLLECTION_NAME"
  }
```

Link a tracker resource to a geofence collection using AWS CLI commands:

Use the `associate-tracker-consumer` command. The following example uses an AWS CLI to create a geofence collection called *GOECOLLECTION_NAME*.

```
aws location \
associate-tracker-consumer \
  --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GOECOLLECTION_NAME" \
  --tracker-name "ExampleTracker"
```

Use AWS Lambda with MQTT

Create a Lambda function:

To create a connection between AWS IoT Core and Amazon Location Service, you need an AWS Lambda function to process messages forwarded by EventBridge CloudWatch events. This function will extract any positional data, format it for Amazon Location Service, and submit it through the Amazon Location Tracker API. You can create this function through the AWS Lambda console, or you can use the AWS Command Line Interface (AWS CLI) or the AWS Lambda APIs. To create a Lambda function that publishes position updates to Amazon Location using the console:

1. Open the AWS Lambda console at <https://console.aws.amazon.com/lambda/>.
2. From the left navigation, choose Functions.
3. Choose Create Function, and make sure that Author from scratch is selected.
4. Fill out the following boxes:
 - a Function name
 - for the **Runtime** option, choose Node.js 16.x.
5. Choose Create function.
6. Choose the Code tab to open the editor.
7. Overwrite the placeholder code in `index.js` with the following:

```
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. Choose Deploy to save the updated function.
9. Choose the Configuration tab.
10. In the Triggers section, click on Add trigger.
11. Select EventBridge (CloudWatch Events) in Source field.
12. Select `Existing Rules` radio option.
13. Enter the rule name like this `AmazonLocationMonitor-GEOFENCECOLLECTION_NAME`.
14. Click on the Add button.
15. This will also attach `Resource-based policy statements` in permissions tab

MQTT Test Client

1. Open the <https://console.aws.amazon.com/iot/>.
2. In the left navigation pane, choose MQTT test client.
3. You'll see a section titled **MQTT test client** where you can configure your MQTT connection.
4. After configuring the necessary settings, click on the **Connect** button to establish a connection to the MQTT broker using the provided parameters.
5. Make note of the Endpoint value.

Once connected, you can subscribe to MQTT topics or publish messages to topics using the respective input fields provided in the MQTT test client interface. Next you will attach the MQTT Policy:

1. On the left side menu, under **Manage** expand **Security** option and click on **Policies**.
2. Click on **Create Policy** button.
3. Enter a policy name.
4. On **Policy Document** select **JSON** tab.
5. Copy paste the policy shown below, but make sure to update all elements with your *REGION* and *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"
    }
  ]
}
```

6. Select the **Create** button to finish.

Set up sample app code

In order to setup the sample code you must have the following tools installed:

- Git
- XCode 15.3 or Later
- iOS Simulator 16 or later

Use this procedure to set up the sample app code:

1. Clone the git repository from this URL: <https://github.com/aws-geospatial/amazon-location-samples-ios/tree/main/tracking-with-geofence-notifications>.
2. Open the `AWSLocationSampleApp.xcodeproj` project file.

3. Wait for the package resolution process.
4. **Optional:** On the project navigation menu rename `ConfigTemplate.xcconfig` to `Config.xcconfig` and fill in the following values:

```
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`
```

Use the sample app

After setting up the sample code you can now run the app on an iOS simulator or a physical device.

1. Build and run the app.
2. The app will ask you for location and notification permissions. You need to allow them.
3. Tap on `Cognito Configuration` button.
4. If you have not filled the values in `Config.xcconfig` file you need to fill out the field with the resources values you have created previously in the configuration screen.

```
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`
```

5. Save the configuration
6. You can now see the Filter options for time, distance and accuracy. Use them as per your need.
7. Go to `Tracking` tab in the app and you will see the map and `Start Tracking` button.
8. If you have installed the app on a simulator you may want to simulate location changes. This can be done in Features -> Location menu option. For example select Features -> Location -> Freeway Drive.
9. Tap on `Start Tracking` button. You should see the tracking points on the map.
10. The app is also tracking the locations in the background. So, when you move the app in the background it will ask for your permission to continue tracking in background mode.
11. You can stop the tracking by tapping on `Stop Tracking` button.

Create an Android application

Follow these procedures to build an iOS application using Amazon Location Service.

Clone the project files from [GitHub](#).

Create Amazon Location resources for your app

You can generate Amazon Location Service resources once your AWS account is ready. These resources will be essential for executing the provided code snippets.

Note

If you haven't created an AWS account yet, please [create an AWS account](#).

To begin you will need to create a Amazon Cognito Identity Pool Id, use the following procedure:

1. In the AWS console, navigate to the Amazon Cognito service and then select **Identity pools** from the left side menu and select **Create Identity pool**.
2. Make sure **Guest Access** is checked, and press **Next** to continue.
3. Next create a new IAM role or Use an existing IAM role.
4. Enter an Identity pool name, and make sure Identity Pool has access to Amazon Location (geo)resources for the map and tracker you will be creating in the next procedure.
- 5.

Now you need to create and style a map in the AWS Amazon Location console, use the following procedure:

1. Navigate to the [Maps section](#) in the Amazon Location console and select **Create Map** to preview available map styles.
2. Give the new map resource a **Name** and **Description**. Record the name you assign to the map resource, as it is used later in the tutorial.
3. When choosing a map style, consider the map data provider. Refer to section 82 of the [AWS service terms](#) for more details.
4. Accept the [Amazon Location Terms and Conditions](#), then select **Create Map**. After map has been created, you can interact with the map by zooming in, out, or panning in any direction.

To create a tracker using the Amazon Location console

1. Open the [Amazon Location Service console](#).
2. In the left navigation pane, choose **Trackers**.
3. Choose **Create tracker**.
4. Fill in the all the required fields.
5. Under **Position filtering**, choose the option that best fits how you intend to use your tracker resource. If you do not set Position filtering, the default setting is TimeBased. For more information, see Trackers in this guide, and PositionFiltering in the Amazon Location Service Trackers API Reference.
6. Choose **Create tracker** to finish.

Create a Geofence Collection

When creating a geofence collection you can use either the console, API or CLI. The following procedures walk you through each option.

Create a geofence collection using the Amazon Location console:

1. Open the Amazon Location Service console at <https://console.aws.amazon.com/location/>.
2. In the left navigation pane, choose Geofence Collections.
3. Choose Create geofence collection.
4. Provide a name and description for the collection.
5. Under EventBridge rule with CloudWatch as a target, you can create an optional EventBridge rule to get started reacting to geofence events. This enables Amazon Location to publish events to Amazon CloudWatch Logs.
6. Choose Create geofence collection.

Create a geofence collection using the Amazon Location APIs:

Use the CreateGeofenceCollection operation from the Amazon Location Geofences APIs. The following example uses an API request to create a geofence collection called *GOECOLLECTION_NAME*.

```
POST /geofencing/v0/collections
Content-type: application/json
```

```
{
  "CollectionName": "GOECOLLECTION_NAME",
  "Description": "Geofence collection 1 for shopping center",
  "Tags": {
    "Tag1" : "Value1"
  }
}
```

Create a geofence collection using AWS CLI commands:

Use the `create-geofence-collection` command. The following example uses an AWS CLI to create a geofence collection called `GOECOLLECTION_NAME`.

```
aws location \ create-geofence-collection \
  --collection-name "GOECOLLECTION_NAME" \
  --description "Shopping center geofence collection" \
  --tags Tag1=Value1
```

Link a tracker to a geofence collection

To link a tracker to a geofence collection you can use either the console, API, or CLI. The following procedures walk you through each option.

Link a tracker resource to a geofence collection using the Amazon Location Service console:

1. Open the Amazon Location console.
2. In the left navigation pane, choose **Trackers**.
3. Under **Device Trackers**, select the name link of the target tracker.
4. Under **Linked Geofence Collections**, choose **Link Geofence Collection**.
5. In the **Linked Geofence Collection window**, select a geofence collection from the dropdown menu.
6. Choose **Link**.
7. After you link the tracker resource, it will be assigned an Active status.

Link a tracker resource to a geofence collection using the Amazon Location APIs:

Use the `AssociateTrackerConsumer` operation from the Amazon Location Trackers APIs. The following example uses an API request that associates `ExampleTracker` with a geofence collection using its Amazon Resource Name (ARN).

```
POST /tracking/v0/trackers/ExampleTracker/consumers
Content-type: application/json
{
  "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GOECOLLECTION_NAME"
}
```

Link a tracker resource to a geofence collection using AWS CLI commands:

Use the `associate-tracker-consumer` command. The following example uses an AWS CLI to create a geofence collection called *GOECOLLECTION_NAME*.

```
aws location \
associate-tracker-consumer \
  --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/GOECOLLECTION_NAME" \
  --tracker-name "ExampleTracker"
```

Use AWS Lambda with MQTT

Create a Lambda function:

To create a connection between AWS IoT Core and Amazon Location Service, you need an AWS Lambda function to process messages forwarded by EventBridge CloudWatch events. This function will extract any positional data, format it for Amazon Location Service, and submit it through the Amazon Location Tracker API. You can create this function through the AWS Lambda console, or you can use the AWS Command Line Interface (AWS CLI) or the AWS Lambda APIs. To create a Lambda function that publishes position updates to Amazon Location using the console:

1. Open the AWS Lambda console at <https://console.aws.amazon.com/lambda/>.
2. From the left navigation, choose Functions.
3. Choose Create Function, and make sure that Author from scratch is selected.
4. Fill out the following boxes:
 - a Function name
 - for the **Runtime** option, choose Node.js 16.x.
5. Choose Create function.
6. Choose the Code tab to open the editor.
7. Overwrite the placeholder code in `index.js` with the following:

```
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("Lambda triggered====>>>", trackerEvent); // successful
    response
        }
    });
}
});
}
```

8. Choose Deploy to save the updated function.
9. Choose the Configuration tab.
10. In the Triggers section, click on Add trigger.
11. Select EventBridge (CloudWatch Events) in Source field.
12. Select `Existing Rules` radio option.
13. Enter the rule name like this `AmazonLocationMonitor-GEOFENCECOLLECTION_NAME`.
14. Click on the Add button.
15. This will also attach `Resource-based policy statements` in permissions tab

MQTT Test Client


1. Open the <https://console.aws.amazon.com/iot/>.
2. In the left navigation pane, choose MQTT test client.
3. You'll see a section titled **MQTT test client** where you can configure your MQTT connection.
4. After configuring the necessary settings, click on the **Connect** button to establish a connection to the MQTT broker using the provided parameters.
5. Note down Endpoint value.

Once connected, you can subscribe to MQTT topics or publish messages to topics using the respective input fields provided in the MQTT test client interface. Next you will attach the MQTT Policy:

1. On the left side menu, under **Manage** expand **Security** option and click on **Policies**.
2. Click on **Create Policy** button.
3. Enter a policy name.
4. On **Policy Document** select **JSON** tab.

5. Copy paste the policy shown below, but make sure to update all elements with your **REGION** and **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"
    }
  ]
}
```

 **Note**

Record the policy name and topic name for use in the next procedure.

6. Select the **Create** button to finish.

After completing the previous procedure, you will now update the permissions for the guest role as follows:

1. Navigate to Amazon Cognito and open your identity pool. Then, proceed to user access and select the guest role.
2. Click on permission policies to enable editing.


```

{
  '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. With the above policy changes, all necessary AWS resources are now configured appropriately for the application.

Set up the sample app code

1. Open Android Studio and select **New** and then **Project from Version Control**.
2. Navigate to the **File** menu in Android Studio's top left corner.
3. Select "New" from the dropdown menu.
4. Choose "Project from Version Control".
5. Enter Repository URL In the dialogue box that appears, find the field marked "URL".
6. Copy and paste the following URL for the sample app into this field:<https://github.com/aws-geospatial/amazon-location-mobile-sdk-sample-app-android.git>
7. Decide on the directory where you want to clone the repository. Use either the default directory or opt for a custom location.
8. After setting the repository URL and directory preferences, hit the "Clone" button. Android Studio will proceed to clone the repository to your specified location.
9. You have now cloned the application to your machine and can begin using it.

Use the sample app

To use the sample follow these procedures:

- **Create a custom.properties:**

To configure your custom.properties file, follow these steps:

1. Open your preferred text editor or IDE.
2. Create a new file.
3. Save the file with the name `custom.properties`.
4. Update the `custom.properties` with the following code sample, and replace the `MQTT_END_POINT`, `POLICY_NAME`, `GEOFENCE_COLLECTION_NAME`, and `TOPIC_TRACKER` with actual values:

```
MQTT_END_POINT=xxxxxxxxxxxxxxx-xxx.xxx.us-east-1.amazonaws.com
POLICY_NAME=xxxxxxxxxxx
GEOFENCE_COLLECTION_NAME=xxxxxxxxxxxxxxxxxxxxxxx
TOPIC_TRACKER=xxxxxxxxxxx
```

5. Clean and Rebuild the project. After this, you can run the project.

- **Sign In:**

To sign in to the application, follow the below steps:

1. Press the **Sign In** button.
2. Provide an **Identity Pool Id**, **Tracker name**, and a **Map name**.
3. Press **Sign In** again to finish.

- **Manage Filters:**

Open the configuration screen, and perform the following:

1. Toggle filters on/off using the switch UI.
2. Update Time and Distance filters when needed.

- **Tracking Operations:**

Open the tracking screen and perform the following:

- You can start and stop tracking in foreground, background, or in battery-saver mode by pressing respective buttons.

Authenticate with Amazon Location Service

To use Amazon Location Service, a user must be granted access to the resources and APIs that make up Amazon Location. There are three strategies you can use to grant access to your resources.

- **Use API keys** – To grant access to unauthenticated users, you can create API keys that give read-only access to your Amazon Location Service resources and actions. This is useful in a case where you don't want to authenticate every user. For example, a web application.

For more information about API keys, see [the section called “Use API keys”](#).

- **Use Amazon Cognito** – An alternative to API keys is to use Amazon Cognito to grant anonymous access. Amazon Cognito allows you to create a richer authorization with policy to define what can be done by the unauthenticated users.

For more information about using Amazon Cognito, see [the section called “Use Amazon Cognito”](#).

- **Use AWS Identity and Access Management (IAM)** – To grant access to users authenticated with AWS IAM Identity Center or AWS Identity and Access Management (IAM), create an IAM policy that allows access to the resources that you want.

For more information about IAM and Amazon Location, see [the section called “Use IAM”](#).

Choose an authentication method

API keys and Amazon Cognito are used in similar ways for similar scenarios, so why would you choose one instead of the other? The following list highlights some of the differences between the two.

- **Performance:**
 - **API key:** Relatively faster
 - **Amazon Cognito:** Relatively slower
- **Availability:**
 - **API key:** Amazon Location APIs for Maps, Places, and Routes
 - **Amazon Cognito:** All APIs
- **Combines with another authentication method?**

- **API key:** No
- **Amazon Cognito:** Yes

Comparison

- API keys are available only for maps, places, and routes actions. Amazon Cognito can be used to authenticate access to most Amazon Location Service APIs.
- The performance of map requests with API keys is typically faster than similar scenarios with Amazon Cognito. Simpler authentication means fewer round trips to the service and cached requests when getting the same map tile again in a short time period.
- With Amazon Cognito, you can use your own authentication process or combine multiple authentication methods, using Amazon Cognito Federated Identities.

For more information, see [Getting Started with Federated Identities](#) in the Amazon Cognito Developer Guide.

Use API keys to authenticate

Note

API keys are available to use only with **map**, **place**, and **route** resources, and you can't modify or create those resources. If your application needs access to other resources or actions for unauthenticated users, you can use Amazon Cognito to provide access along with, or instead of, API keys. For more information, see [the section called "Use Amazon Cognito"](#).

API keys are a key value that is associated with specific Amazon Location Service resources or API in your AWS account, and specific actions that you can perform on those resources. You can use an API key in your application to make unauthenticated calls to the Amazon Location APIs for those resources.

For example, if you associate an API key with a resource and/or the GetPlace* API, then an application that uses that API key will be able to call specific APIs. That same API key would not give permissions to change or update any resource or call APIs that it isn't associated with.

When you call Amazon Location Service APIs in your applications, you typically make this call as an *authenticated user* who is authorized to make the API calls. However, there are some cases where you don't want to authenticate every user of your application.

For example, you might want a web application that shows your business location to be available to anyone using the website, whether they are logged in or not. In this case, one alternative is to use API keys to make the API calls.

See [the section called “Best practices”](#) for additional information about when to use API keys.

Create an API key for Amazon Location Service

You can create an API key through the Amazon Location Service console, AWS CLI, or Amazon Location API. Continue with the appropriate procedures below.

Amazon Location console

To create an API key using the Amazon Location Service console

1. In the [Amazon Location console](#), choose **API keys** from the left menu.
2. On the **API keys** page, choose **Create API key**.
3. On the **Create API key** page, fill in the following information:
 - **Name** – A name for your API key, such as `ExampleKey`.
 - **Description** – An optional description for your API key.
 - **Resources** – In the dropdown, choose the Amazon Location resources to give access to with this API key. You can add more than one resource by choosing **Add resource**.
 - **Actions** – Specify the actions you want to authorize with this API key. You must select at least one action to match each resource type you have selected. For example, if you selected a place resource, you must select at least one of the choices under **Places Actions**.
 - **Expiration time** – Optionally, add an expiration date and time for your API key. For more information, see [the section called “Best practices”](#).
 - **Referrers** – Optionally, add one or more domains where you can use the API key. For example, if the API key is to allow an application running on the website `example.com`, then you could put `*.example.com/` as an allowed referrer.
 - **Tags** – Optionally, add tags to the API key.

4. Choose **Create API key** to create the API key.
5. On the detail page for the API key, you can see information about the API key that you have created. Choose **Show API key** to see the key value that you use when calling Amazon Location APIs. The key value will have the format `v1.public.a1b2c3d4...`

AWS CLI

1. Use the [create-key](#) command. The following example creates an API key called `ExampleKey` with no expiration date and access to a single map resource.

```
aws location \  
  create-key \  
    --key-name ExampleKey \  
    --restrictions '{"AllowActions":["geo:GetMap*"],"AllowResources":  
["arn:aws:geo:region:map/mapname"]}' \  
    --no-expiry
```

2. The response includes the API key value to use when accessing resources in your applications. The key value will have the format `v1.public.a1b2c3d4...`. To learn more about using the API key to render maps, see [the section called "Call an API"](#). The response to `create-key` looks like the following:

```
{  
  "Key": "v1.public.a1b2c3d4...",  
  "KeyArn": "arn:aws:geo:region:accountId:api-key/ExampleKey",  
  "KeyName": "ExampleKey",  
  "CreateTime": "2023-02-06T22:33:15.693Z"  
}
```

3. You can also use `describe-key` to find the key value at a later time. The following example shows how to call `describe-key` on an API key named `ExampleKey`.

```
aws location describe-key \  
  --key-name ExampleKey
```

Amazon Location API

Use the [CreateKey](#) operation from the Amazon Location APIs. The following example is an API request to create an API key called `ExampleKey` with no expiration date and access to a single map resource.

```
POST /metadata/v0/keys HTTP/1.1
Content-type: application/json
{
  "KeyName": "ExampleKey",
  "NoExpiry": true,
  "Restrictions": {
    "AllowActions": [
      "geo-places:*",
      "geo-routes:*",
      "geo-maps:*"
    ],
    "AllowResources": [
      "arn:aws:geo-places:Region::provider/default",
      "arn:aws:geo-routes:Region::provider/default",
      "arn:aws:geo-maps:Region::provider/default"
    ]
  }
}
```

The response includes the API key value to use when accessing resources in your applications. The key value will have the format `v1.public.a1b2c3d4....`

You can also use the [DescribeKey](#) API to find the key value for a key at a later time.

Use an API key to call an Amazon Location API

After you create an API key, you can use the key value to make calls to Amazon Location APIs in your application.

API

The APIs that support API keys have an additional parameter that takes the API key value. For example, if you call the `GetPlace` API, you can fill in the [key](#) parameter, as follows

```
curl --request GET --url 'https://places.geo.eu-central-1.amazonaws.com/v2/place/
{PLACEID}?key={APIKEY}&language=jp'
```


AWS CLI

When you use the `--key` parameter, you should also use the `--no-sign-request` parameter, to avoid signing with Sig v4.

```
aws geo-places get-place --place-id $PLACEID --language jp --key $APIKEY
```

SDK (web)

Use the following code:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Display a map</title>
    <meta property="og:description" content="Initialize a map in an HTML element
with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>
    <style>
      body { margin: 0; }
      #map { height: 100vh; }
    </style>
  </head>
  <body>

    <div id="map"></div>
    <script>

      const apiKey = "<api key>"; // check how to create api key for Amazon
Location
      const mapStyle = "Standard"; // eg. Standard, Monochrome, Hybrid,
Satellite
      const awsRegion = "eu-central-1"; // eg. us-east-2, us-east-1, us-
west-2, ap-south-1, ap-southeast-1, ap-southeast-2, ap-northeast-1, ca-central-1,
eu-central-1, eu-west-1, eu-west-2, eu-south-2, eu-north-1, sa-east-1
      const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/styles/
${mapStyle}/descriptor?key=${apiKey}`;
```

```
        const map = new maplibregl.Map({
            container: 'map', // container id
            style: styleUrl, // style URL
            center: [25.24,36.31], // starting position [lng, lat]
            zoom: 2, // starting zoom
        });
    </script>
</body>
</html>
```

SDK (iOS, Swift)

Use the following code:

```
import UIKit
import MapLibre

class ViewController: UIViewController {
    let apiKey = "Enter your API key" // The previously-created API Key to use
    let regionName = "Enter your region name" // The service region - us-east-1, ap-
south-1, etc
    var mapView: MLNMapView!

    override func viewDidLoad() {
        super.viewDidLoad()
        loadMap()
    }

    func loadMap() {
        let styleName = "Standard" // The map style - Standard, Monochrome, Hybrid,
Satellite
        let colorName = "Light" // The color scheme - Light, Dark

        // The Amazon Location Service map style URL that MapLibre will use to
render the maps.
        let styleURL = URL(string: "https://maps.geo.\(regionName).amazonaws.com/v2/
styles/\(styleName)/descriptor?key=\(apiKey)&color-scheme=\(colorName)")

        // Initialize MapLibre
        mapView = MLNMapView(frame: view.bounds)
        mapView.styleURL = styleURL
        mapView.autoresizingMask = [.flexibleWidth, .flexibleHeight]
        // Set the starting camera position and zoom level for the map
```

```
        mapView.setCenter(CLLocationCoordinate2D(latitude: 49.246559, longitude:
-123.063554), zoomLevel: 10, animated: false)
        view.addSubview(mapView!)
    }
}
```

SDK (Android, Kotlin)

Use the following code:

```
class MapActivity : Activity(), OnMapReadyCallback {

    private lateinit var mBinding: ActivityMapBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        initializeMap(savedInstanceState)
    }

    private fun initializeMap(savedInstanceState: Bundle?) {
        // Init MapLibre
        // See the MapLibre Getting Started Guide for more details
        // https://maplibre.org/maplibre-native/docs/book/android/getting-started-
guide.html
        MapLibre.getInstance(this@MapActivity)
        mBinding = ActivityMapBinding.inflate(layoutInflater)
        setContentView(mBinding.root)
        mBinding.mapView.onCreate(savedInstanceState)
        mBinding.mapView.getMapAsync(this)
    }

    override fun onMapReady(mapLibreMap: MapLibreMap) {
        mapLibreMap.setStyle(Style.Builder().fromUri(getMapUrl())) {
            // Set the starting camera position
            mapLibreMap.cameraPosition =
CameraPosition.Builder().target(LatLng(49.246559, -123.063554)).zoom(10.0).build()
            mapLibreMap.uiSettings.isLogoEnabled = false
            mapLibreMap.uiSettings.attributionGravity = Gravity.BOTTOM or
Gravity.END

            mapLibreMap.uiSettings.setAttributionDialogManager(AttributionDialogManager(this,
mapLibreMap))
        }
    }
}
```

```
// Return the Amazon Location Service map style URL
// MapLibre will use this to render the maps.
// awsRegion: The service region - us-east-1, ap-south-1, etc
// mapStyle: The map style - Standard, Monochrome, Hybrid, Satellite
// API_KEY: The previously-created API Key to use
// colorName: The color scheme to use - Light, Dark
private fun getMapUrl() =
    "https://maps.geo.${getString(R.string.awsRegion)}.amazonaws.com/v2/
styles/${getString(R.string.mapStyle)}/descriptor?key=${BuildConfig.API_KEY}&color-
scheme=${getString(R.string.colorName)}"

override fun onStart() {
    super.onStart()
    mBinding.mapView.onStart()
}

override fun onResume() {
    super.onResume()
    mBinding.mapView.onResume()
}

override fun onPause() {
    super.onPause()
    mBinding.mapView.onPause()
}

override fun onStop() {
    super.onStop()
    mBinding.mapView.onStop()
}

override fun onSaveInstanceState(outState: Bundle) {
    super.onSaveInstanceState(outState)
    mBinding.mapView.onSaveInstanceState(outState)
}

override fun onLowMemory() {
    super.onLowMemory()
    mBinding.mapView.onLowMemory()
}

override fun onDestroy() {
    super.onDestroy()
}
```

```
        mBinding.mapView.onDestroy()  
    }  
}
```

API key best practices

API keys include a plain text *value* that gives access to one or more resources or APIs in your AWS account. If someone copies your API key, they can access those same resources and APIs. To minimize the potential impact, review the following best practices:

- **Limit the API key**

To avoid the situation above, it is best to limit your API key. When you create the key, you can specify the domain or referrer where the key can be used.

- **Manage API key lifetimes**

You can create API keys that work indefinitely. However, if you want to create a temporary API key, rotate API keys on a regular basis, or revoke an existing API key, you can use *API key expiration*.

- You can set the expiration time for an API key when you create or update it.
- When an API key reaches its expiration time, the key is automatically deactivated. Inactive keys can no longer be used to make requests.
- You can change a temporary key to a permanent key by removing the expiration time.
- You can delete an API key 90 days after deactivating it.
- If you attempt to deactivate an API key that has been used within the last seven days, you'll be prompted to confirm that you want to make the change.

If you are using the Amazon Location Service API or the AWS CLI, set the `ForceUpdate` parameter to `true`, otherwise you'll receive an error.

Use Amazon Cognito to authenticate

You can use Amazon Cognito authentication as an alternative to directly using AWS Identity and Access Management (IAM) users with frontend SDK requests.

Amazon Cognito provides authentication, authorization, and user management for web and mobile apps. You can use Amazon Cognito unauthenticated identity pools with Amazon Location as a way for applications to retrieve temporary, scoped-down AWS credentials.

For more information, see [Getting Started with User Pools](#) in the *Amazon Cognito Developer Guide*.

You may want to use this form of authentication for the following reasons:

- **Unauthenticated users** – If you have a website with anonymous users, you can use Amazon Cognito identity pools.

For more information, see the section on [the section called “Use Amazon Cognito”](#).

- **Your own authentication** – If you would like to use your own authentication process, or combine multiple authentication methods, you can use Amazon Cognito Federated Identities.

For more information, see [Getting Started with Federated Identities](#) in the *Amazon Cognito Developer Guide*.

Use Amazon Cognito and Amazon Location Service

You can use AWS Identity and Access Management (IAM) policies associated with unauthenticated identity roles with the following actions:

Maps

List of maps actions

- `geo-maps:GetStaticMap`
- `geo-maps:GetTile`

Note

Resource names for the actions above are:

```
arn:aws:geo-maps:region::provider/default
```

Places

List of place actions:

- `geo-places:Geocode`
- `geo-places:ReverseGeocode`
- `geo-places:SearchNearby`
- `geo-places:SearchText`
- `geo-places:Autocomplete`
- `geo-places:Suggest`
- `geo-places:GetPlace`

Note

Resource names for the actions above are:

```
arn:aws:geo-places:region::provider/default
```

Routes

List of routes actions:

- `geo-routes:CalculateRoutes`
- `geo-routes:CalculateRouteMatrix`
- `geo-routes:CalculateIsolines`
- `geo-routes:OptimizeWaypoints`
- `geo-routes:SnapToRoads`

Note

Resource names for the actions above are:

```
arn:aws:geo-routes:region::provider/default
```

Geofences and Trackers

List of Geofences and Trackers actions

- `geo:GetGeofence`
- `geo:ListGeofences`
- `geo:PutGeofence`
- `geo:BatchDeleteGeofence`
- `geo:BatchPutGeofence`
- `geo:BatchEvaluateGeofences`
- `geo:GetDevicePosition*`
- `geo:ListDevicePositions`
- `geo:BatchDeleteDevicePositionHistory`
- `geo:BatchGetDevicePosition`
- `geo:BatchUpdateDevicePosition`

Note

Resource names for the actions above are:

```
arn:aws:geo:region:accountID:tracker/ExampleTracker
```

Previous version

List of previous version actions:

- `geo:GetMap*`
- `geo:SearchPlaceIndexForText`
- `geo:SearchPlaceIndexForPosition`
- `geo:GetPlace`
- `geo:CalculateRoute`
- `geo:CalculateRouteMatrix`

Note

Resource names for the actions above are:

Maps

```
arn:aws:geo:region:accountID:map/ExampleMap
```

Places

```
arn:aws:geo:region:accountID:place-index/ExamplePlaceIndex
```

Routes

```
arn:aws:geo:region:accountID:route-calculator/ExampleCalculator
```

Create an Amazon Cognito identity pool

You can create Amazon Cognito identity pools to allow unauthenticated guest access to your application through the Amazon Cognito console, the AWS CLI, or the Amazon Cognito APIs.

⚠ Important


The pool that you create must be in the same AWS account and AWS Region as the Amazon Location Service resources that you're using.

Console

To create an identity pool using the Amazon Cognito console

1. Go to the [Amazon Cognito console](#).
2. Choose **Manage Identity Pools**.
3. Choose **Create new identity pool**, then enter a name for your identity pool.
4. From the **Unauthenticated identities** collapsible section, choose **Enable access to unauthenticated identities**.
5. Choose **Create Pool**.

6. Choose which IAM roles you want to use with your identity pool.
7. Expand **View Details**.
8. Under **Unauthenticated identities**, enter a role name.
9. Expand the **View Policy Document** section, then choose **Edit** to add your policy.
10. Add your policy to grant access to your resources.

 **Note**

See the [the section called “Use Amazon Cognito and Amazon Location Service”](#) section above for a list of actions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      Sid: "RoutesReadOnly",
      Effect: "Allow",
      Action": [
        // add comma separated value from the previous section
      ],
      Resource: "value from previous section"
    }
  ]
}
```

11. Choose **Allow** to create your identity pools.

Use the Amazon Cognito identity pool in web

The following example exchanges the unauthenticated identity pool that you created for credentials that are then used to call `CalculateIsolines`. To simplify this work, the example uses the Amazon Location [the section called “Use authentication helpers”](#) procedures. This is in place of both getting and refreshing the credentials.

This example uses the AWS SDK for JavaScript v3.

```
import { GeoRoutesClient, CalculateIsolinesCommand , } from "@aws-sdk/client-geo-routes"; // ES Modules import
```

```
import { withIdentityPoolId } from "@aws/amazon-location-utilities-auth-helper";

const identityPoolId = "<identity pool ID>"; // for example, us-
east-1:1sample4-5678-90ef-aaaa-1234abcd56ef

const authHelper = await withIdentityPoolId(identityPoolId);

const client = new GeoRoutesClient({
  ...authHelper.getClientConfig(),
  region: "<region>", // The region containing the identity pool
});

const input = {
  DepartNow: true,
  TravelMode: "Car",
  Origin: [-123.12327, 49.27531],
  Thresholds: {
    Time: [5, 10, 30],
  },
};

const command = new CalculateIsolinesCommand(input);
const response = await client.send(command);

console.log(JSON.stringify(response, null, 2))
```

Use AWS Identity and Access Management to authenticate

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon Location resources. IAM is an AWS service that you can use with no additional charge.

Topics

- [Audience](#)
- [Authenticating with identities](#)
- [Managing access using policies](#)
- [How Amazon Location Service works with IAM](#)
- [How Amazon Location Service works with unauthenticated users](#)

- [Identity-based policy examples for Amazon Location Service](#)
- [Troubleshooting Amazon Location Service identity and access](#)

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Amazon Location.

Service user – If you use the Amazon Location service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon Location features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon Location, see [Troubleshooting Amazon Location Service identity and access](#).

Service administrator – If you're in charge of Amazon Location resources at your company, you probably have full access to Amazon Location. It's your job to determine which Amazon Location features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon Location, see [How Amazon Location Service works with IAM](#).

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon Location. To view example Amazon Location identity-based policies that you can use in IAM, see [Identity-based policy examples for Amazon Location Service](#).

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see [How to sign in to your AWS account](#) in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see [AWS Signature Version 4 for API requests](#) in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see [Multi-factor authentication](#) in the *AWS IAM Identity Center User Guide* and [AWS Multi-factor authentication in IAM](#) in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see [Tasks that require root user credentials](#) in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For

information about IAM Identity Center, see [What is IAM Identity Center?](#) in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An [IAM user](#) is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see [Rotate access keys regularly for use cases that require long-term credentials](#) in the *IAM User Guide*.

An [IAM group](#) is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see [Use cases for IAM users](#) in the *IAM User Guide*.

IAM roles

An [IAM role](#) is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. To temporarily assume an IAM role in the AWS Management Console, you can [switch from a user to an IAM role \(console\)](#). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see [Methods to assume a role](#) in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see [Create a role for a third-party identity provider \(federation\)](#) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see [Permission sets](#) in the *AWS IAM Identity Center User Guide*.

- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see [Cross account resource access in IAM](#) in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
 - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see [Forward access sessions](#).
 - **Service role** – A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Create a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.
 - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Use an IAM role to grant permissions to applications running on Amazon EC2 instances](#) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see [Overview of JSON policies](#) in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Define custom IAM permissions with customer managed policies](#) in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see [Choose between managed policies and inline policies](#) in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that

support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see [Access control list \(ACL\) overview](#) in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see [Permissions boundaries for IAM entities](#) in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see [Service control policies](#) in the *AWS Organizations User Guide*.

- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about Organizations and RCPs, including a list of AWS services that support RCPs, see [Resource control policies \(RCPs\)](#) in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see [Session policies](#) in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see [Policy evaluation logic](#) in the *IAM User Guide*.

How Amazon Location Service works with IAM

Before you use IAM to manage access to Amazon Location, learn what IAM features are available to use with Amazon Location.

IAM features you can use with Amazon Location Service

IAM feature	Amazon Location support
the section called "Identity-based policies"	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes

IAM feature	Amazon Location support
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Principal permissions	No
Service roles	No
Service-linked roles	No

To get a high-level view of how Amazon Location and other AWS services work with most IAM features, see [AWS services that work with IAM](#) in the *IAM User Guide*.

Identity-based policies for Amazon Location

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Define custom IAM permissions with customer managed policies](#) in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see [IAM JSON policy elements reference](#) in the *IAM User Guide*.

Identity-based policy examples for Amazon Location

To view examples of Amazon Location identity-based policies, see [Identity-based policy examples for Amazon Location Service](#).

Resource-based policies within Amazon Location

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see [Cross account resource access in IAM](#) in the *IAM User Guide*.

Policy actions for Amazon Location

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of Amazon Location actions, see [Actions Defined by Amazon Location Service](#) in the *Service Authorization Reference*.

Policy actions in Amazon Location use the following prefix before the action:

```
geo
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [  
  "geo:action1",  
  "geo:action2"  
]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word Get, include the following action:

```
"Action": "geo:Get*"
```

To view examples of Amazon Location identity-based policies, see [Identity-based policy examples for Amazon Location Service](#).

Policy resources for Amazon Location

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The Resource JSON policy element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. As a best practice, specify a resource using its [Amazon Resource Name \(ARN\)](#). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

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

To see a list of Amazon Location resource types and their ARNs, see [Resources Defined by Amazon Location Service](#) in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see [Actions Defined by Amazon Location Service](#).

To view examples of Amazon Location identity-based policies, see [Identity-based policy examples for Amazon Location Service](#).

Policy condition keys for Amazon Location

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or *Condition block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use [condition operators](#), such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple `Condition` elements in a statement, or multiple keys in a single `Condition` element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see [IAM policy elements: variables and tags](#) in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see [AWS global condition context keys](#) in the *IAM User Guide*.

To see a list of Amazon Location condition keys, see [Condition Keys for Amazon Location Service](#) in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see [Actions Defined by Amazon Location Service](#).

Amazon Location supports condition keys to allow you to allow or deny access to specific geofences or devices in your policy statements. The following condition keys are available:

- `geo:GeofenceIds` for use with Geofence actions. The type is `ArrayOfString`.
- `geo:DeviceIds` for use with Tracker actions. The type is `ArrayOfString`.

The following actions can be used with `geo:GeofenceIds` in your IAM policy:

- `BatchDeleteGeofences`

- BatchPutGeofences
- GetGeofence
- PutGeofence

The following actions can be used with `geo:DeviceIds` in your IAM policy:

- BatchDeleteDevicePositionHistory
- BatchGetDevicePosition
- BatchUpdateDevicePosition
- GetDevicePosition
- GetDevicePositionHistory

 **Note**

You can't use these condition keys with the `BatchEvaluateGeofences`, `ListGeofences`, or `ListDevicePosition` actions.

To view examples of Amazon Location identity-based policies, see [Identity-based policy examples for Amazon Location Service](#).

ACLs in Amazon Location

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with Amazon Location

Supports ABAC (tags in policies): Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then

you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the [condition element](#) of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see [Define permissions with ABAC authorization](#) in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see [Use attribute-based access control \(ABAC\)](#) in the *IAM User Guide*.

For more information about tagging Amazon Location resources, see [the section called "How to use tags"](#).

To view an example identity-based policy for limiting access to a resource based on the tags on that resource, see [Control resource access based on tags](#).

Using temporary credentials with Amazon Location

Supports temporary credentials: Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see [AWS services that work with IAM](#) in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see [Switch from a user to an IAM role \(console\)](#) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate

temporary credentials instead of using long-term access keys. For more information, see [Temporary security credentials in IAM](#).

Cross-service principal permissions for Amazon Location

Supports forward access sessions (FAS): No

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see [Forward access sessions](#).

Service roles for Amazon Location

Supports service roles: No

A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Create a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.

Warning

Changing the permissions for a service role might break Amazon Location functionality. Edit service roles only when Amazon Location provides guidance to do so.

Service-linked roles for Amazon Location

Supports service-linked roles: No

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see [AWS services that work with IAM](#). Find a service in the table that includes a Yes in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

How Amazon Location Service works with unauthenticated users

Many scenarios for using Amazon Location Service, including showing maps on the web or in a mobile application, require allowing access to users who haven't signed in with IAM. For these unauthenticated scenarios, you have two options.

- **Use API keys** – To grant access to unauthenticated users, you can create API Keys that give read-only access to your Amazon Location Service resources. This is useful in a case where you do not want to authenticate every user. For example, a web application. For more information about API keys, see [the section called “Use API keys”](#).
- **Use Amazon Cognito** – An alternative to API keys is to use Amazon Cognito to grant anonymous access. Amazon Cognito allows you to create a richer authorization with IAM policy to define what can be done by the unauthenticated users. For more information about using Amazon Cognito, see [the section called “Use the identity pool in web”](#).

For an overview of providing access to unauthenticated users, see [Authentication](#).

Identity-based policy examples for Amazon Location Service

By default, users and roles don't have permission to create or modify Amazon Location resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see [Create IAM policies \(console\)](#) in the *IAM User Guide*.

For details about actions and resource types defined by Amazon Location, including the format of the ARNs for each of the resource types, see [Actions, Resources, and Condition Keys for Amazon Location Service](#) in the *Service Authorization Reference*.

Topics

- [Policy best practices](#)
- [Using the Amazon Location console](#)
- [Allow users to view their own permissions](#)
- [Using Amazon Location Service resources in policy](#)

- [Permissions for updating device positions](#)
- [Read-only policy for tracker resources](#)
- [Policy for creating geofences](#)
- [Read-only policy for geofences](#)
- [Permissions for rendering a map resource](#)
- [Permissions to allow search operations](#)
- [Read-only policy for route calculators](#)
- [Control resource access based on condition keys](#)
- [Control resource access based on tags](#)

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon Location resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see [AWS managed policies](#) or [AWS managed policies for job functions](#) in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see [Policies and permissions in IAM](#) in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see [IAM JSON policy elements: Condition](#) in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides

more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see [Validate policies with IAM Access Analyzer](#) in the *IAM User Guide*.

- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see [Secure API access with MFA](#) in the *IAM User Guide*.

For more information about best practices in IAM, see [Security best practices in IAM](#) in the *IAM User Guide*.

Using the Amazon Location console

To access the Amazon Location Service console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon Location resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can use the Amazon Location console, attach the following policy to the entities. For more information, see [Adding permissions to a user](#) in the *IAM User Guide*.

The following policy gives access to the Amazon Location Service console, to be able to create, delete, list and view details about Amazon Location resources in your AWS account.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GeoPowerUser",
      "Effect": "Allow",
      "Action": [
        "geo:*",
        "geo-maps:*",
        "geo-places:*",
        "geo-routes:*"
      ]
    }
  ]
}
```

```

    ],
    "Resource": "*"
  }
]
}

```

Alternatively, you can grant read-only permissions to facilitate read-only access. With read-only permissions, an error message will appear if the user attempts write actions such as creating or deleting resources. As an example, see [the section called “Read-only policy for trackers”](#)

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupForUser",
        "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": "*"
  }
]
}

```

Using Amazon Location Service resources in policy

Amazon Location Service uses the following prefixes for resources:

Amazon Location resource prefix

Resource	Resource prefix
Map resources	map
Place resources	place-index
Route resources	route-calculator
Tracking resources	tracker
Geofence Collection resources	geofence-collection

Use the following ARN syntax:

```
arn:Partition:geo:Region:Account:ResourcePrefix/ResourceName
```

For more information about the format of ARNs, see [Amazon Resource Names \(ARNs\) and AWS Service Namespaces](#).

Examples

- Use the following ARN to allow access to a specified map resource.

```
"Resource": "arn:aws:geo:us-west-2:account-id:map/map-resource-name"
```

- To specify access to all map resources that belong to a specific account, use the wildcard (*):

```
"Resource": "arn:aws:geo:us-west-2:account-id:map/*"
```

- Some Amazon Location actions, such as those for creating resources, can't be performed on a specific resource. In those cases, you must use the wildcard (*).

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

To see a list of Amazon Location resource types and their ARNs, see [Resources Defined by Amazon Location Service](#) in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see [Actions Defined by Amazon Location Service](#).

Permissions for updating device positions

To update device positions for multiple trackers, you'll want to grant a user access to one or more of your tracker resources. You will also want to allow the user to update a batch of device positions.

In this example, in addition to granting access to the *Tracker1* and *Tracker2* resources, the following policy grants permission to use the `geo:BatchUpdateDevicePosition` action against the *Tracker1* and *Tracker2* resources.

```
{
  "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"
      ]
    }
  ]
}
```

If you want to limit the user to only be able to update device positions for a specific device, you can add a condition key for that device id.

```
{
  "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"
        ]
      }
    }
  }
]
}

```

Read-only policy for tracker resources

To create a read-only policy for all tracker resources in your AWS account, you'll need to grant access to all tracker resources. You'll also want to grant a user access to actions that allow them to get the device position for multiple devices, get the device position from a single device and get the position history.

In this example, the following policy grants permission to the following actions:

- `geo:BatchGetDevicePosition` to retrieve the position of multiple devices.
- `geo:GetDevicePosition` to retrieve the position of a single device.
- `geo:GetDevicePositionHistory` to retrieve the position history of a device.

```

{
  "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/*"
  }
]
}

```

Policy for creating geofences

To create a policy to allow a user to create geofences, you'll need to grant access to specific actions that allow users to create one or more geofences on a geofence collection.

The policy below grants permission to the following actions on *Collection*:

- `geo:BatchPutGeofence` to create multiple geofences.
- `geo:PutGeofence` to create a single geofence.

```

{
  "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"
    }
  ]
}

```

Read-only policy for geofences

To create a read-only policy for geofences stored in a geofence collection in your AWS account, you'll need to grant access to actions that read from the geofence collection storing the geofences.

The policy below grants permission to the following actions on *Collection*:

- `geo:ListGeofences` to list geofences in the specified geofence collection.
- `geo:GetGeofence` to retrieve a geofence from the geofence collection.

```
{
  "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"
    }
  ]
}
```

Permissions for rendering a map resource

To grant sufficient permissions to render maps, you'll need to grant access to map tiles, sprites, glyphs, and the style descriptor:

- `geo:GetMapTile` retrieves map tiles used to selectively render features on a map.
- `geo:GetMapSprites` retrieves the PNG sprite sheet and corresponding JSON document describing offsets within it.
- `geo:GetMapGlyphs` retrieves the glyphs used for displaying text.
- `geo:GetMapStyleDescriptor` retrieves the map's style descriptor, containing rendering rules.

```
{
  "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"
}
]
}
```

Permissions to allow search operations

To create a policy to allow search operations, you'll first need to grant access to the place index resource in your AWS account. You'll also want to grant access to actions that let the user search using text by geocoding and search using a position by reverse geocoding.

In this example, in addition to granting access to *PlaceIndex*, the following policy also grants permission to the following actions:

- `geo:SearchPlaceIndexForPosition` allows you to search for places, or points of interest near a given position.
- `geo:SearchPlaceIndexForText` allows you to search for an address, name, city or region using free-form text.

```

{
  "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"
    }
  ]
}
```

Read-only policy for route calculators

You can create a read-only policy to allow a user access to a route calculator resource to calculate a route.

In this example, in addition to granting access to *ExampleCalculator*, the following policy grants permission to the following operation:

- `geo:CalculateRoute` calculates a route given a departure position, destination position, and a list of waypoint positions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RoutesReadOnly",
      "Effect": "Allow",
      "Action": [
        "geo:CalculateRoute"
      ],
      "Resource": "arn:aws:geo:us-west-2:accountID:route-calculator/ExampleCalculator"
    }
  ]
}
```

Control resource access based on condition keys

When you create an IAM policy to grant access to use geofences or device positions, you can use [Condition operators](#) for more precise control over which geofences or devices a user can access. You can do this by including the geofence id or device id in the `Condition` element of your policy.

The following example policy shows how you might create a policy that allows a user to update device positions for a specific device.

```
{
  "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"
        ]
      }
    }
  ]
}
```

Control resource access based on tags

When you create an IAM policy to grant access to use your Amazon Location resources, you can use [attribute-based access control](#) for better control over which resources a user can modify, use, or delete. You can do this by including tag information in the `Condition` element of your policy to control access based on your resource [tags](#).

The following example policy shows how you might create a policy that allows a user to create geofences. This grants the permission to the following actions to create one or more geofences on a geofence collection called *Collection*:

- `geo:BatchPutGeofence` to create multiple geofences.
- `geo:PutGeofence` to create a single geofence.

However, this policy uses the `Condition` element to grant the permission only if the *Collection* tag, `Owner`, has the value of that user's user name.

- For example, if a user named `richard-roe` attempts to view an Amazon Location *Collection*, the *Collection* must be tagged `Owner=richard-roe` or `owner=richard-roe`. Otherwise the user is denied access.

Note

The condition tag key `Owner` matches both `Owner` and `owner` because condition key names are not case-sensitive. For more information, see [IAM JSON Policy Elements: Condition](#) in the *IAM User Guide*.

```
{
  "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}"}
      }
    }
  ]
}
```

For a tutorial about [how to define permissions to access AWS resources based on tags](#), see the *AWS Identity and Access Management User Guide*.

Troubleshooting Amazon Location Service identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon Location and IAM.

Topics

- [I am not authorized to perform an action in Amazon Location](#)
- [I am not authorized to perform iam:PassRole](#)
- [I want to allow people outside of my AWS account to access my Amazon Location resources](#)

I am not authorized to perform an action in Amazon Location

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional `my-example-widget` resource but doesn't have the fictional `geo:GetWidget` permissions.

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

In this case, the policy for the `mateojackson` user must be updated to allow access to the `my-example-widget` resource by using the `geo: GetWidget` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Amazon Location.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon Location. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:  
iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Amazon Location resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon Location supports these features, see [How Amazon Location Service works with IAM](#).
- To learn how to provide access to your resources across AWS accounts that you own, see [Providing access to an IAM user in another AWS account that you own](#) in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see [Providing access to AWS accounts owned by third parties](#) in the *IAM User Guide*.
- To learn how to provide access through identity federation, see [Providing access to externally authenticated users \(identity federation\)](#) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see [Cross account resource access in IAM](#) in the *IAM User Guide*.

Amazon Location Service Maps



The Amazon Location Service Map gives you access to the underlying base map data for 190 countries with 5 million daily updates. Through both static and dynamic map capabilities, you gain flexibility to cater to diverse user needs and deliver immersive, contextually relevant mapping solutions.

Static maps offer pre-rendered representations of geographic data, enabling you to embed high-quality visuals that enhance reports and provide spatial context in emails. Dynamic maps allow you to create interactive and responsive experiences, where users can pan, zoom, and explore the map in real-time, aligning with the requirements and preferences of your business. Whether you're showing real-time turn-by-turn navigation, visualizing location-based data, or enabling users to explore new areas, Amazon's services equip you with the tools to deliver tailored solutions that resonate with your audience.

Features

The Amazon Location Service offers dynamic maps and static maps.

Dynamic maps

You can use AWS Map Styles such as standard, monochrome, hybrid, and satellite. You can add an interactive map to your application using AWS Map Styles with a map rendering engine such as MapLibre. Dynamic maps also support map gestures such as zoom, pan, ease, fly, pitch, rotate, and bearing.

Static maps

You can use static map URLs to embed simple map images on your website, report, or email without the need for a map rendering engine. Static maps allow you to overlay markers (or pins), routes, and polygon areas as needed for your application.

Political view

To switch from the international perspective to a country-specific geopolitical view, use the political view parameter in your API query. This helps businesses comply with local laws, as certain countries require adherence to their specific geopolitical views for maps and map data.

Common use cases

Embed maps in your application

Build maps into your applications to create location-based experiences. Visualize business locations, search for points of interest, and help users find specific addresses. Enable seamless location sharing and geotagging features to engage your customers. Use comprehensive map data, robust geocoding, and flexible rendering to create customized, interactive maps tailored to your needs. Integrate dynamic, high-quality mapping experiences that drive user engagement and business insights into your application, whether you're building a directory, ride-sharing app, or social platform.

Static maps for reporting or printing

Seamlessly add images of street maps, satellite imagery, and location-based visuals into your websites, documents, and applications. Static maps enable you to create customizable map images that provide geographical context, without complex client-side rendering. Display delivery route on receipts, include location details in documents, or integrate maps into your digital experiences.

Analyze and visualize data

Overlay your data onto high-quality maps to uncover transformative spatial patterns and trends. Empower your teams to create customizable, interactive map visualizations with your geographic data. Use maps and your data to optimize site selection, plan infrastructure, or analyze market opportunities.

Enhance real estate experiences

Provide prospective buyers with comprehensive location context for real estate listings. Display the property's exact location, as well as surrounding neighborhood details like jurisdictional boundaries, local businesses, parks, and schools. Help customers find directions to your open houses. Create informative, location-centric real estate experiences that engage and inform your clients.

Build engaging travel experiences

Display dynamic maps showcasing destinations, with detailed street views and key geographical features. Highlight hotels, restaurants, and other points of interest for tourists and travelers. Plot outdoor amenities, like hiking trails, to help users plan their ideal itinerary.

Use maps to support disaster response efforts

Timely and accurate location information is critical during crises. Use mapping capabilities to build websites and applications that provide essential context to communities during pending disasters like fires, hurricanes, and floods. Display dynamic maps showcasing evacuation routes, safe shelters, road closures, and traffic congestion to help empower communities to quickly assess the situation and make informed decisions.

Standalone Map APIs

API Name	Short Description	Resources
GetStyleDescriptor	Retrieves the available map styles, such as standard, monochrome, hybrid, and satellite, that can be applied to maps.	AWS map styles and customization
GetTile	Fetches individual map tiles based on a specified style and zoom level, allowing for the rendering of maps at various levels of detail.	Tiles
GetStaticMap	Generates a static map image based on specific coordinates and parameters, useful for embedding maps in reports or emails.	the section called "Static maps"

Displaying Map

Topic	Short Description	Resources
Styling Dynamic Map	Amazon Location Service provides two options for styling your dynamic maps namely using predesigned AWS Map Styles and customizing map style using style descriptors.	the section called “Style dynamic maps” the section called “Standard map style” the section called “Monochrome map style” the section called “Hybrid map style”
Rendering Dynamic Map	Amazon Location Service recommends rendering maps using the MapLibre rendering engine. MapLibre is an engine for displaying maps in web or mobile applications.	the section called “Map Rendering SDK by language”
Customizing Static Map	How to customize static maps generated using Amazon Location Service.	the section called “Customize static maps”
Overlaying Static Map	Overlay on your static maps to enhance the map's visual representation.	the section called “Overlay on the static map”

Map concepts

This section provides foundational knowledge for using Amazon Location Service's mapping capabilities. It details various map design options available and also explains how you can leverage AWS Map Styles and Design principles to customize the look and feel of your maps, ensuring visual consistency and branding in applications.

In addition, this topic covers critical mapping concepts such as Maps terminology, localization, and internationalization.

Topics

- [Maps terminology](#)
- [Localization and internationalization](#)

Maps terminology

This section defines key terms related to Amazon Location Service Maps, providing insights into core concepts like basemaps, vector and raster data, map rendering, and map styles. Understanding these terms is essential for effectively utilizing the Amazon Location Service mapping APIs and resources.

Basemap

A basemap provides geographic context to your map, stored as vector tile layers. These tile layers include geographical features such as street names, buildings, and land use for visual reference.

Vector

Vector data consists of points, lines, and polygons, and is used to represent roads, locations, and areas on a map. Vector shapes can also be used as icons for markers on a map.

Raster

Raster data is image data made up of a grid, typically representing continuous data like terrain, satellite imagery, or heat maps. Raster images can also be used as icons or textures on a map.

Map Rendering

The map rendering library pulls data from Amazon Location Service at runtime, rendering the map based on the selected map resource. A map resource defines the data provider and map style. Amazon Location Service recommends using the MapLibre rendering engine.

Vector Tile

A vector tile is a format that stores map data using vector shapes. It adjusts to the display resolution and selectively renders features, while maintaining a small file size for optimal performance. Supported format: Mapbox Vector Tiles (MVT).

Map Style

A map style defines color and other styling information for the map data, determining how the map will appear when rendered. Amazon Location Service provides map styles based on the Mapbox GL style specification.

Glyph File

A binary file containing encoded Unicode characters, used by a map renderer to display labels.

Sprite File

A Portable Network Graphic (PNG) image file that contains small raster images and corresponding location descriptions in a JSON file. Used by a map renderer to display icons or textures on a map.

Bounding Box

A bounding box is defined by two diagonal corner points: the northwest (NW) (top-left) and southeast (SE) (bottom-right) points. These points are key for specifying the spatial extent of a map.

Localization and internationalization

Amazon Location Service supports localization features that allow developers to customize maps for specific languages and regions. This includes support for local place names and the ability to render maps in different languages.

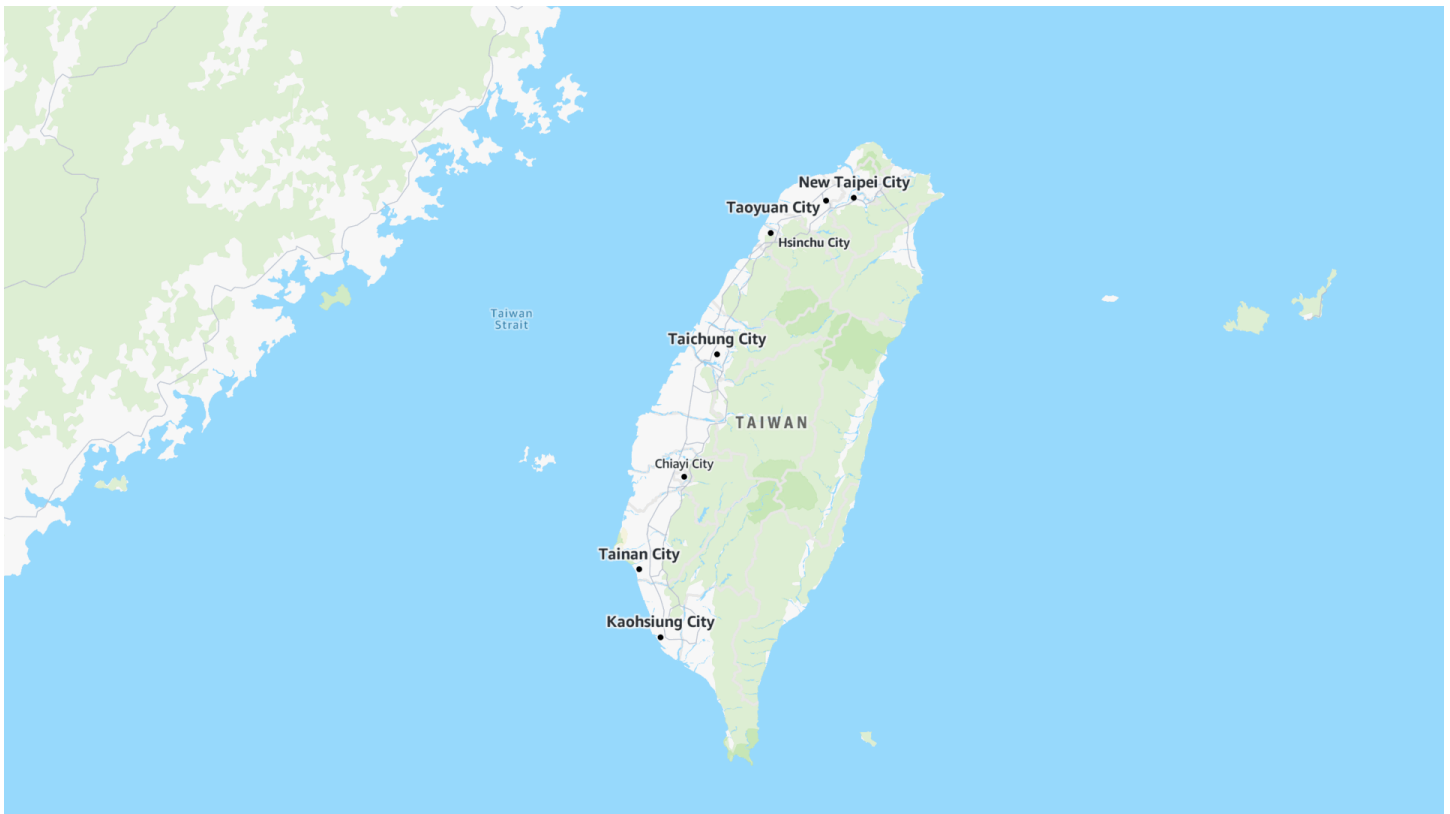
Style	Political View	Languages
Standard	Argentina, Cyprus, Egypt, Georgia, Greece, Kenya, Morocco, Palestine, Serbia, Russia, Sudan, Suriname, Syria, Türkiye, Tanzania, Uruguay	Supported through client-side library.
Monochrome	Argentina, Cyprus, Egypt, Georgia, Greece, Kenya, Morocco, Palestine, Serbia,	Supported through client-side library.

Style	Political View	Languages
	Russia, Sudan, Suriname, Syria, Türkiye, Tanzania, Uruguay	
Hybrid	Argentina, Cyprus, Egypt, Georgia, Greece, Kenya, Morocco, Palestine, Serbia, Russia, Sudan, Suriname, Syria, Türkiye, Tanzania, Uruguay	Supported through client-side library.
Satellite	Not required	Not required

Languages

Amazon Location Service provide Maps APIs that allow you to customize the language of map labels and text elements. This capability enables your applications to cater to a global audience or regions with multiple languages. By displaying maps in the user's preferred language, you enhance the overall user experience, making the maps more accessible and relevant to your diverse user base.

For more information, see [How to set a preferred language for a map](#).



Political view

By default, Amazon Location Service present an international perspective, which visually represents disputed territories with dashed borders. To switch from the international perspective to a country-specific geopolitical view, you must use the *political view* parameter in your API query. This helps businesses comply with local laws, as certain countries require adherence to their specific geopolitical views for maps and map data.

In addition to the default international perspective, Amazon Location Service supports the geopolitical views of the following countries: Argentina, Cyprus, Egypt, Georgia, Greece, Kenya, Morocco, Palestine, Serbia, Russia, Sudan, Suriname, Syria, Türkiye, Tanzania, and Uruguay. To activate a geopolitical view, pass the appropriate value to the *political view* parameter.

For more information, see [How to set the political view of a map](#).



AWS map styles and customization

Map styles overview

To request a map, you must choose first a map style. Map styles define the visual appearance of the rendered map, including the styling for map tiles, glyphs, and sprites. Map tiles can be either vector (MVT) or raster (image). While the style may change as you zoom in or out, it generally maintains a consistent theme. You can override parts or the entire style before passing it to the map rendering library.

AWS map styles

AWS map styles adhere to industry standards, offering a sophisticated and professional look. These styles reduce time-to-market and eliminate the need for dedicated cartographers to create map styles from scratch. These predesigned styles enable you to quickly and effectively create visually appealing maps for your end users.

By leveraging the predesigned AWS map styles, you can bypass the time-consuming and resource-intensive process of designing and constructing map styles from scratch. This accelerates your development process, allowing you to focus on core functionalities.

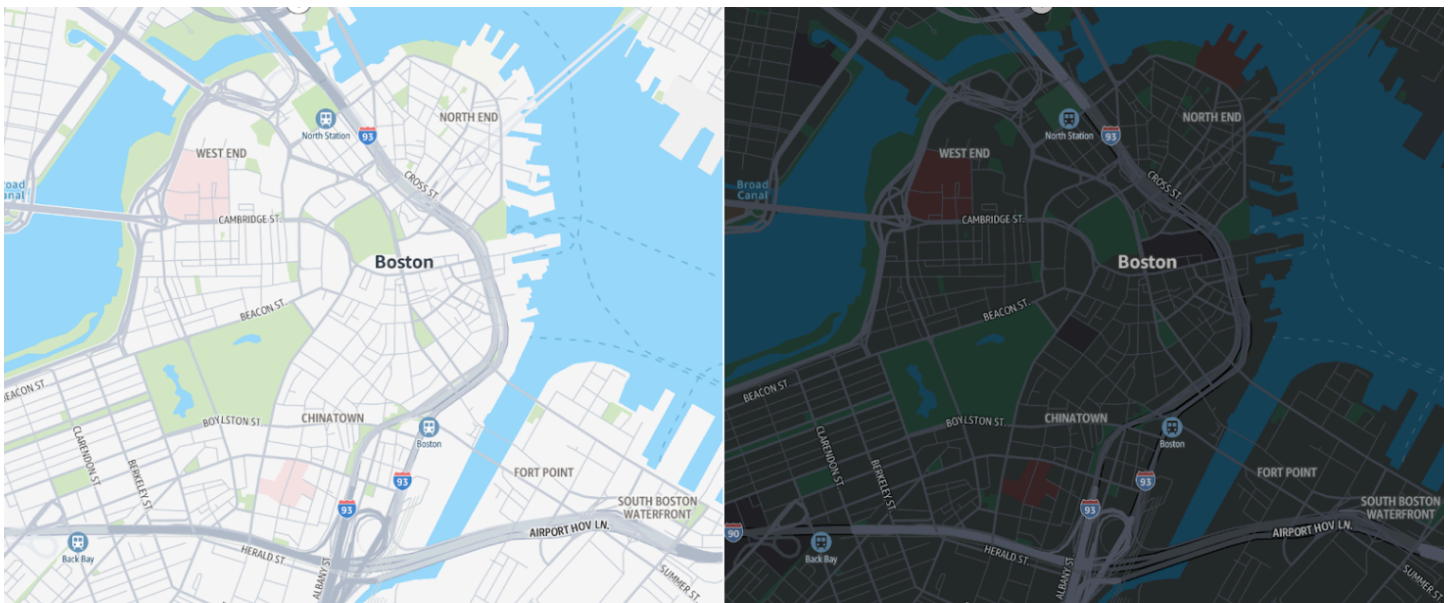
Map style name	Description	Color scheme	Supports
Standard	Colored map style	Dark and Light	Dynamic Map: Yes, Static Maps: No
Monochrome	Grey scale map styles	Dark and Light	Dynamic Map: Yes, Static Maps: No
Hybrid	Road and label overlay on satellite imagery	Not Applicable	Dynamic Map: Yes, Static Maps: No
Satellite	Satellite imagery-based map style	Not Applicable	Dynamic Map: Yes, Static Maps: Yes

Amazon Location Service provides styles following the [MapLibre GL style specification](#).

Standard map style

The Standard map style is a clean and modern general-purpose map design that fits beautifully and functionally into almost any application or website.

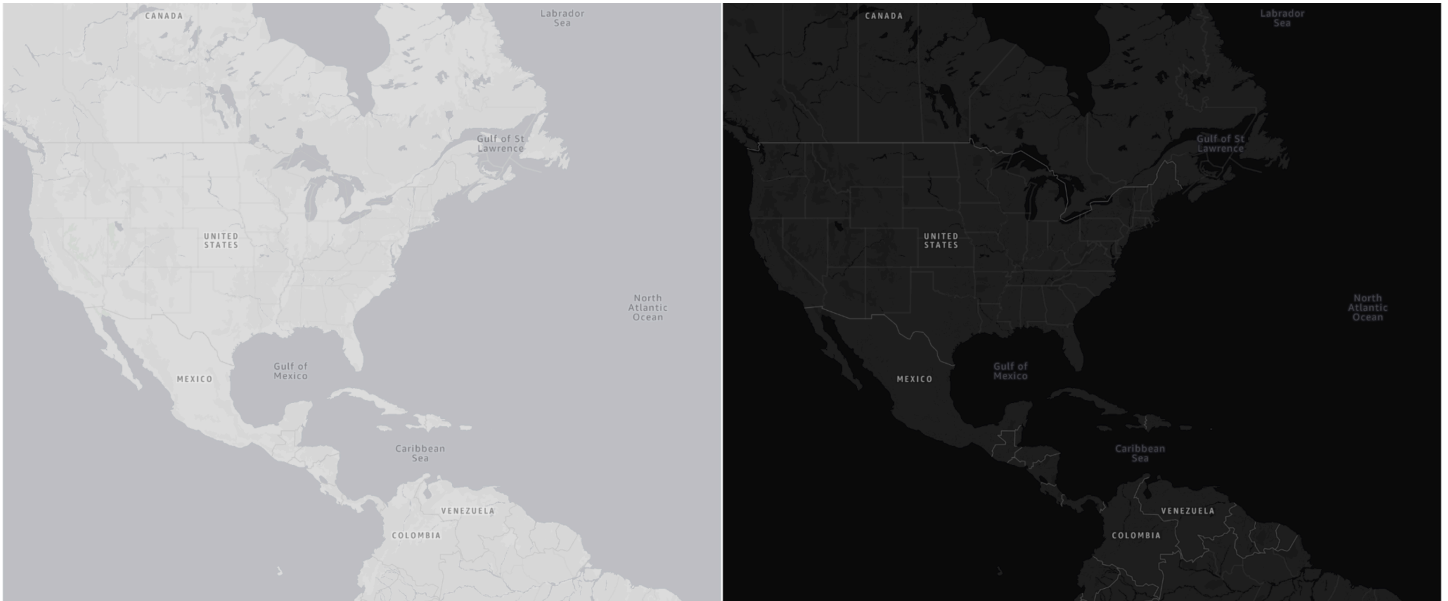
To learn more, see [Standard map style](#).



Monochrome map style

The Monochrome map style is a minimalist canvas with a constrained color palette, designed for use with data visualization overlays. The Monochrome style offers both light and dark modes, communicating all the essential information needed for geographic context.

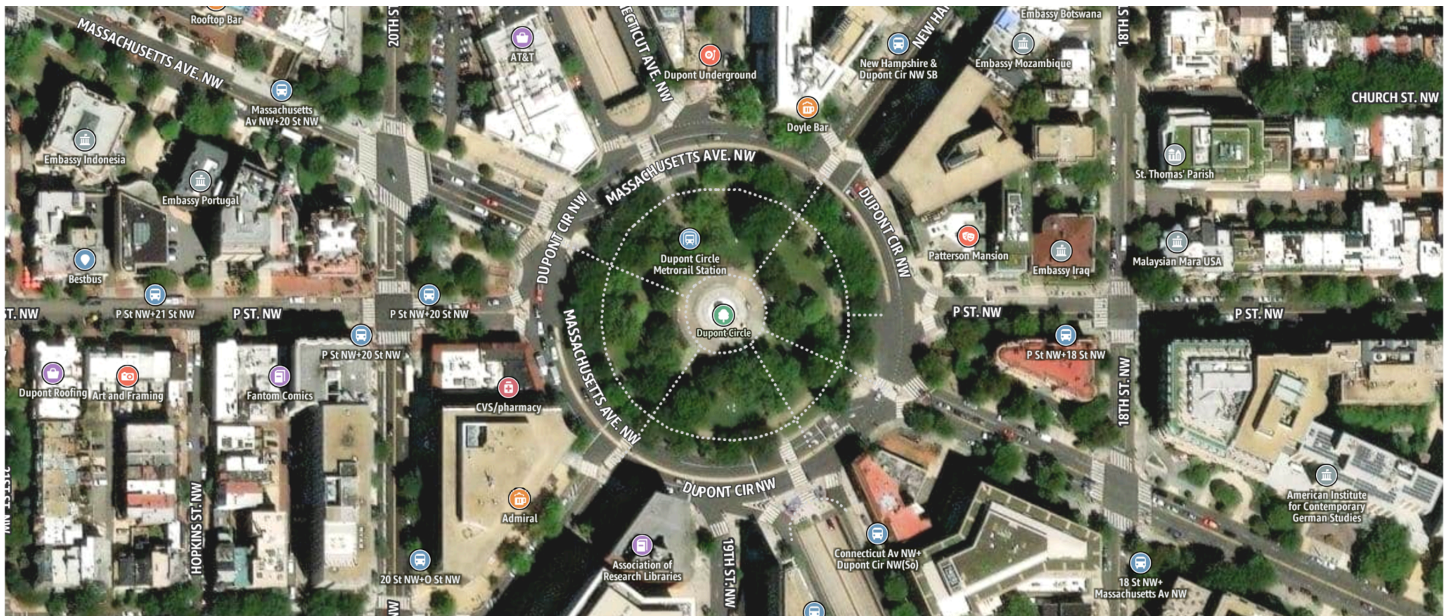
To learn more, see [the section called “Monochrome map style”](#).



Hybrid map style

The hybrid map style combines global satellite imagery with clear labels and configurable POI categories from our vector maps.

To learn more, see [the section called “Hybrid map style”](#).



Satellite map style

The Satellite map style presents high-resolution, real-world imagery captured by satellites, offering a realistic view of landscapes, buildings, and terrain. This style typically includes minimal labels or overlays to keep the focus on geographical details.



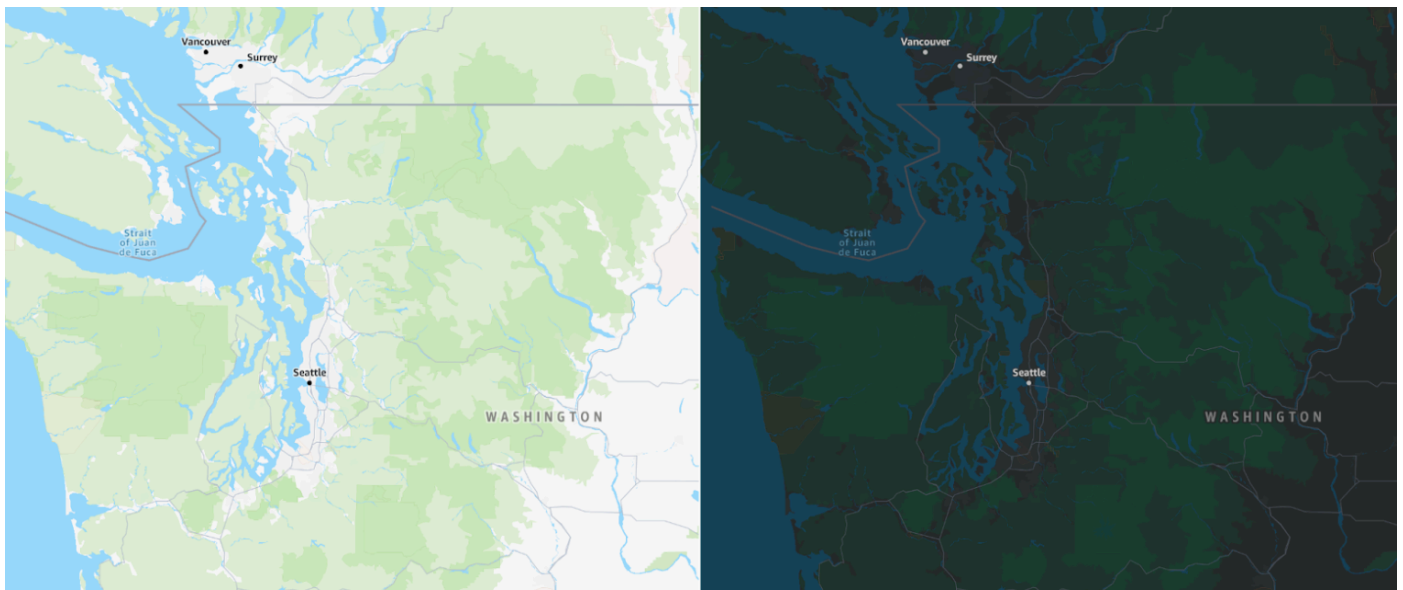
Standard map style

The Standard map style offers a clean, modern, and general-purpose map design that can seamlessly fit into almost any application or website.

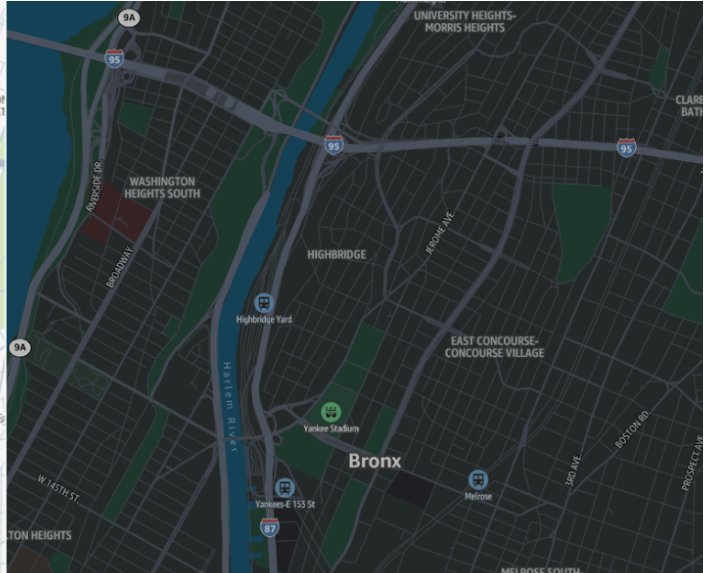
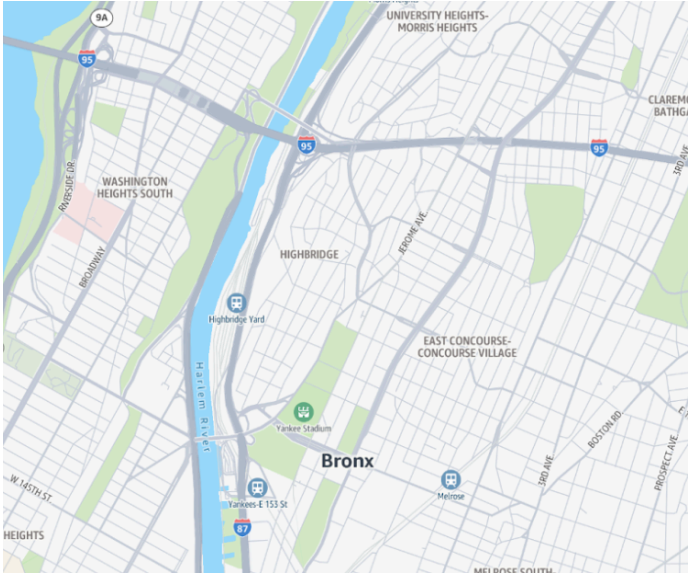
Color scheme

The Standard map style comes in both light and dark modes. The light mode is versatile and can fit into any context, while the dark mode features a constrained palette, designed to show details clearly and maintain readability in darker environments. This ensures minimal distractions, especially in scenarios such as night-time navigation.

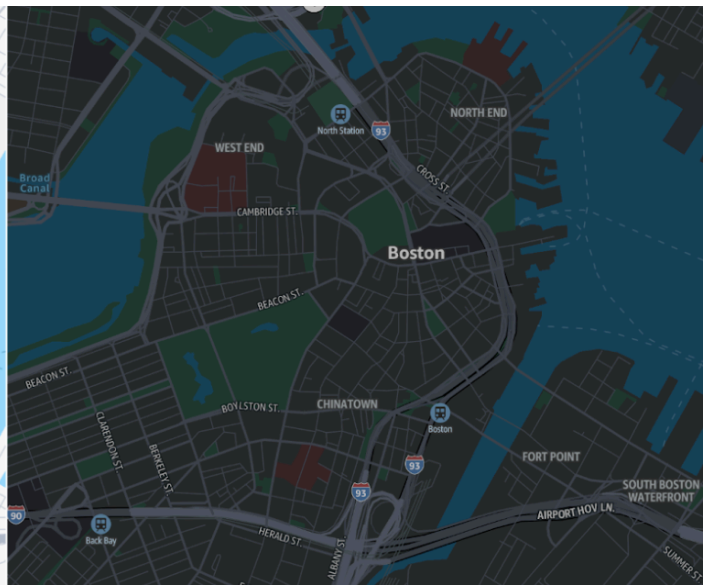
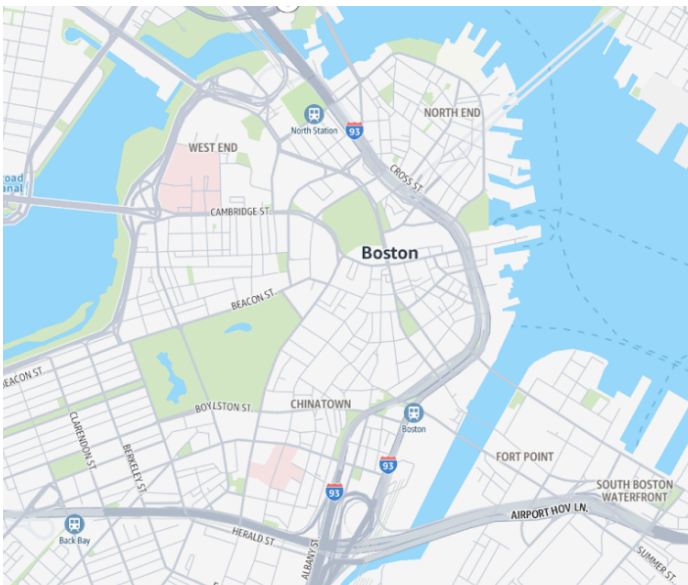
Forest



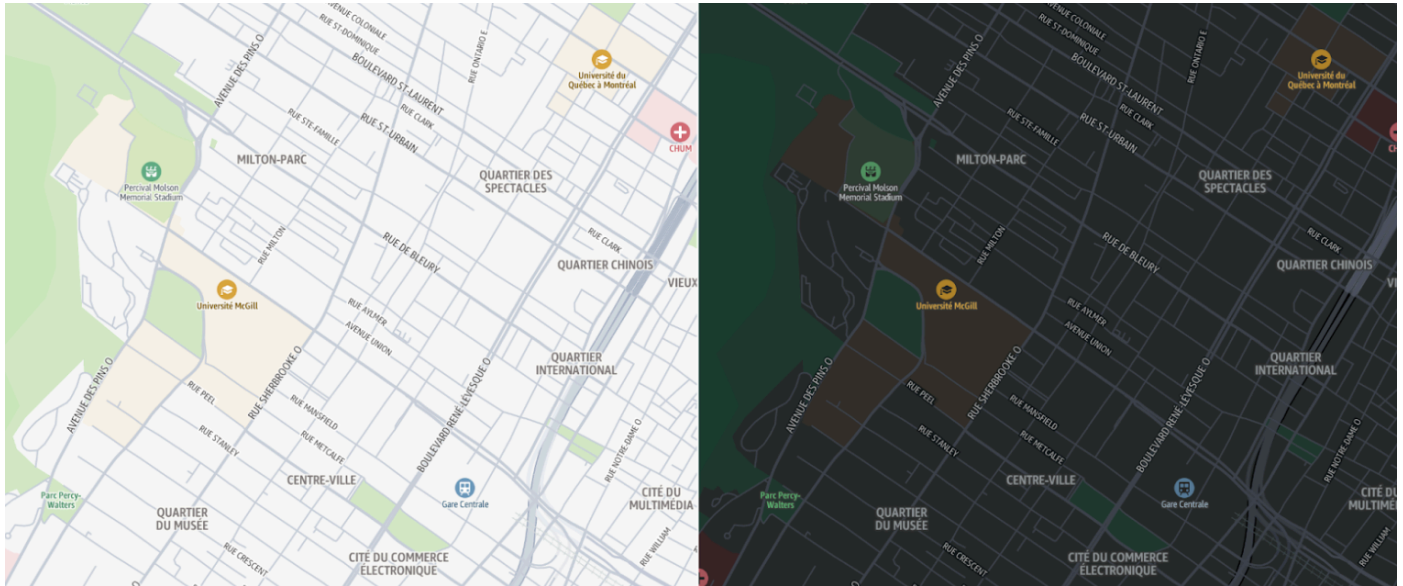
Road



City



Neighborhood

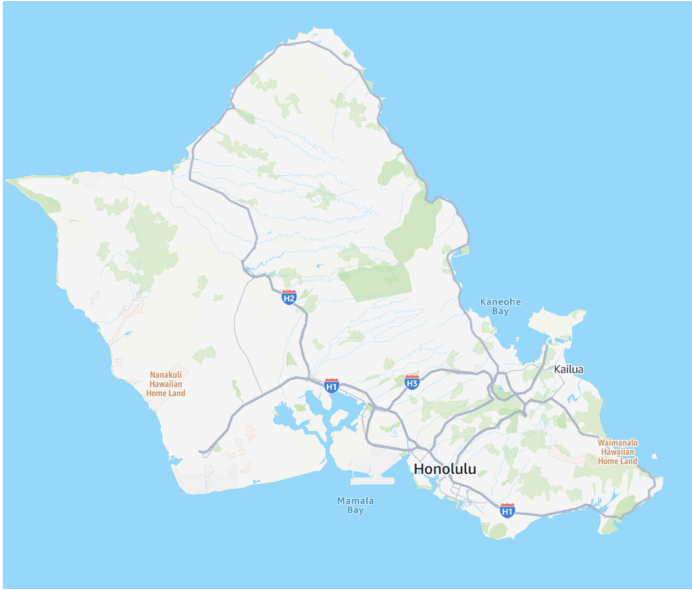


A pleasing, modern palette

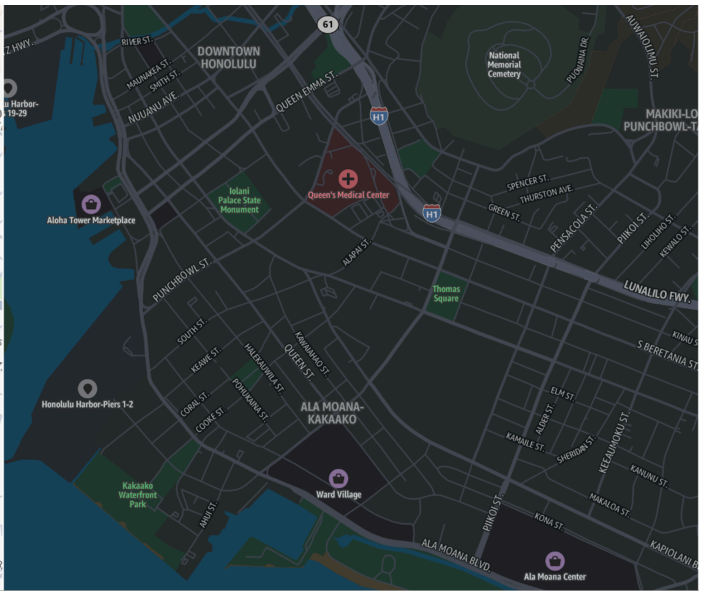
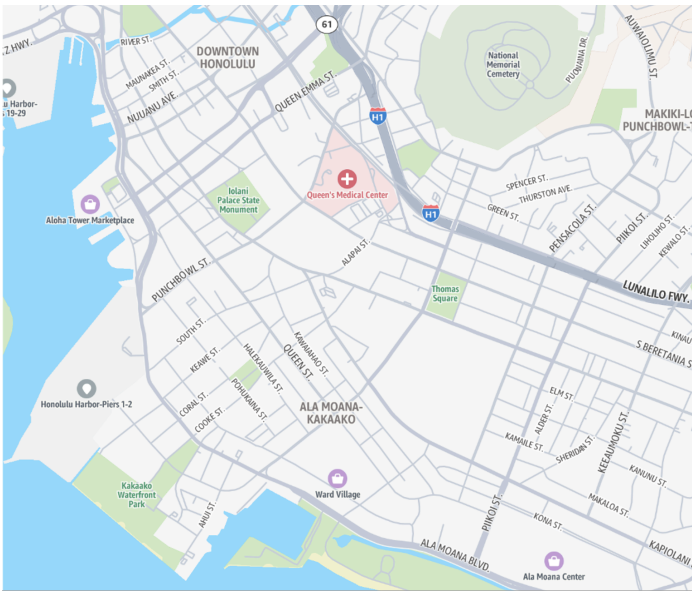
Soft colors provide important land-use context without overwhelming the map, offering useful information at both high and low zoom levels. Zoomed out, features such as forests, deserts, and glaciers add richness to the map. When zoomed in, a range of colors highlights important landmarks like schools, hospitals, recreation areas (like parks and sports facilities), and urban districts like commercial and industrial zones.

The overall style features a cohesive color palette, including POI markers that complement their respective land-use areas. The road network is displayed in shades of gray, providing detail without overwhelming the map with bright, distracting colors.

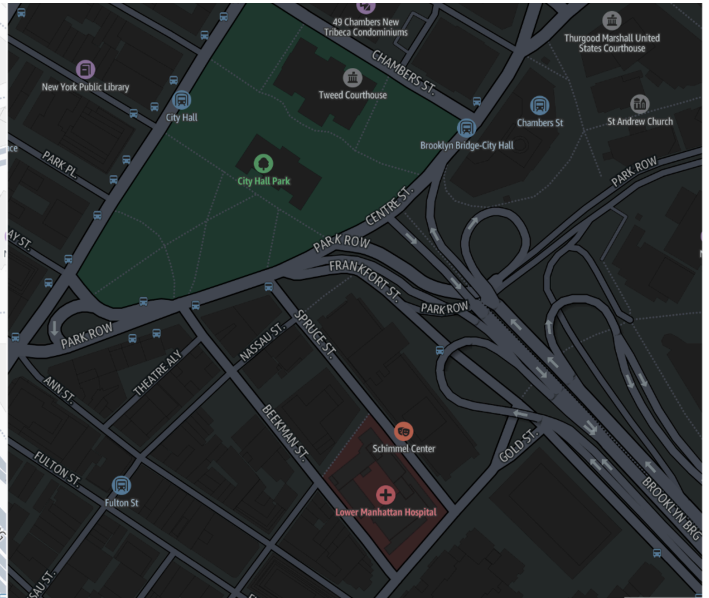
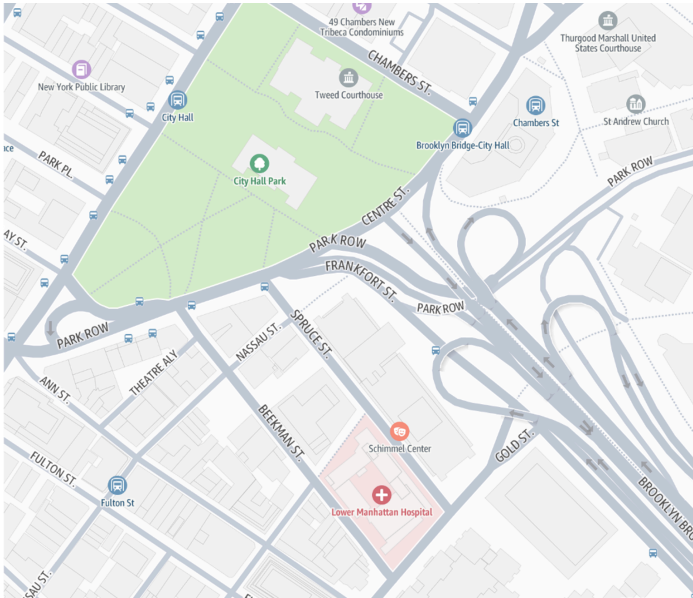
Island



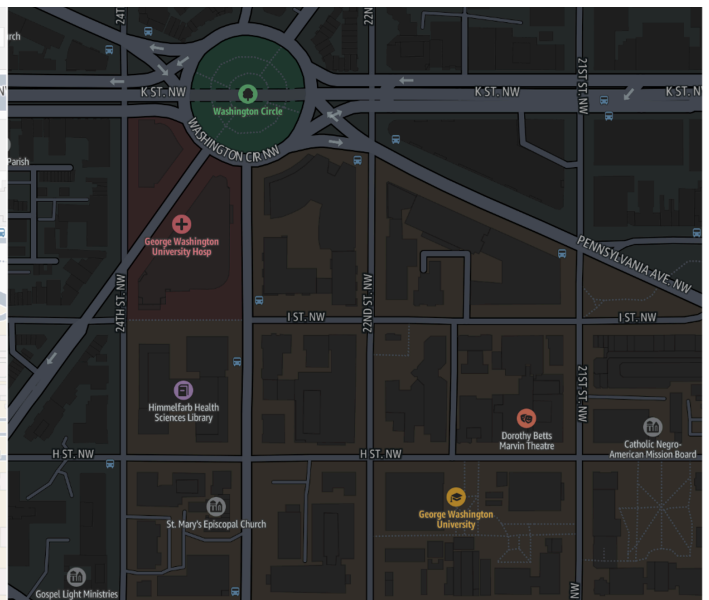
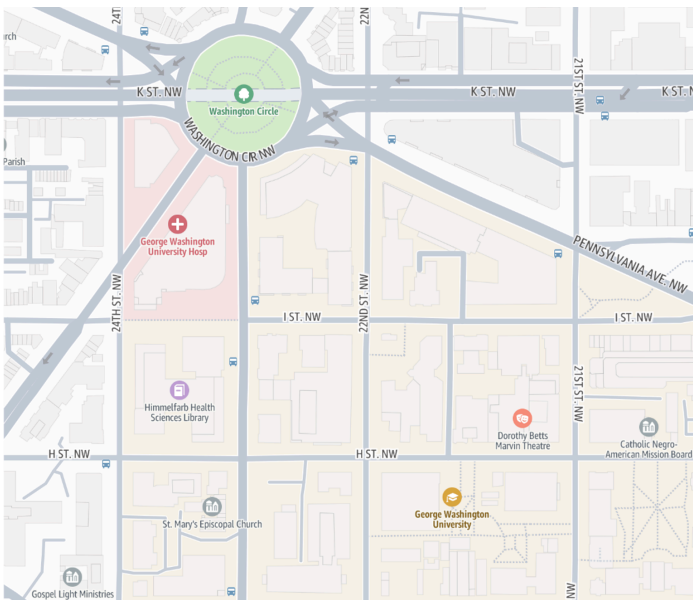
Neighborhood



Intersection

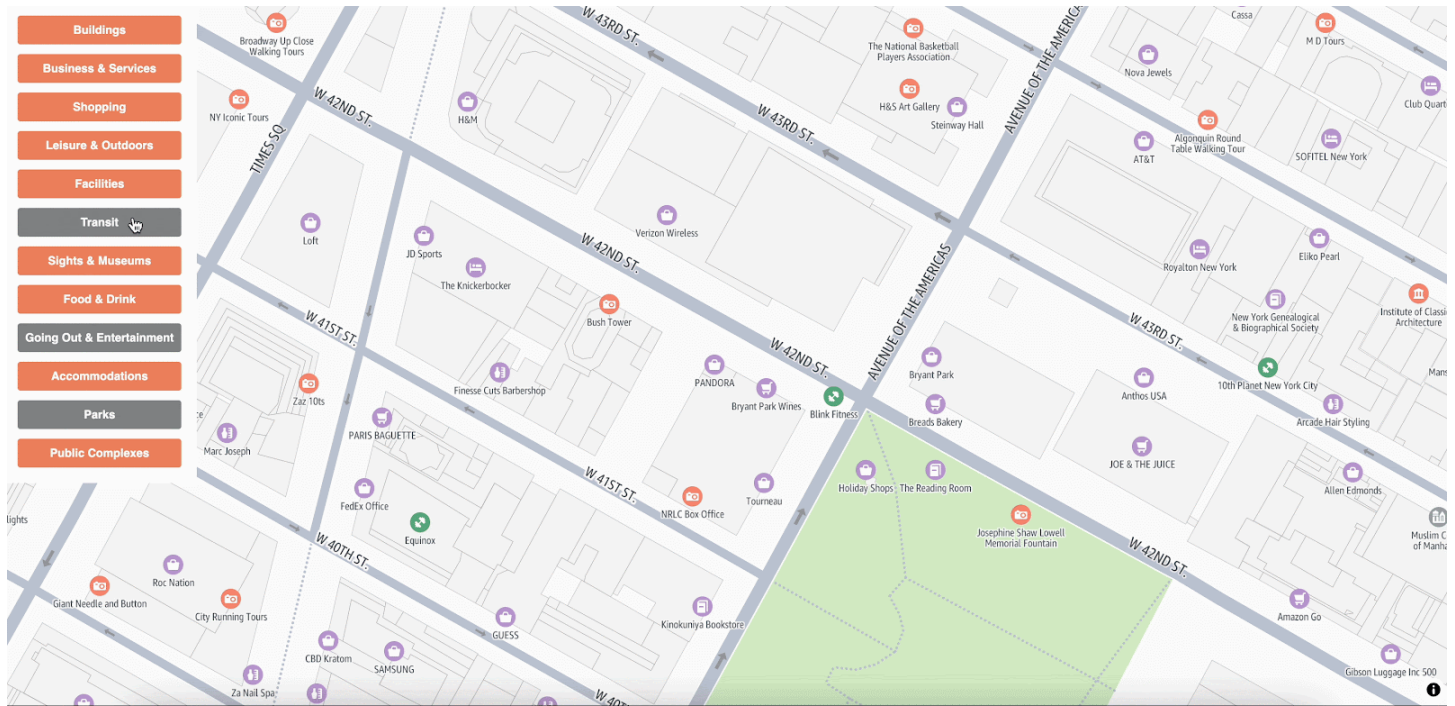


Roundabout



Rich points of interest (POI)

The Standard map style supports a rich array of configurable points of interest (POIs). With just a few lines of code, you can select the POI categories relevant to your use case.

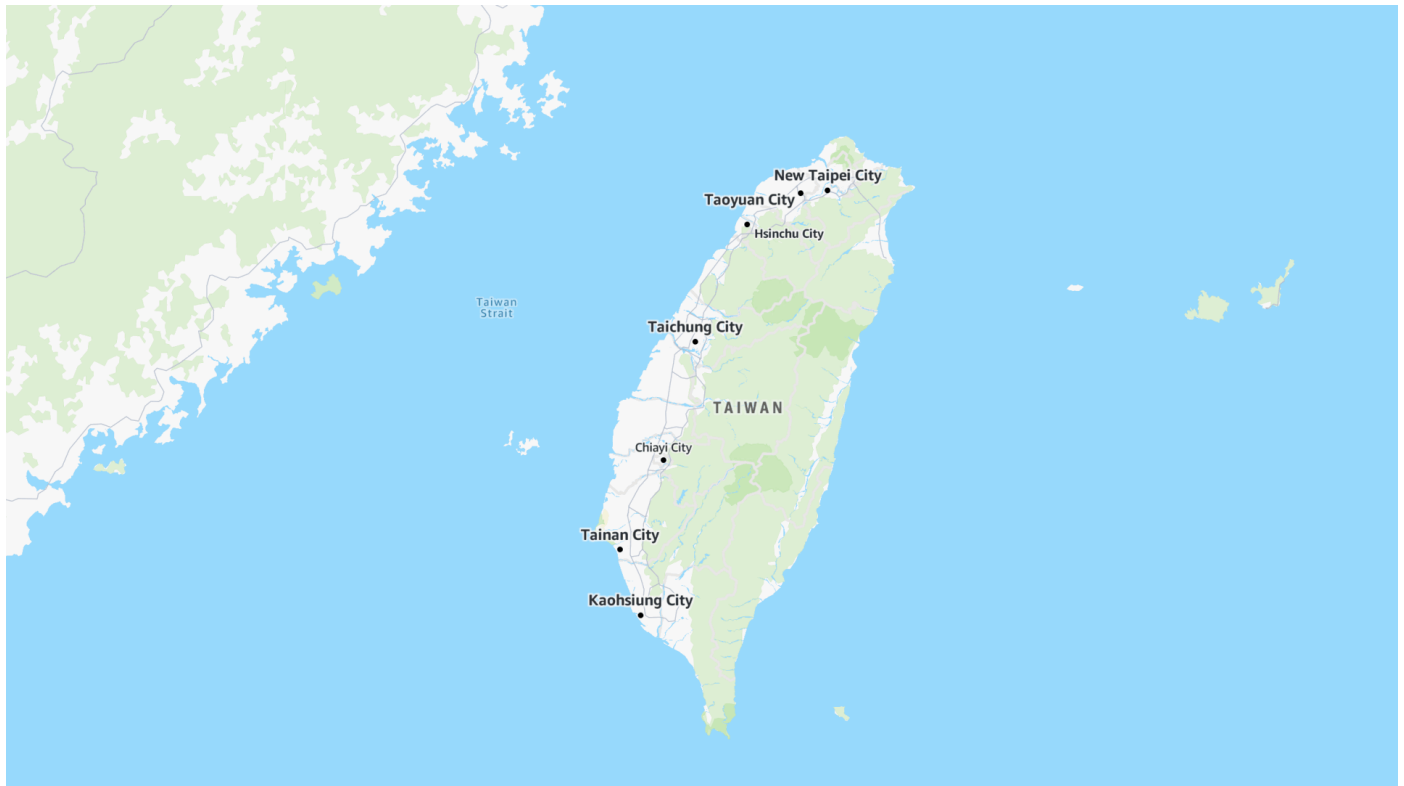


Designed for the world

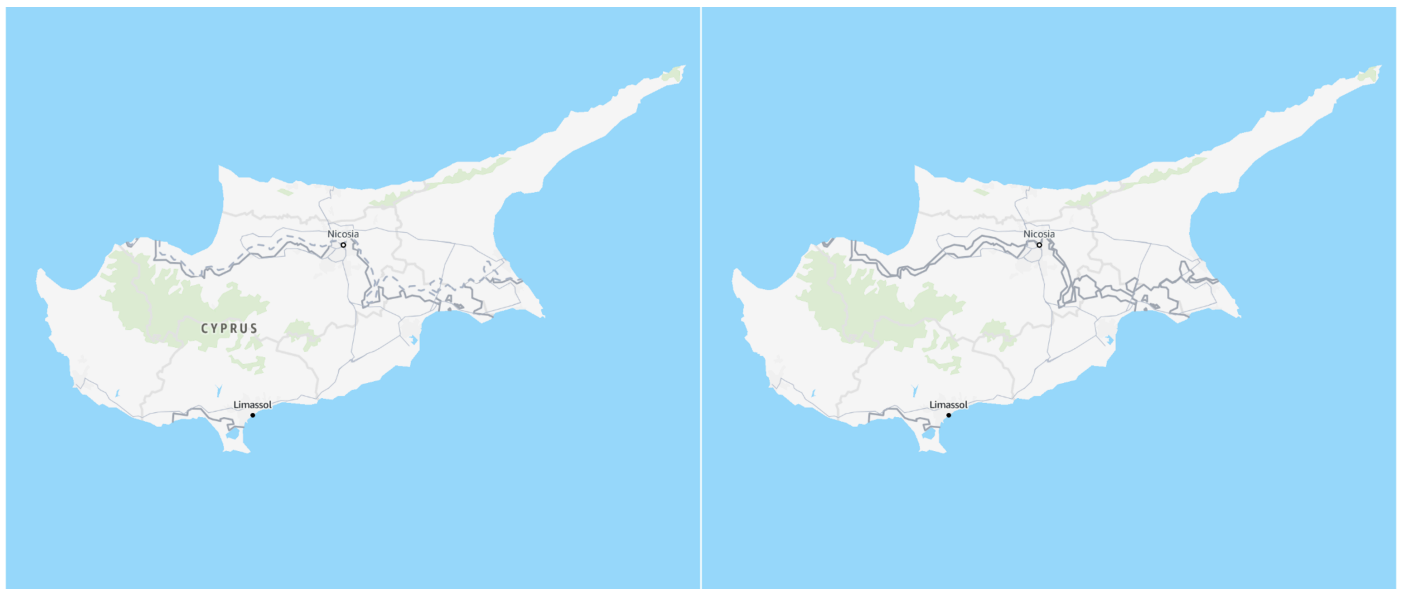
The Standard style supports different political views, ensuring that maps display the correct borders for your users. The style also allows easy language switching for map labels, with dozens of supported languages and writing systems.

To learn more, see [the section called “Localization and internationalization”](#).

Languages



Political view

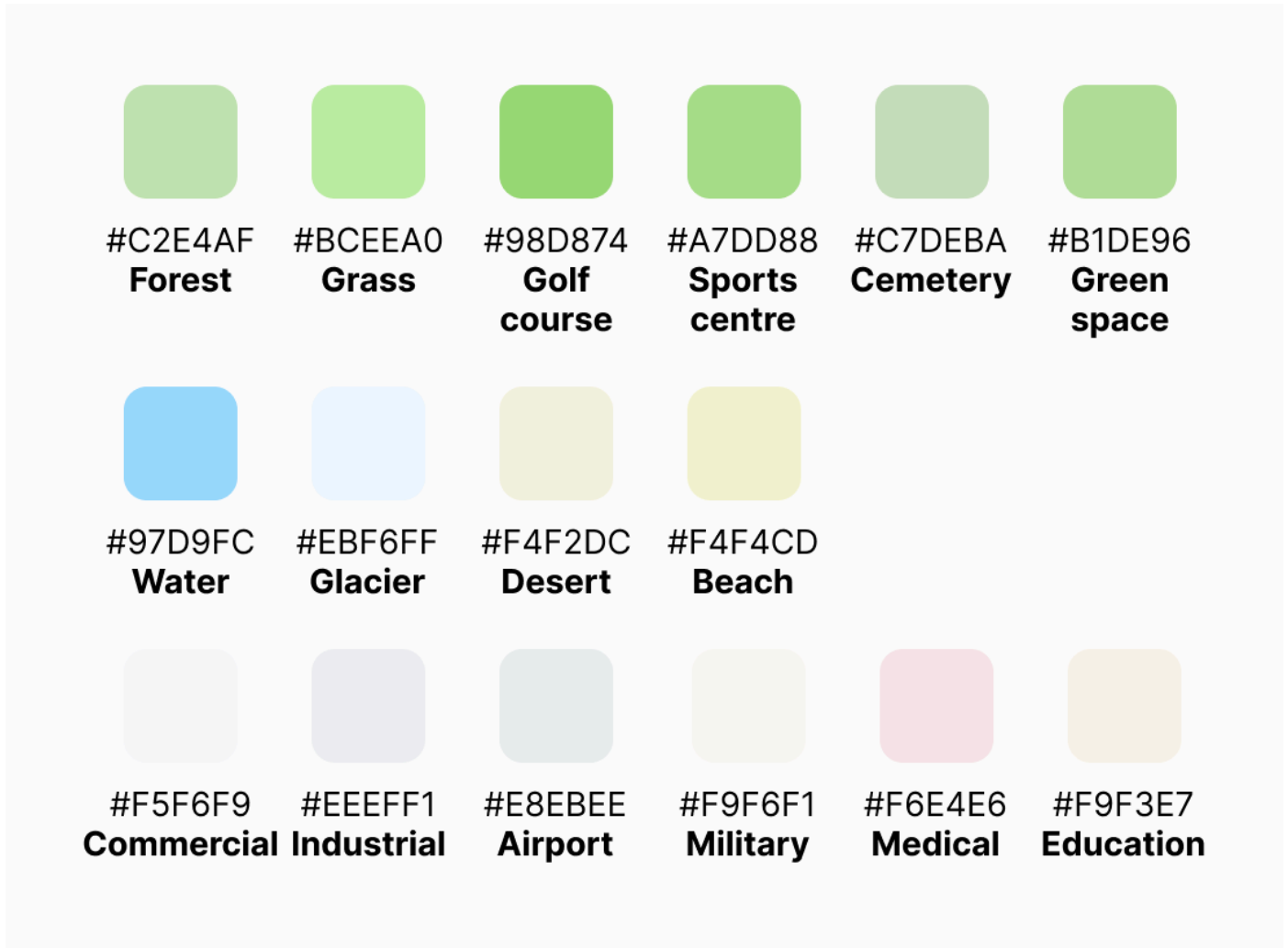


Land use

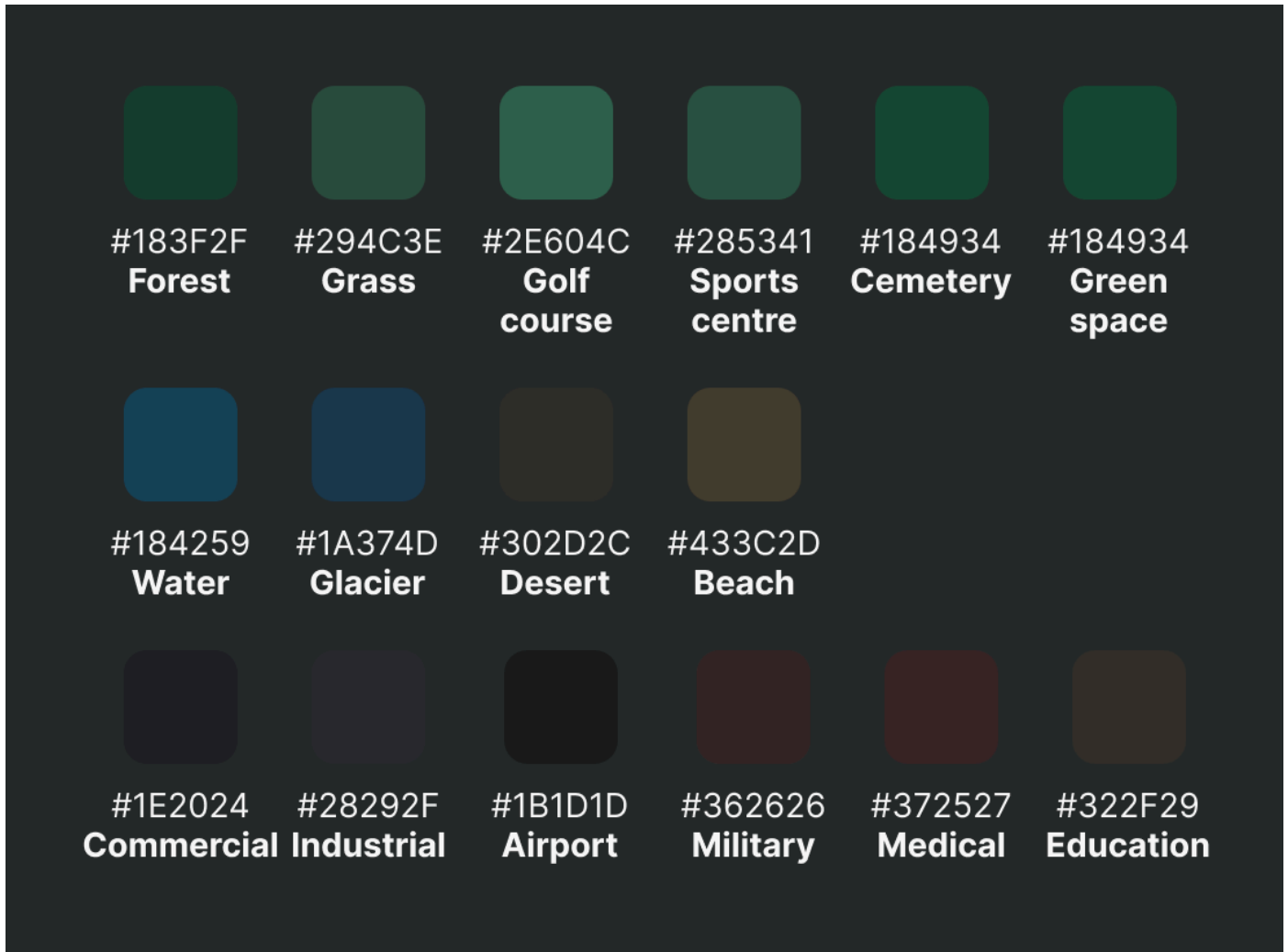
The Standard map style uses vibrant colors to indicate designated land uses. Greens represent forests, grass, golf courses, sports centers, and parks. Relevant colors are used for water bodies,

glaciers, deserts, and beaches. Additionally, land uses such as commercial, industrial, airports, military zones, medical facilities, and educational areas are highlighted with specific vibrant categories.

Light



Dark



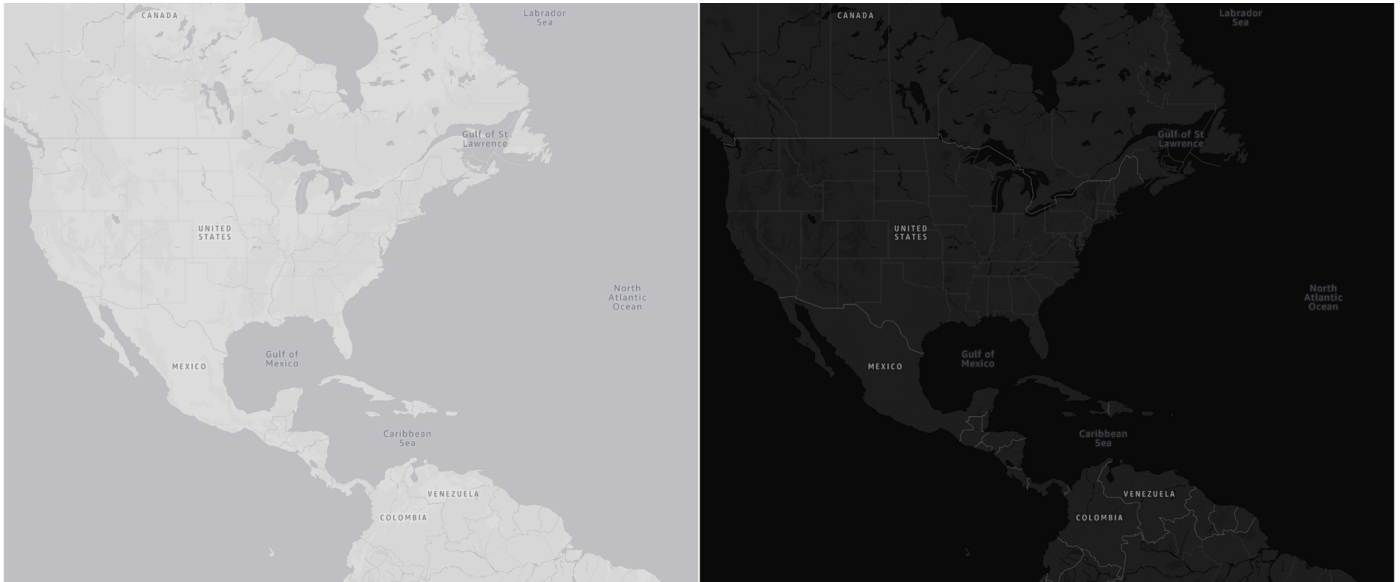
Monochrome map style

The Monochrome style is a minimalist canvas with a constrained color palette, designed for use with data visualization overlays. This style supports both light and dark modes, each of which communicates all the essential information needed for geographic context.

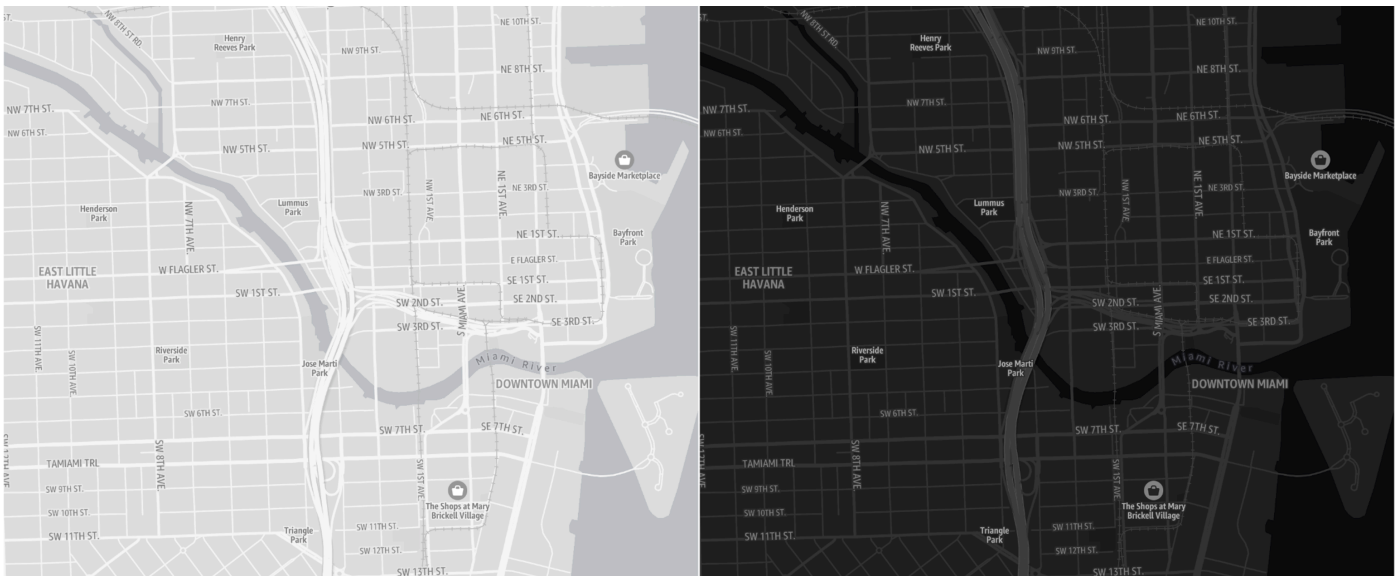
Color schemes

The Monochrome style offers color choices for both dark and light modes.

Continent



Neighborhood

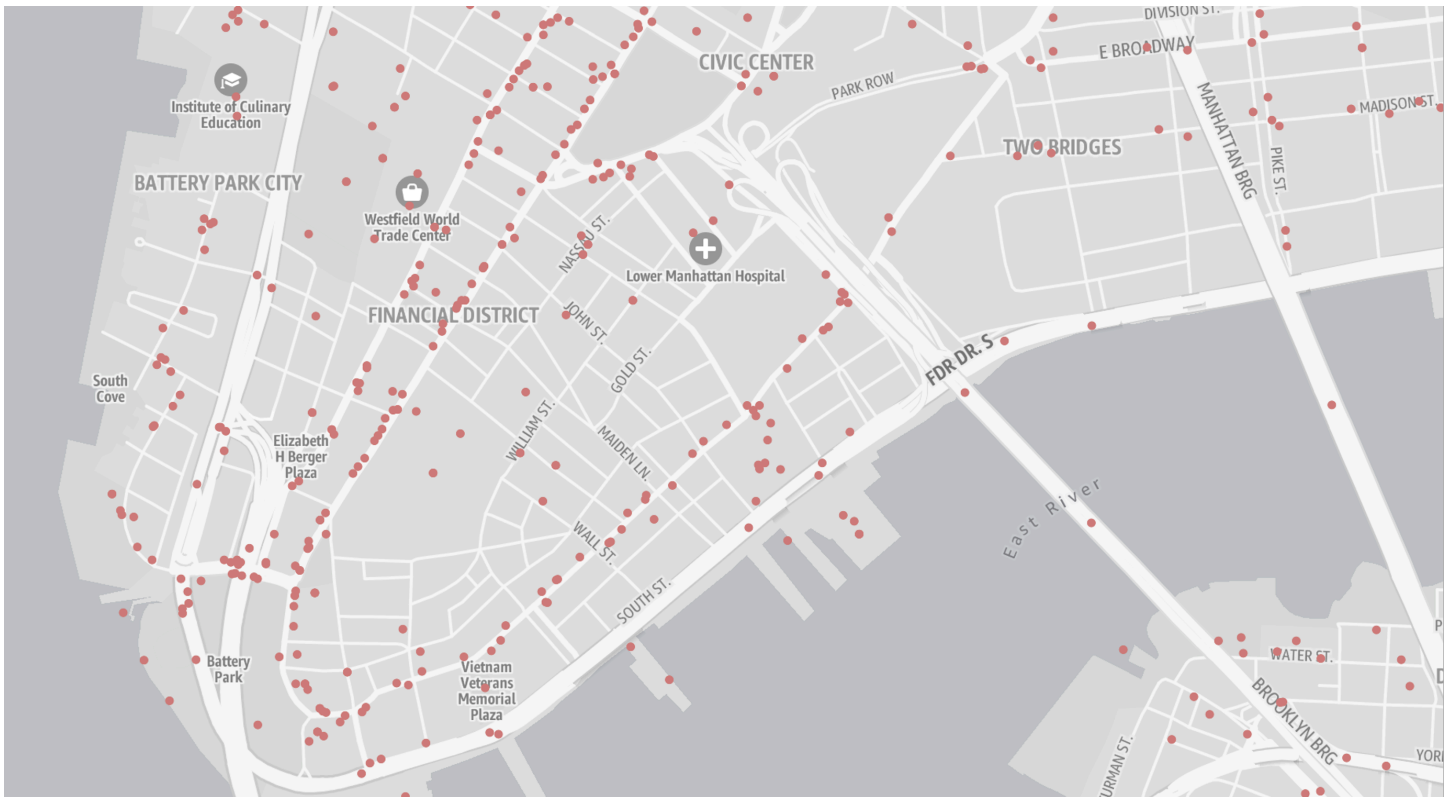


Use cases

The Monochrome style is well-suited for data visualization and minimalistic design needs.

Data visualization

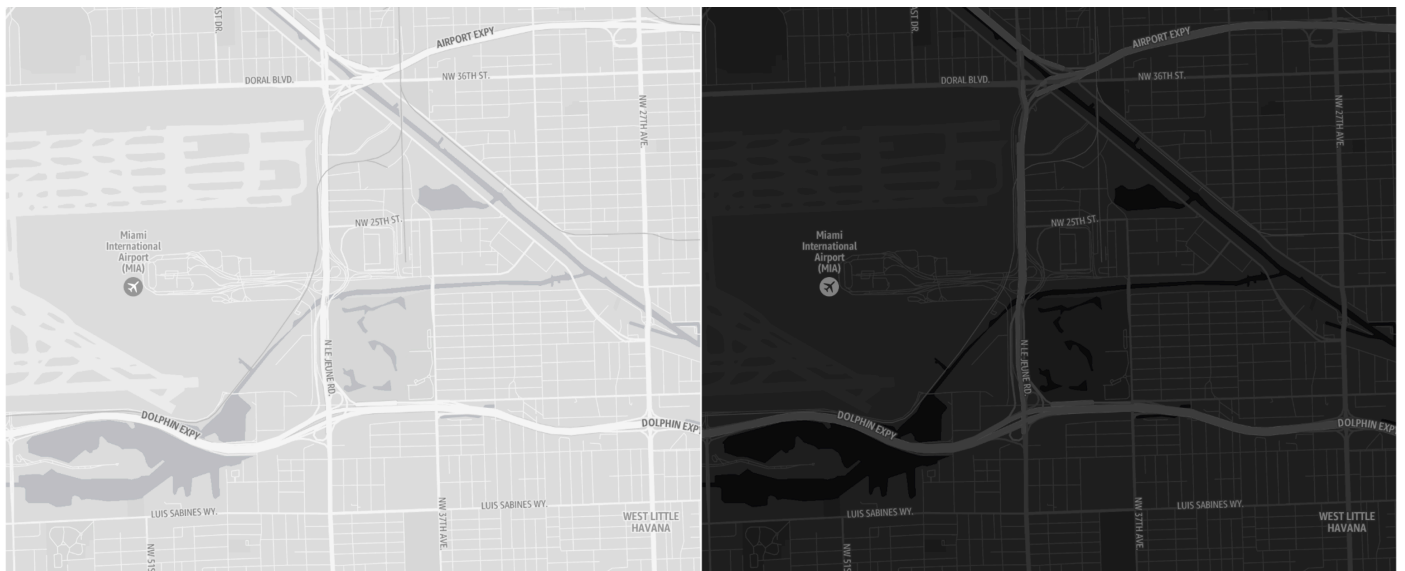
The Monochrome style deliberately uses only shades of gray, allowing you complete freedom of color choice for data overlay layers such as choropleths, heatmaps, or dot maps.



Minimalist design

To maintain a clean and unobtrusive map, the Monochrome styles include a reduced set of points of interest (POIs) for essential features, such as airports, parks, hospitals, and universities.

Airport



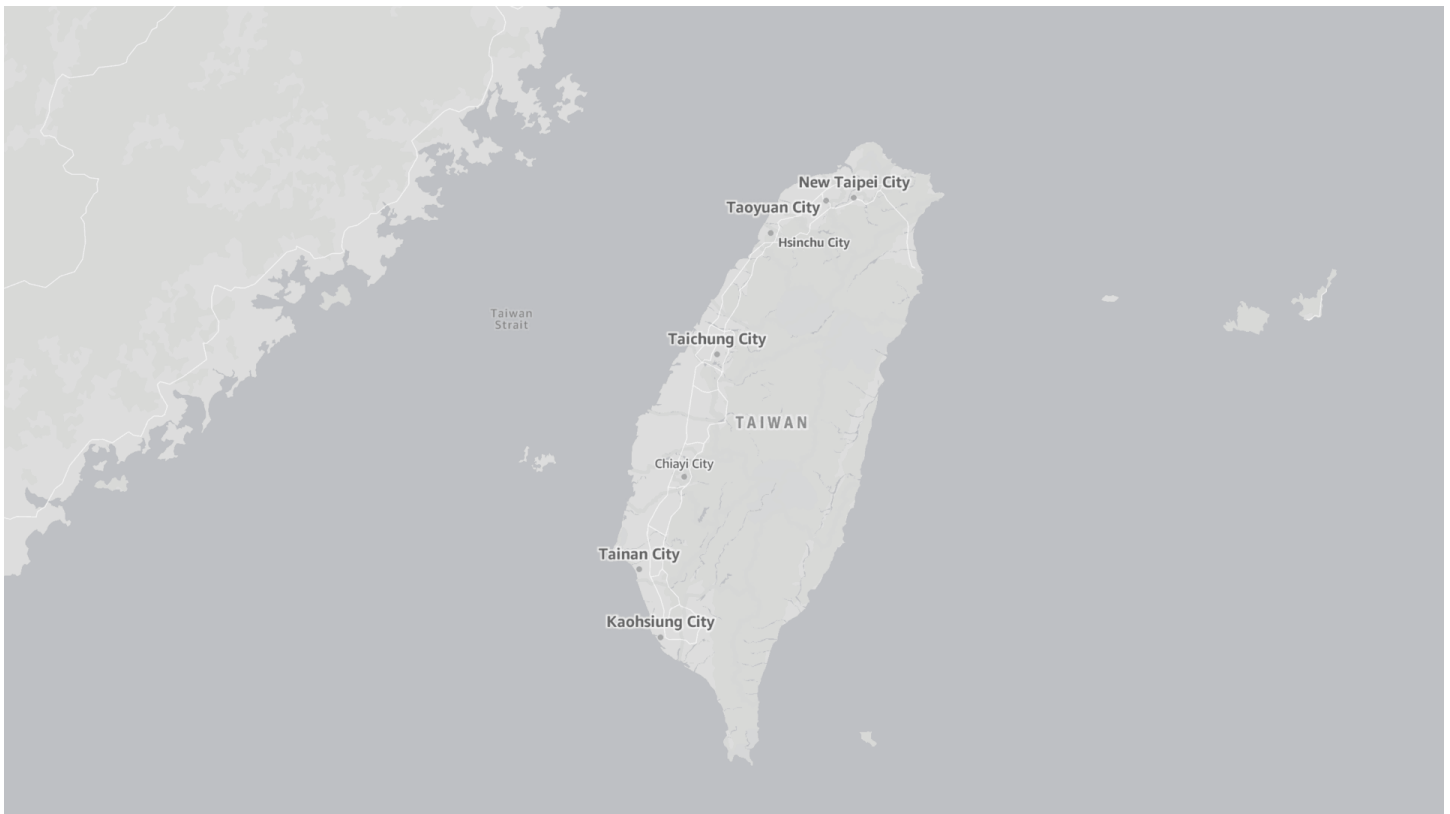
Neighborhood



Although the Monochrome style includes a reduced set of POIs, the underlying tiles still contain the complete set of POI data. This allows you to display POIs that are not visually present in the style.

Designed for the world

The Monochrome style supports different political views, ensuring that maps display the correct borders for your users. The style also allows for easy switching between languages for map labels, with dozens of supported languages and writing systems.



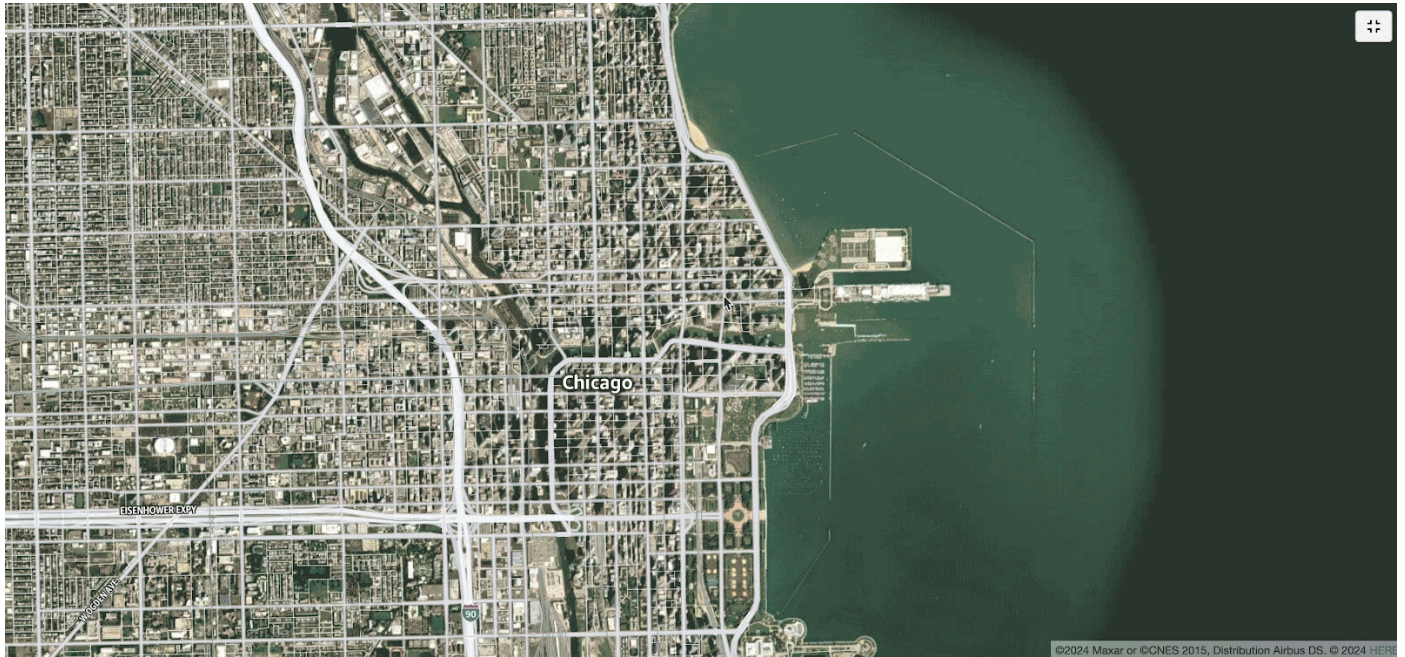
Hybrid map style

The Hybrid map style combines global satellite imagery with the same clear labels and configurable points of interest (POI) categories found in the Standard map style. This combination provides rich geographic detail while ensuring readability and usability for your application.

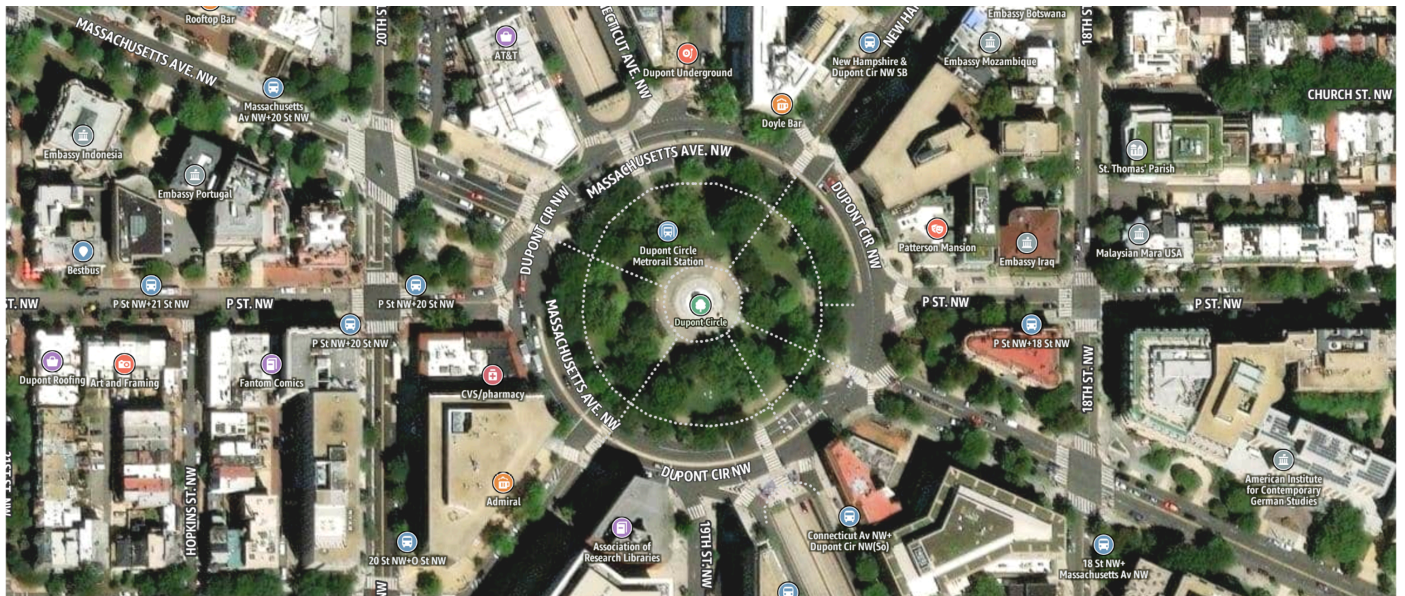
Rich points of interest (POI)

The labels and POIs have been specifically designed for contrast and readability, providing the necessary context for the satellite layer without distracting from the detailed imagery. Light road lines highlight the urban structure when zoomed out and gradually fade as you zoom in, revealing more detailed street-level information.

Zoom



Neighborhood



Zoomed-in



Designed for the world

The Hybrid style supports different political views, ensuring that the map displays the correct borders for your users. This style also allows for easy switching between languages for map labels, with dozens of supported languages and writing systems available to ensure a localized experience.

Map APIs

Maps provide access to both dynamic and static map types for a variety of applications. For more information, See [Maps](#).

- **Dynamic Maps:** Interactive maps that can be customized in real time, allowing users to pan, zoom, and overlay data. For more information, See [the section called “Dynamic maps”](#).
- **Static Maps:** Static images of maps that display specific locations or routes without interactive elements, suitable for applications with limited interactivity. For more information, See [the section called “Static maps”](#).

The following table presents a number of business use cases that are best solved with Maps APIs.

Maps use cases

The following section presents a number of business use cases that are best solved with Maps APIs.

Business need	Useful API	Examples
<p>Display interactive maps</p> <p>Supports map gestures, such as zoom, pan, ease, fly, pitch, rotate, and bearing.</p>	<p>GetTile and GetStyled escriptor with rendering engine (MapLibre)</p>	<p>the section called “Display a map”</p>
<p>Add markers to a map</p> <p>Examples are markers, icon, and more.</p>	<p>GetTile and GetStyled escriptor with rendering engine (MapLibre)</p>	<p>the section called “Add a marker on the map”</p> <p>the section called “Add an icon on the map”</p>
<p>Add user interaction components to a map</p> <p>Examples are showing map in preferred language or geographical view.</p>	<p>GetTile and GetStyled escriptor with rendering engine (MapLibre)</p>	<p>the section called “Add control on the map”</p> <p>the section called “Add a popup to a map”</p>
<p>Visualize real time or pre-recorded data on a map</p> <p>Examples are heat map, KML, GeoJSON features, polygons, rectangles, polylines, circles, markers, and more.</p>	<p>GetTile and GetStyled escriptor with rendering engine (MapLibre)</p>	<p>the section called “Add a line on the map”</p> <p>the section called “Add a polygon on the map”</p>
<p>Display map with localization</p> <p>Examples are showing map in preferred language or geographical view.</p>	<p>GetTile and GetStyled escriptor with rendering engine (MapLibre)</p>	<p>the section called “Set a preferred language for a map”</p> <p>the section called “Set the political view of a map”</p>
<p>Display a static map image</p>	<p>GetStaticMap</p>	<p>the section called “Get a static map of a specific position”</p>

Business need	Useful API	Examples
<p>For example, use map image in application, email, report, or print.</p>		<p>the section called “Get a static map of a specific dimension”</p> <p>the section called “Decide between radius and zoom for a static map”</p> <p>the section called “Add scale for a static map”</p>
<p>Add marker to a map image</p> <p>Examples are markers, proximity circle, icon, and more.</p>	GetStaticMap	<p>the section called “Add a marker to a static map”</p>
<p>Visualize data on a map image</p> <p>Examples are GeoJSON features, polygons, rectangles, polylines, circles, and more.</p>	GetStaticMap	<p>the section called “Add a line to a static map”</p>
<p>Visualize real world use case on a map</p> <p>Examples include routes, proximity circle, and more.</p>	GetStaticMap	<p>the section called “Add a route to a static map”</p>
<p>Visualize Places search and/or geocode result on a map</p> <p>All APIs return geocoordinates, except autocomplete.</p>	GetTile and GetStyled escriptor with rendering engine (MapLibre) with Places API	
<p>Draw a route on a map</p> <p>Supports way point marking.</p>	GetTile and GetStyled escriptor with rendering engine (MapLibre) with Calculate route	

Business need	Useful API	Examples
Visualize matched GPS traces on a map Supports travel modes, such as truck, pedestrian, car, and scooter.	GetStyleDescriptor with rendering engine (MapLibre) with Snap to road	

Dynamic maps

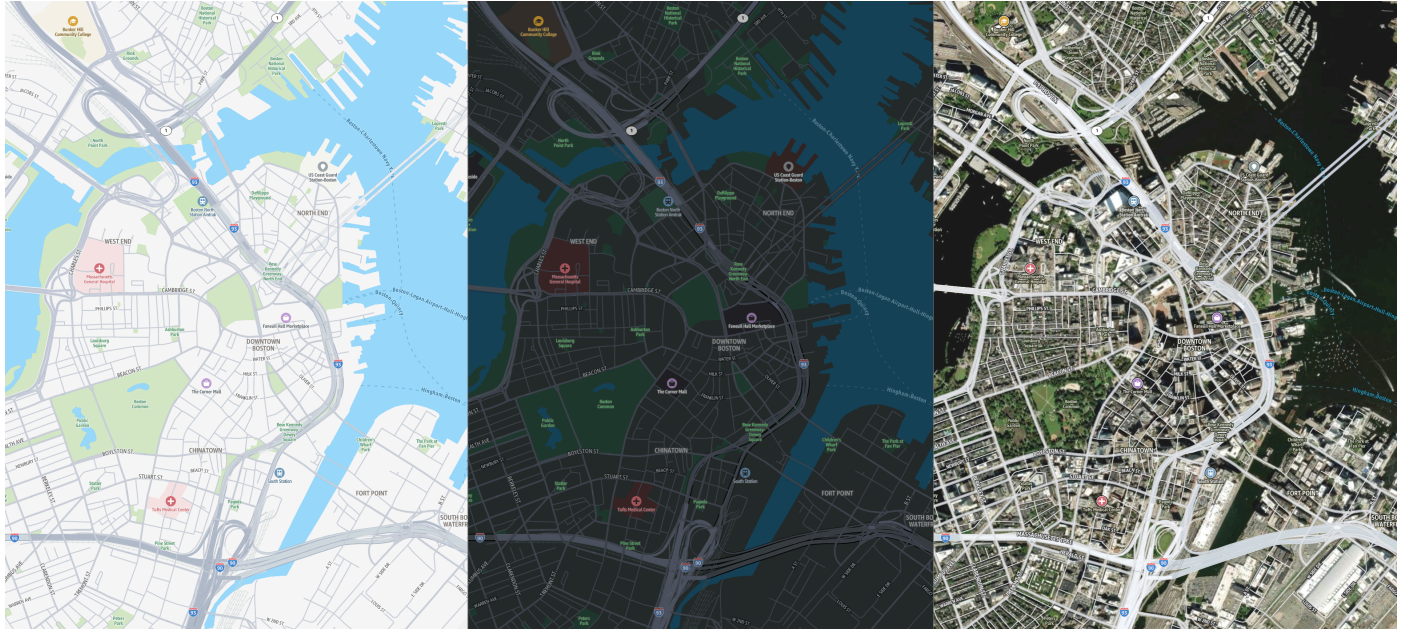
Note

You must use the political view feature to comply with applicable laws, including those related to mapping the country or region where maps, images, and other data you access through Amazon Location Service are made available.

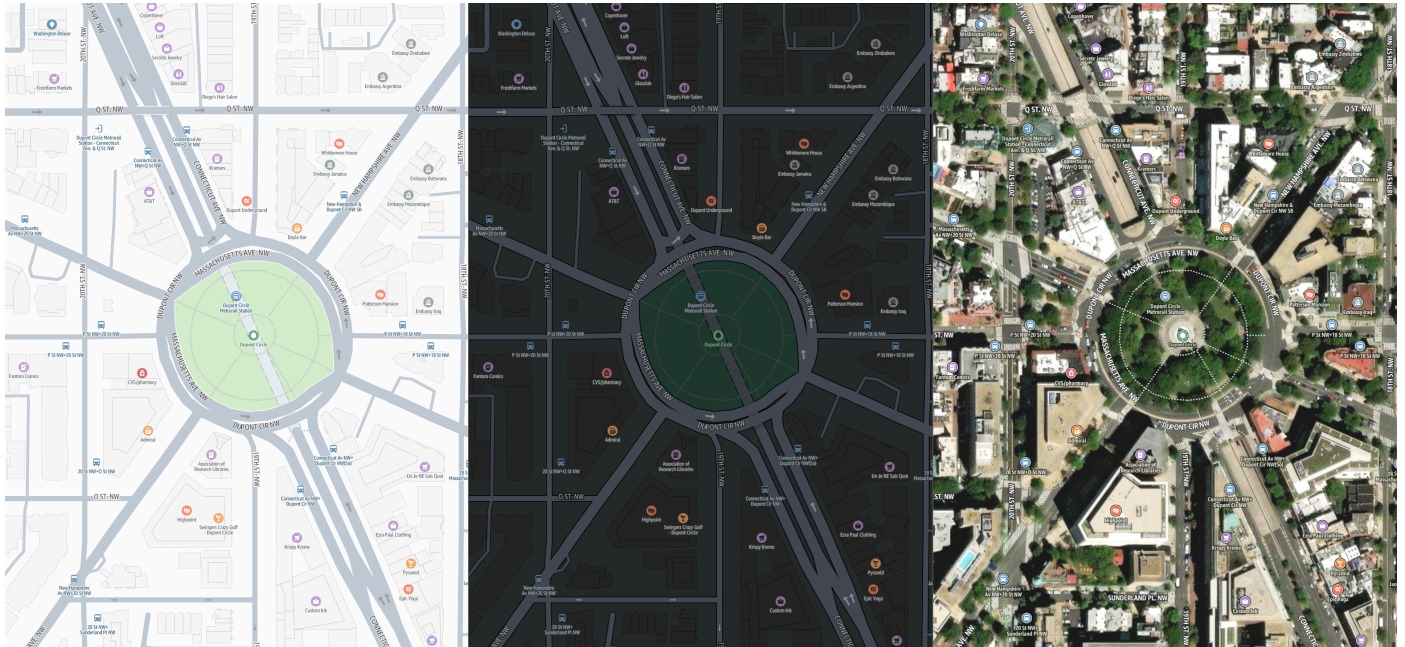
Dynamic maps, also known as interactive maps, are digital maps that support gestures such as zoom, pan, ease, fly, pitch, rotate, and bearing. With Amazon Location Service, you can build map applications that provide responsive, interactive, and immersive experiences for your users. These maps help users visualize and analyze real-time and historical data based on user input, allowing them to pan, zoom, and explore the map in real-time. Maps offered by Amazon Location Service also support multiple languages and political views.

Learn more about [the section called “Localization and internationalization”](#).

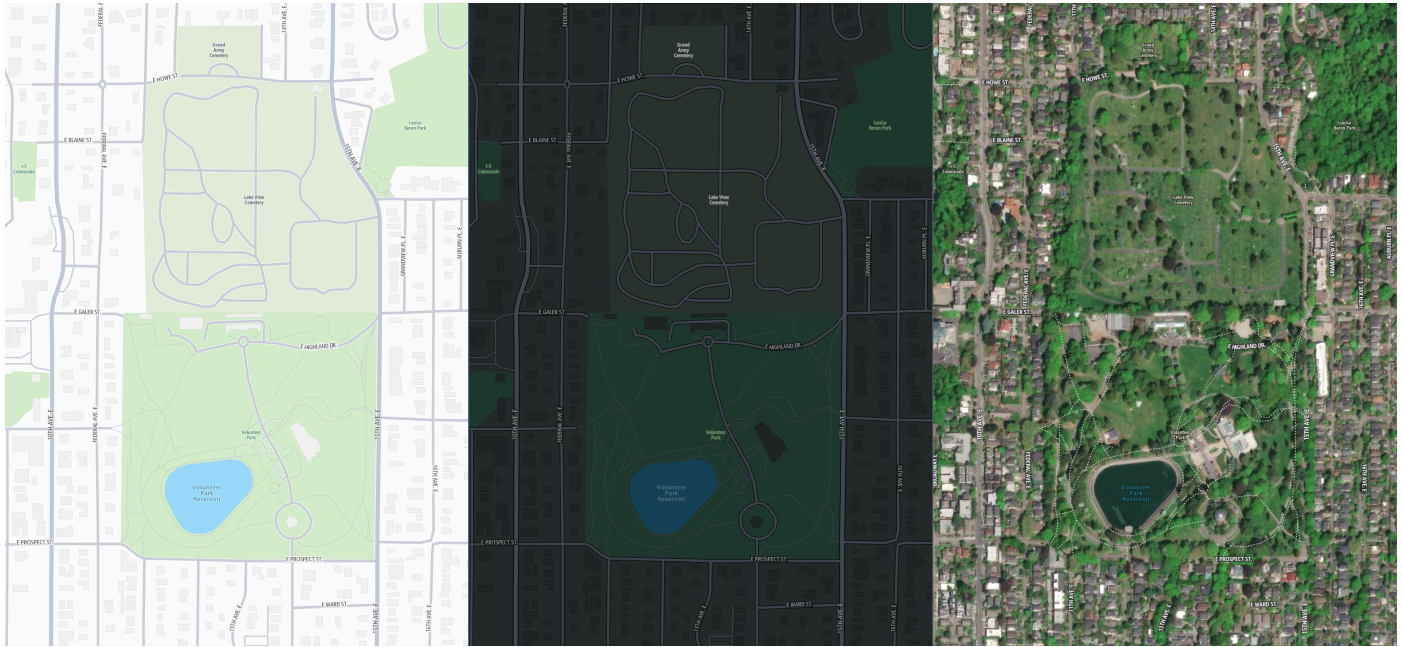
City



Roads



Park



For more information about AWS Map Styles, see [the section called “Map styles”](#).

Common use cases

Analyze and visualize data

Overlay your data onto high-quality maps to uncover transformative spatial patterns and trends. Empower your teams to create customizable, interactive map visualizations with your geographic data. Use maps and data to optimize site selection, plan infrastructure, or analyze market opportunities.

Enhance real estate experiences

Provide prospective buyers with comprehensive location context for real estate listings. Display the property's exact location along with surrounding neighborhood details such as jurisdictional boundaries, local businesses, parks, and schools. Help customers find directions to your open houses and create informative, location-centric real estate experiences.

Build engaging travel experiences

Display dynamic maps showcasing destinations, with detailed street views and key geographic features. Highlight points of interest such as hotels, restaurants, and attractions for tourists and travelers. Plot outdoor amenities like hiking trails to help users plan their ideal itinerary.

Rendering dynamic maps

A map rendering engine is a library responsible for the visual rendering of maps on digital screens. The rendering engine stitches map tiles (vector, hybrid, satellite), map data (points, lines, polygons), or raster data (imagery) together to display an interactive map in web browsers or native apps. Amazon Location Service recommends using the [MapLibre](#) rendering engine, which supports both web and mobile (iOS and Android) platforms. MapLibre also provides a plugin model and supports user interfaces for searching and routing in various languages.

For more information, see [the section called “Create your first Maps and Places application”](#) and [the section called “Dynamic maps”](#).

Requesting map assets

The rendering engine uses a map style, which contains references to map tiles, sprites (icons), and glyphs (fonts). As users interact with the map—loading, panning, or zooming—the rendering engine calls APIs (for tiles, sprites, and glyphs) with the desired input parameters. You can also directly call these APIs based on your application's needs.

Map tiles

Small square tiles containing data that are retrieved from servers and assembled by a rendering engine to create an interactive digital map.

Map style

A collection of rules that define the visual appearance of the map, such as colors and styles. Amazon Location Service follows the [Mapbox GL style specification](#).

Glyph file

A binary file containing encoded Unicode characters, used by the map renderer to display text labels.

Sprite file

A Portable Network Graphics (PNG) image file that contains small raster images, with location descriptions in a JSON file. Used by the map renderer to render icons or textures on the map.

Tiles

Map tiles are pre-rendered, small sections of a larger map, typically displayed as square images. They are used to efficiently display geographic data by loading only the visible portions at different zoom levels. There are three main types of map tiles:

Tile types

Vector map tiles

Vector map tiles store map data as geometric shapes (points, lines, polygons) rather than as images. This enables the creation of high-quality, scalable maps that remain clear at any resolution.

Raster map tiles

Raster map tiles are pre-rendered images representing a specific geographic area. Unlike vector tiles, raster tiles are simpler but lack flexibility for restyling.

Hybrid map tiles

Hybrid map tiles combine both vector and raster data. They use vector data for core map elements, such as roads, while using raster imagery for more complex elements like detailed satellite or aerial photography.

Vector tile layers

The following are the 10 layers of vector map tiles:

- **Boundaries:** Defines administrative and geographic boundaries, including country, state, and city borders.
- **Buildings and addresses:** Represents building shapes and detailed address points.
- **Earth:** Shows global terrain and surface coverage for natural features like deserts, mountains, and forests.
- **Land use:** Displays categorized areas such as parks, farmland, and urban zones.
- **Places:** Identifies important locations like cities, towns, and notable landmarks.
- **Points of interest (POIs):** Highlights attractions, businesses, and other key locations.
- **Roads:** Represents the network of streets, highways, and pathways.
- **Road labels:** Provides text labels for road names and route numbers.

- **Transit:** Depicts public transport lines such as buses, trains, and subways.
- **Water:** Displays bodies of water, including lakes, rivers, and oceans.

Use cases

- Fetching map tiles for rendering different sections of a map at various zoom levels.
- Optimizing map tile requests based on user interaction, such as panning and zooming.
- Accessing vector or raster tiles for detailed rendering purposes.

Understand the request

The request requires the following parameters: `Tileset`, `X`, `Y`, and `Z` to identify the specific tile to be fetched. The `Key` parameter can be optionally included for authorization.

- **Tileset:** Specifies the desired tileset for fetching the tile.
- **X:** The X-axis value for the map tile.
- **Y:** The Y-axis value for the map tile.
- **Z:** The zoom value, defining the zoom level for the tile.
- **Key:** Optionally included for authorization purposes.

Understand the response

The response includes headers such as `CacheControl`, `ContentType`, and `ETag`, and contains the map tile data as a binary blob in MVT format. These headers manage cache control, provide content format details, and version control for tiles.

- **CacheControl:** Controls client-side caching for the map tile.
- **ContentType:** Specifies the format of the tile data.
- **ETag:** Provides a version identifier for the tile.
- **Blob:** Contains the vector tile data in MVT format.

Style dynamic maps

Amazon Location Service provides two options for styling your dynamic maps: using predesigned AWS Map Styles or customizing the map style using style descriptors.

Use predefined AWS map styles

AWS map styles are predefined styles that adhere to industry standards to deliver a sophisticated, professional aesthetic. By leveraging these styles in Amazon Location Service, you can reduce time-to-market and eliminate the need for dedicated cartographers to create map styles from scratch.

For more information, see [the section called “Map styles”](#).

To learn more about predefined map styles, see:

- [the section called “Standard map style”](#)
- [the section called “Monochrome map style”](#)
- [the section called “Hybrid map style”](#)
- [the section called “Satellite map style”](#)

Benefits of using AWS map styles

- **Time and resource efficiency:** AWS Map Styles allow you to bypass the time-consuming and resource-intensive process of designing map styles from scratch. This allows you to focus on core functionalities while providing visually appealing maps.
- **Professional and consistent aesthetics:** Skilled cartographers have meticulously crafted AWS Map Styles, following industry best practices. Every detail, from color palettes to label placements, has been optimized for clarity and legibility.
- **Seamless integration:** AWS Map Styles integrate seamlessly with your application's design language, providing a polished and consistent mapping experience for your end-users.

Get started with AWS map styles

- **Check the AWS map styles offering:** In the Amazon Location Service console, navigate to the **Map** section to explore the available styles.
- **Choose the style that matches your needs:** Select the style that best aligns with your application's design and user experience requirements.
- **Integrate the style:** Follow the provided documentation to integrate the chosen style into your application using Amazon Location Service APIs or SDKs.

Learn more about [the section called “Display a map”](#).

Use cases

- Customizing map styles based on color schemes like "Light" or "Dark".
- Displaying maps according to specific political views or geographic boundaries.
- Optimizing map styles for different use cases, such as logistics or default views.

Understand the request

The request supports parameters like `ColorScheme`, `Key`, and `PoliticalView` to define the map's style and presentation. The `Style` parameter is required to specify the desired map style.

- **ColorScheme:** Sets the map's color tone, such as "Light" or "Dark".
- **PoliticalView:** Specifies the political view for map visualization.
- **Style:** Defines the style of the map, like "Standard" or "Monochrome".

Understand the response

The response provides headers like `CacheControl`, `ContentType`, and `ETag`, and contains the style descriptor as a JSON blob. The headers give caching information, content format details, and versioning for style changes.

- **CacheControl:** Controls caching configurations for the style descriptor.
- **ContentType:** Indicates the response format as JSON.
- **ETag:** Provides a version identifier for the style descriptor.
- **Blob:** Contains the body of the style descriptor in JSON format.

Customize style descriptors

To customize map styles, you must understand the structure of the style descriptor, which is usually a JSON object defining the visual representation of map elements. The style descriptor comprises several layers, each controlling the style for a specific type of map element, such as roads, parks, buildings, or labels.

- **Use a predefined style descriptor as a base:** You can either start with a predefined style descriptor or create one from scratch using map style editors such as [Maputnik](#).

- **Understand the structure:** The style descriptor is a hierarchical JSON object that contains layers, each representing a different map element. Each layer has properties that control the visual appearance of that element, such as color, opacity, and line width.
- **Modify styles for layers:** Depending on the map style editor you're using, you can change existing layers or add new ones to customize the style. For example, you can adjust the color of roads, modify the font size of labels, or add custom icons for specific locations.
- **Define styles for different zoom levels:** Map style editors allow you to define different styles for different zoom levels, which is useful for controlling the level of detail and visibility based on user zoom interactions.
- **Test and iterate:** After modifying or creating the style descriptor, test the customized style on a map to ensure it displays as intended. Iterate and adjust until you achieve the desired visual style.

Style iconography with sprites

A sprite is a Portable Network Graphic (PNG) image file that contains small raster images such as icons, markers, and other elements rendered on a map. Sprites can be customized based on parameters like style, color scheme, and variant. Amazon Location Service provides a sprite sheet through the `GetSprites` API. You can also use custom icons by either loading your own icon set (see [the section called "Add an icon on the map"](#)) or customizing the style descriptor to load your custom sprites.

Use cases

- Rendering custom map elements using sprite sheets for specific styles and color schemes.
- Fetching sprites for various map styles such as Standard, Monochrome, or Hybrid.
- Customizing iconography on the map by modifying sprites.

Understand the request

The request requires URI parameters such as `ColorScheme`, `FileName`, and `Style`. These parameters allow for the customization of the sprite sheet based on the map's color scheme, style, and the specific sprite file required.

- **ColorScheme:** Defines the color scheme for the sprites, such as "Light" or "Dark".
- **FileName:** The name of the sprite file to retrieve, which could be a PNG or JSON file.

- **Style:** Specifies the map style, such as "Standard" or "Monochrome".

Understand the response

The response contains headers such as `CacheControl`, `ContentType`, and `ETag`, and returns the sprite data as either a binary blob or a JSON file. These headers provide caching information, the content type of the response, and version control for the sprite data.

- **CacheControl:** Caching configurations for the sprite file.
- **ContentType:** The format of the response, indicating whether it contains PNG or JSON data.
- **ETag:** Identifier for the sprite's version, used for cache validation.
- **Blob:** Contains the body of the sprite sheet or the JSON offset file.

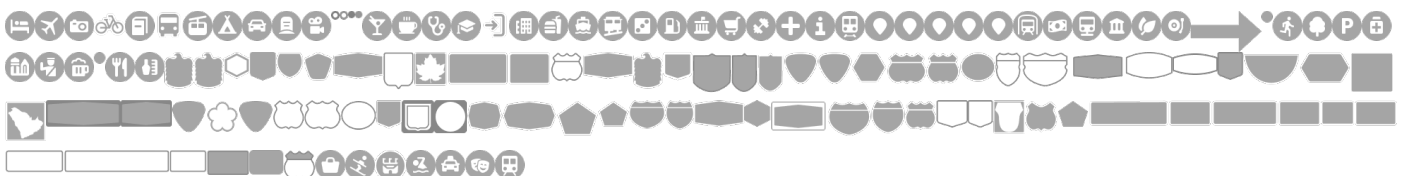
Standard Light



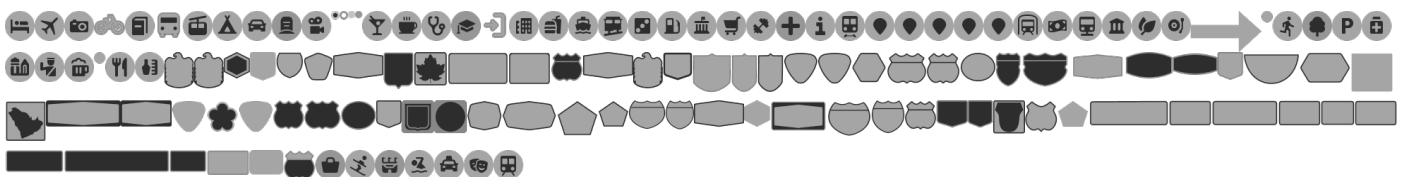
Standard Dark



Monochrome Light



Monochrome Dark



- Amazon Ember Condensed RC LtItalic
- Amazon Ember Condensed RC Regular
- Amazon Ember Condensed RC Regular Italic
- Amazon Ember Condensed RC Regular,Noto Sans Regular
- Amazon Ember Condensed RC Regular,Noto Sans Regular,Noto Sans Arabic Condensed Regular
- Amazon Ember Condensed RC RgItalic
- Amazon Ember Condensed RC ThItalic
- Amazon Ember Condensed RC Thin
- Amazon Ember Condensed RC Thin Italic
- Amazon Ember Heavy
- Amazon Ember Heavy Italic
- Amazon Ember Light
- Amazon Ember Light Italic
- Amazon Ember Medium
- Amazon Ember Medium Italic
- Amazon Ember Medium,Noto Sans Medium
- Amazon Ember Medium,Noto Sans Medium,Noto Sans Arabic Medium
- Amazon Ember Regular
- Amazon Ember Regular Italic
- Amazon Ember Regular Italic,Noto Sans Italic
- Amazon Ember Regular Italic,Noto Sans Italic,Noto Sans Arabic Regular
- Amazon Ember Regular,Noto Sans Regular
- Amazon Ember Regular,Noto Sans Regular,Noto Sans Arabic Regular
- Amazon Ember Thin
- Amazon Ember Thin Italic
- AmazonEmberCdRC_Bd
- AmazonEmberCdRC_BdIt
- AmazonEmberCdRC_Lt

- AmazonEmberCdRC_LtIt
- AmazonEmberCdRC_Rg
- AmazonEmberCdRC_RgIt
- AmazonEmberCdRC_Th
- AmazonEmberCdRC_ThIt
- AmazonEmber_Bd
- AmazonEmber_BdIt
- AmazonEmber_He
- AmazonEmber_Helt
- AmazonEmber_Lt
- AmazonEmber_LtIt
- AmazonEmber_Md
- AmazonEmber_MdIt
- AmazonEmber_Rg
- AmazonEmber_RgIt
- AmazonEmber_Th
- AmazonEmber_ThIt
- Noto Sans Black
- Noto Sans Black Italic
- Noto Sans Bold
- Noto Sans Bold Italic
- Noto Sans Extra Bold
- Noto Sans Extra Bold Italic
- Noto Sans Extra Light
- Noto Sans Extra Light Italic
- Noto Sans Italic
- Noto Sans Light
- Noto Sans Light Italic
- Noto Sans Medium

- Noto Sans Medium Italic
- Noto Sans Regular
- Noto Sans Semi Bold
- Noto Sans Semi Bold Italic
- Noto Sans Thin
- Noto Sans Thin Italic
- NotoSans-Bold
- NotoSans-Italic
- NotoSans-Medium
- NotoSans-Regular
- Open Sans Regular, Arial Unicode MS Regular

Understand the request

The request accepts two required URI parameters, `FontStack` and `FontUnicodeRange`, which determine the font and the Unicode range for glyphs. The `FontStack` parameter specifies which font to use, while the `FontUnicodeRange` defines the character range to fetch. The request does not include a body, focusing only on the URI parameters for its functionality.

- **FontStack:** Specifies the name of the font stack to retrieve. Example: "Amazon Ember Bold, Noto Sans Bold".
- **FontUnicodeRange:** A Unicode range of characters to download glyphs for. The range must be multiples of 256. Example: "0-255".

Understand the response

The response returns glyph data as a binary blob, along with HTTP headers for caching, content type, ETag, and pricing information. The glyph data is returned as a binary blob to be rendered on maps, and the headers provide additional metadata for handling the response effectively.

- **CacheControl:** Instructs the client on caching configurations for the response.
- **ContentType:** Specifies the format of the response body, indicating the type of glyph data returned.
- **ETag:** An identifier for the glyph's version, used for cache validation.

- **PricingBucket:** Indicates the pricing tier associated with the request.
- **Blob:** The glyph data returned as a binary blob, used to render map text.

Static maps

Note

Static maps only support Satellite style. For more information, see [the section called “Map styles”](#).

Static maps offer a pre-rendered representation of geographic data with the option to overlay markers (or pins), routes, and polygon areas, as needed for your application. The Static Map lets you generate static (non-interactive) map images based on customizable parameters and data inputs. By customizing overlays, shapes, or applying custom styles, Static Map enables you to create map visualizations that meet specific needs, enhancing the end-user experience and effectively communicating geographical information. The server customizes the requested map images and delivers them to the client as JPEG files. You can programmatically request and generate map images tailored to your specific requirements.

The *GetStaticMap* API generates a static image of a map based on specified parameters like center coordinates, bounding boxes, or overlays. The API allows customization of map features and style, enabling use in web or mobile applications without interactive map functionality.

Common use cases

- **Embedded maps in web or mobile application:** Static map images can be efficiently embedded in websites or mobile applications to provide visualizations of locations, routes, or points of interest with non-interactive maps, reducing load times and data usage. Examples include search engines (such as Yahoo) showing map images with search results for POIs.
- **Location details in e-mails:** Static map images can be used to share location information via email to help your end users understand the context of the email. For example, food delivery or ride-sharing apps use static map images to display pickup/drop-off locations, routes, or surrounding areas in post-trip or delivery emails containing bill and summary.
- **Marketing materials and printed documents:** Customized static map images can be incorporated into brochures, flyers, or other printed materials, providing visually appealing representations of geographical information relevant to the content.

Understand the request

The request includes optional URI parameters, like `BoundedPositions`, `BoundingBox`, and `Center`, among others, to define the visible area and overlays of the map. The parameters `Height` and `Width` are required for defining the image size. To learn more, see [the section called “Customize static maps”](#) and [the section called “Overlay on the static map”](#).

- `BoundedPositions`: Coordinates to encompass in the image.
- `BoundingBox`: Coordinates defining the south-west and north-east edges of the map.
- `Height`: Specifies the height of the image.
- `Width`: Specifies the width of the image.
- `GeoJsonOverlay`: A valid GeoJSON object for adding overlays.

Understand the response

The response contains headers like `CacheControl`, `ContentType`, and `ETag`, and returns the static map as a binary blob in either JPEG or PNG format. The headers provide metadata like cache control, content type, and version for static images.

- `CacheControl`: Specifies caching configurations for the map image.
- `ContentType`: Indicates the format of the map image (JPEG or PNG).
- `ETag`: An identifier for the version of the static map image.
- `Blob`: Represents the map image in either JPEG or PNG format.

Customize static maps

Note

Static maps only support the Satellite style. For more information, see [the section called “Map styles”](#).

This section provides an overview of how to customize static maps generated using Amazon Location Service. It covers various features, such as adjusting the map's position, size, language, scale, overlays, and attribution, enabling you to tailor the map to your specific requirements.

Position

The position allows you to define the center and boundaries of the map. You can control the map's focus by setting the center coordinates, a bounding box, or using a zoom level to determine how much area to display. To learn how it works, see [the section called “Get a static map of a specific position”](#).

- **Center:** Defines the center point of the map using longitude and latitude coordinates.
- **Radius:** Specifies the radius (distance from the center) that will be displayed on the static map.
- **Bounding Box:** Defines a rectangular area of the map, set by providing the coordinates of the top-left and bottom-right corners.
- **Zoom:** Controls the zoom level of the map. Higher zoom levels show more detail in a smaller area, while lower zoom levels show less detail over a larger area.

Dimension and quality

You can customize the size and visual quality of the static map by defining its dimensions (height and width) and adding padding for better presentation of markers and other elements. To learn how it works, see [the section called “Get a static map of a specific dimension”](#).

- **Height and Width:** Specifies the size of the static map image by defining its height and width in pixels.
- **Padding:** Adds extra space around the edges of the map, allowing for better visualization when placing markers, lines, or shapes.

Scale

The scale provides control over the scale of the map and defines the units (kilometers, miles) to measure distances. This is useful for accurately representing the map's size and distance relationships. To learn how it works, see [the section called “Add scale for a static map”](#).

- **Scale Unit:** Defines the units for the map's scale bar (for example, kilometers or miles), allowing users to accurately gauge distances on the map.

Overlay

You can add markers, lines to show routes, polygons to show areas, and more. To learn how it works, see [the section called “Add a marker to a static map”](#), [the section called “Add a line to a static map”](#), or [the section called “Add a route to a static map”](#).

Overlay on the static map

This section explains how to overlay additional information onto static maps using Amazon Location Service. You can customize your static maps by adding various geographical features, such as points, lines, and polygons, to enhance the map's visual representation. Amazon Location Service supports multiple formats, including GeoJSON and a compact overlay format, to provide flexible and efficient ways of adding overlays.

With GeoJSON

GeoJSON is a versatile format that allows you to overlay custom data on static maps. By defining geographical features such as points, lines, and polygons, you can enhance the visual representation of your maps, providing valuable context for users. GeoJSON is widely supported and offers flexibility when it comes to styling and customizing map overlays, making it an ideal format for displaying regions, plotting routes, or showing spatial relationships.

With Amazon Location Service, you can leverage GeoJSON to add dynamic, location-based features directly onto your static maps. This enables you to create highly customizable overlays that can be tailored to meet your specific business needs. GeoJSON supports several geometry types, including `Point`, `LineString`, `Polygon`, and `MultiPolygon`, allowing you to display a wide range of features, from markers and routes to complex area representations.

Colors

When styling GeoJSON features, you have flexibility in defining colors. You can specify colors using different formats, such as hexadecimal values (like `#ff0000` for red) or with alpha transparency (like `#ff000080` for semi-transparent red). This ensures your overlays can be visually consistent with the map style. If no color is specified, the default color for the selected map style will be applied.

Drawing order

Custom overlays are drawn in a specific order to maintain clarity and avoid visual clutter. In Amazon Location Service, overlay features like polygons, lines, and points will appear above the base map, but below map labels. The drawing order prioritizes polygons first, followed by lines, and then points or markers.

Measurement units

For properties like `width` and `outline-width`, you can use different measurement units to specify size, including pixels (px), meters (m), kilometers (km), miles (mi), and percentages (%). The percentage unit adjusts the property relative to a default value, providing more flexibility in styling your overlays.

Geometry types

Amazon Location Service supports multiple GeoJSON geometry types, such as `Point`, `LineString`, `Polygon`, and `MultiPolygon`. Each geometry type can be styled and adjusted using the `properties` object in GeoJSON, allowing for extensive customization of markers, routes, and areas on your map.

With compact overlay

Note

Compact overlay supports the following geometry types: `point`, `line`, and `polygon`. It doesn't support `multiPoint`, `multiLine`, or `multiPolygon`.

The compact overlay option allows you to efficiently display multiple geometries on a static map by using a single query parameter. This streamlined approach simplifies the request format and reduces the size of the request, making it easier to transmit overlay data. Customers can input various geometry types and their corresponding style properties in one query parameter, and Amazon Location Service will handle the heavy lifting by parsing and rendering the overlay as specified.

While using the compact overlay format, keep in mind that there are limits on the size of the request URL. Although Amazon Location Service optimizes the query, ensure that your request stays within reasonable limits, especially when dealing with multiple geometries and their associated properties.

Format

The compact overlay format is structured as follows:

```
geometry_type:geometry;property_1=value_1;property_2=value_2|  
geometry_type:geometry;property_1=value_1...
```

Each geometry type is defined along with its style properties. Multiple geometries are separated by a pipe operator (`|`), and properties for each geometry are separated using a semicolon.

Supported geometry types

Amazon Location Service supports several geometry types, including `Point`, `MultiPoint`, `LineString`, `Polygon`, and `MultiPolygon`. These geometry types can be combined and styled within the same query parameter using the compact overlay format.

Styling properties

Each geometry can be customized using various style properties, such as color, outline color, size, and more. These properties allow you to control the appearance of each geometry on the map, ensuring that the overlay aligns with your business requirements.

How to

This section contains a variety of how to guides and examples for how to use Maps APIs.

Examples

- [How to use dynamic maps](#)
- [How to use static maps](#)

How to use dynamic maps

These how-to topics offer a comprehensive walkthrough teaching you how to enhance your mapping applications using the Amazon Location Service. From displaying interactive maps to adding markers, lines, and polygons, these tutorials demonstrate how to utilize essential features like setting map controls, adding custom icons, and handling real-time data. Furthermore, the topics also cover localization and internationalization aspects, such as setting preferred languages, adjusting political views, and ensuring compliance with regional laws by customizing map content based on the user's location or perspective.

Each how-to is designed to be accessible, with step-by-step instructions for implementing these features in web applications using MapLibre GL JS. Whether your goal is to build a dynamic map experience with custom icons and popups or to tailor map views for specific political perspectives and languages, these examples will help you achieve your objectives while ensuring a rich, localized mapping experience for users across different regions. These tutorials ensure that you can fully

leverage the capabilities of Amazon Location Service, from basic mapping functions to complex geopolitical and localization needs.

Topics

- [How to display a map](#)
- [How to add control on the map](#)
- [How to add a marker on the map](#)
- [How to add an icon on the map](#)
- [How to add a line on the map](#)
- [How to add a polygon on the map](#)
- [How to add a popup to a map](#)
- [How to set a preferred language for a map](#)
- [How to set the political view of a map](#)

How to display a map

Amazon Location Service allows you to display both interactive and non-interactive maps using our map styles. See [the section called “Map styles”](#) to learn more.

Interactive map

In this example, you will display an interactive map that allows users to zoom, pan, pinch, and pitch.

Interactive map code example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Display a map</title>
    <meta property="og:description"
content="Initialize a map in an HTML element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport"
content="width=device-width, initial-scale=1">
```

```

        <link rel='stylesheet' href='https://
unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.css' />
        <link rel='stylesheet'
href='style.css' />
        <script src='https://unpkg.com/maplibre-
gl@4.x/dist/maplibre-gl.js'></script>
    </head>
    <body>
        <!-- Map container -->
        <div id="map"></div>
        <script>
            const apiKey = "<API_KEY>";
            const mapStyle = "Standard"; //
e.g., Standard, Monochrome, Hybrid, Satellite
            const awsRegion = "eu-central-1"; //
e.g., us-east-2, us-east-1, us-west-2, etc.
            const styleUrl = `https://maps.geo.
${awsRegion}.amazonaws.com/v2/styles/${mapStyle}/descriptor?key=${apiKey}`;

            const map = new maplibregl.Map({
                container: 'map', // container
id
                style: styleUrl, // style URL
                center: [25.24, 36.31], //
starting position [lng, lat]
                zoom: 2, // starting zoom
            });
        </script>
    </body>
</html>

```

style.css

```

body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }

```

Restrict map panning beyond an area

In this example, you will restrict the map from being panned beyond a defined boundary.

Restrict map panning code example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Display a map</title>
    <meta property="og:description" content="Initialize
a map in an HTML element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/
maplibre-gl@4.x/dist/maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.js'></script>
  </head>
  <body>
    <!-- Map container -->
    <div id="map"></div>
    <script>
      const apiKey = "<API_KEY>";
      const mapStyle = "Standard"; // e.g., Standard,
Monochrome, Hybrid, Satellite
      const awsRegion = "eu-central-1"; // e.g., us-
east-2, us-east-1, us-west-2, etc.
      const styleUrl = `https://maps.geo.
${awsRegion}.amazonaws.com/v2/styles/${mapStyle}/descriptor?key=${apiKey}`;

      var bounds = [
        [90.0, -21.943045533438166], // Southwest
coordinates
        [146.25, 31.952162238024968] // Northeast
coordinates
      ];

      const map = new maplibregl.Map({
        container: 'map', // container id
        style: styleUrl, // style URL
        maxBounds: bounds, // Sets bounds of SE Asia
      });
```

```
        </script>
      </body>
    </html>
```

style.css

```
body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }
```

Non-interactive map

In this example, you will display a non-interactive map by disabling user interaction.

Non-interactive map code example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Display a map</title>
    <meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>
  </head>
  <body>
    <!-- Map container -->
    <div id="map"></div>
    <script>
      const apiKey = "<API_KEY>";
      const mapStyle = "Standard"; // e.g., Standard, Monochrome, Hybrid,
Satellite
```

```
const awsRegion = "eu-central-1"; // e.g., us-east-2, us-east-1, us-
west-2, etc.
const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?key=${apiKey}`;

const map = new maplibregl.Map({
  container: 'map', // container id
  style: styleUrl, // style URL
  center: [25.24, 36.31], // starting position [lng, lat]
  zoom: 2, // starting zoom
  interactive: false, // Disable pan & zoom handlers
});
</script>
</body>
</html>
```

style.css

```
body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }
```

How to add control on the map

Amazon Location Service allows you to add multiple controls to the map, including navigation, geolocation, fullscreen, scale, and attribution controls.

- **Navigation control:** Contains zoom buttons and a compass.
- **Geolocate control:** Provides a button that uses the browser's geolocation API to locate the user on the map.
- **Fullscreen control:** Contains a button for toggling the map in and out of fullscreen mode.
- **Scale control:** Displays the ratio of a distance on the map to the corresponding distance on the ground.
- **Attribution control:** Presents the map's attribution information. By default, the attribution control is expanded (regardless of map width).

You can add the controls to any corner of the map: top-left, bottom-left, bottom-right, or top-right.

Adding map controls

In the following example, you'll add the map controls listed above.

Map control code example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Map Controls</title>
    <meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>
  </head>
  <body>
    <!-- Map container -->
    <div id="map"></div>
    <script>
      const apiKey = "<API_KEY>";
      const mapStyle = "Standard"; // e.g., Standard, Monochrome, Hybrid,
Satellite
      const awsRegion = "eu-central-1"; // e.g., us-east-2, us-east-1, us-
west-2, etc.
      const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?key=${apiKey}`;

      const map = new maplibregl.Map({
        container: 'map', // container id
        style: styleUrl, // style URL
        center: [-123.13203602550998, 49.28726257639254], // starting
position [lng, lat]
        zoom: 10, // starting zoom
```



```
        attributionControl: false, // hide default attributionControl in
bottom left
        });

        // Adding controls to the map
        map.addControl(new maplibregl.NavigationControl()); // Zoom and
rotation controls
        map.addControl(new maplibregl.FullscreenControl()); // Fullscreen
control
        map.addControl(new maplibregl.GeolocateControl()); // Geolocation
control
        map.addControl(new maplibregl.AttributionControl(), 'bottom-
left'); // Attribution in bottom-left
        map.addControl(new maplibregl.ScaleControl(), 'bottom-right'); //
Scale control in bottom-right
    </script>
</body>
</html>
```

style.css

```
body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }
```

Developer tips

Navigation control options

```
new maplibregl.NavigationControl({
  showCompass: true, // show or hide compass (default: true)
  showZoom: true // show or hide zoom controls (default: true)
});
```

Geolocate control options

```
new maplibregl.GeolocateControl({
```

```
    positionOptions: { enableHighAccuracy: true }, // default: false
    trackUserLocation: true // default: false
  });
```

Attribution control options

```
new maplibregl.AttributionControl({
  compact: true, // compact (collapsed) mode (default: false)
});
```

Scale control options

```
new maplibregl.ScaleControl({
  maxWidth: 100, // width of the scale (default: 50)
  unit: 'imperial' // imperial or metric (default: metric)
});
```

Fullscreen control options

```
map.addControl(new maplibregl.FullscreenControl({
  container: document.querySelector('body') // container for fullscreen mode
}));
```

How to add a marker on the map

With Amazon Location, you can add both fixed and draggable markers, and you can also customize the color of the markers based on your data and preferences.

Add a fixed marker

Fixed marker code example

index.html

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Add marker on a map</title>
    <meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>
  </head>
  <body>
    <!-- Map container -->
    <div id="map"></div>
    <script>
      const apiKey = "<API_KEY>";
      const mapStyle = "Standard"; // e.g., Standard, Monochrome, Hybrid,
Satellite
      const awsRegion = "eu-central-1"; // e.g., us-east-2, us-east-1, us-
west-2, etc.
      const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?key=${apiKey}`;

      const map = new maplibregl.Map({
        container: 'map', // container id
        style: styleUrl, // style URL
        center: [0, 0], // starting position [lng, lat]
        zoom: 1, // starting zoom
      });

      const marker = new maplibregl.Marker() // Create fixed marker
        .setLngLat([85.1376, 25.5941]) // Set coordinates [long, lat]
for marker (e.g., Patna, BR, IN)
        .addTo(map); // Add marker to the map
    </script>
  </body>
</html>

```

style.css

```
body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }
```

Add a draggable marker

Draggable marker code example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Add draggable marker on a map</title>
    <meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>
  </head>
  <body>
    <!-- Map container -->
    <div id="map"></div>
    <script>
      const apiKey = "<API_KEY>";
      const mapStyle = "Standard"; // e.g., Standard, Monochrome, Hybrid,
Satellite
      const awsRegion = "eu-central-1"; // e.g., us-east-2, us-east-1, us-
west-2, etc.
      const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?key=${apiKey}`;

      const map = new maplibregl.Map({
        container: 'map', // container id
```

```

        style: styleUrl, // style URL
        center: [0, 0], // starting position [lng, lat]
        zoom: 1, // starting zoom
    });

    const marker = new maplibregl.Marker({ draggable: true }) // Create
draggable marker
        .setLngLat([85.1376, 25.5941]) // Set coordinates [long, lat]
for marker (e.g., Patna, BR, IN)
        .addTo(map); // Add marker to the map
</script>
</body>
</html>

```

style.css

```

body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }

```

Changing marker color

Colorful marker code example

index.html

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Adding colorful marker on a map</title>
    <meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>

```

```
</head>
<body>
  <!-- Map container -->
  <div id="map"></div>
  <script>
    const apiKey = "<API_KEY>";
    const mapStyle = "Monochrome"; // e.g., Standard, Monochrome,
Hybrid, Satellite
    const awsRegion = "eu-central-1"; // e.g., us-east-2, us-east-1, us-
west-2, etc.
    const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?key=${apiKey}`;

    const map = new maplibregl.Map({
      container: 'map', // container id
      style: styleUrl, // style URL
      center: [0, 0], // starting position [lng, lat]
      zoom: 1, // starting zoom
    });

    const marker = new maplibregl.Marker({ color: "black" }) // Create
colored marker
      .setLngLat([85.1376, 25.5941]) // Set coordinates [long, lat]
for marker (e.g., Patna, BR, IN)
      .addTo(map); // Add marker to the map
  </script>
</body>
</html>
```

style.css

```
body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }
```

Add multiple markers

Multiple markers code example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Adding multiple markers on a map</title>
    <meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>
  </head>
  <body>
    <!-- Map container -->
    <div id="map"></div>
    <script>
      const apiKey = "<API_KEY>";
      const mapStyle = "Standard"; // e.g., Standard, Monochrome, Hybrid,
Satellite
      const awsRegion = "eu-central-1"; // e.g., us-east-2, us-east-1, us-
west-2, etc.
      const colorScheme = "Dark"; // e.g., Dark, Light (default)
      const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?color-scheme=${colorScheme}&key=${apiKey}`;

      const map = new maplibregl.Map({
        container: 'map', // container id
        style: styleUrl, // style URL
        center: [0, 0], // starting position [lng, lat]
        zoom: 1, // starting zoom
      });

      const locations = [
        { lng: 85.1376, lat: 25.5941, label: 'Patna', color:
'#FF5722' },
```

```

        { lng: 77.1025, lat: 28.7041, label: 'Delhi', color:
'#2196F3' },
        { lng: 77.5946, lat: 12.9716, label: 'Bangalore', color:
'#FF9800' },
        { lng: 78.4867, lat: 17.3850, label: 'Hyderabad', color:
'#9C27B0' },
        { lng: -87.6298, lat: 41.8781, label: 'Chicago', color:
'#4CAF50' },
        { lng: -122.3321, lat: 47.6062, label: 'Seattle', color:
'#FFC107' },
        { lng: 4.3517, lat: 50.8503, label: 'Brussels', color:
'#3F51B5' },
        { lng: 2.3522, lat: 48.8566, label: 'Paris', color: '#E91E63' },
        { lng: -0.1276, lat: 51.5074, label: 'London', color:
'#795548' },
        { lng: 28.0473, lat: -26.2041, label: 'Johannesburg', color:
'#673AB7' },
        { lng: -123.1216, lat: 49.2827, label: 'Vancouver', color:
'#FF5722' },
        { lng: -104.9903, lat: 39.7392, label: 'Denver', color:
'#FF9800' },
        { lng: -97.7431, lat: 30.2672, label: 'Austin', color:
'#3F51B5' }
    ];

    // Loop through the locations array and add a marker for each one
    locations.forEach(location => {
        const marker = new maplibregl.Marker({ color: location.color,
draggable: true }) // Create colored marker
            .setLngLat([location.lng, location.lat]) // Set longitude
and latitude
            .addTo(map); // Add marker to the map
    });
</script>
</body>
</html>

```

style.css

```
body { margin: 0; padding: 0; }
```



```
html, body, #map { height: 100%; }
```

How to add an icon on the map

Amazon Location Service enables you to add icons, preferably in PNG format, to the map. You can add icons to specific geolocations and style them as needed.

Add a static icon

In this example, you will use an external URL to add an icon to the map using a symbol layer.

Static icon code example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Static icon on map</title>
    <meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>
  </head>
  <body>
    <!-- Map container -->
    <div id="map"></div>
    <script>
      const apiKey = "<API_KEY>";
      const mapStyle = "Standard"; // e.g., Standard, Monochrome, Hybrid,
Satellite
      const awsRegion = "eu-central-1"; // e.g., us-east-2, us-east-1, us-
west-2, etc.
      const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?key=${apiKey}`;
```

```
const map = new maplibregl.Map({
  container: 'map', // container id
  style: styleUrl, // style URL
  center: [-123.1144289, 49.2806672], // starting position [lng,
lat] (e.g., Vancouver)
  zoom: 12, // starting zoom
});

map.on('load', async () => {
  image = await map.loadImage('https://upload.wikimedia.org/
wikipedia/commons/1/1e/Biking_other.png');
  map.addImage('biking', image.data);
  map.addSource('point', {
    'type': 'geojson',
    'data': {
      'type': 'FeatureCollection',
      'features': [
        {
          'type': 'Feature',
          'geometry': {
            'type': 'Point',
            'coordinates': [-123.1144289, 49.2806672]
          }
        }
      ]
    }
  });
  map.addLayer({
    'id': 'points',
    'type': 'symbol',
    'source': 'point',
    'layout': {
      'icon-image': 'biking',
      'icon-size': 0.25
    }
  });
});
</script>
</body>
</html>
```

style.css

```
body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }
```

How to add a line on the map

With Amazon Location Service, you can add both pre-recorded GPS traces as line-strings and real-time GPS traces to dynamic maps.

Adding a pre-recorded line

In this example, you will add a pre-recorded GPS trace as a GeoJSON (main.js) to the dynamic map. To do so, you need to add a source (like GeoJSON) and a layer with line styling of your choice to the map.

Pre-recorded line code example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Add a line on the map</title>
    <meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>
    <script src='main.js'></script>
  </head>
  <body>
    <!-- Map container -->
    <div id="map"></div>
    <script>
```

```
const apiKey = "<API_KEY>";
const mapStyle = "Standard";
const awsRegion = "eu-central-1";
const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?key=${apiKey}`;

const map = new maplibregl.Map({
  container: 'map',
  style: styleUrl,
  center: [-123.126575, 49.290585],
  zoom: 12.5
});

map.on('load', () => {
  map.addSource('vancouver-office-to-stanley-park-route', {
    type: 'geojson',
    data: routeGeoJSON
  });

  map.addLayer({
    id: 'vancouver-office-to-stanley-park-route',
    type: 'line',
    source: 'vancouver-office-to-stanley-park-route',
    layout: {
      'line-cap': 'round',
      'line-join': 'round'
    },
    paint: {
      'line-color': '#008296',
      'line-width': 2
    }
  });
});
</script>
</body>
</html>
```

style.css

```
body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }
```

main.js

```
const routeGeoJSON = {
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      geometry: {
        type: "LineString",
        coordinates: [
          [-123.117011, 49.281306],
          [-123.11785, 49.28011],
          ...
          [-123.135735, 49.30106]
        ]
      },
      properties: {
        name: "Amazon YVR to Stanley Park",
        description: "An evening walk from Amazon office to Stanley
Park."
      }
    }
  ]
};
```

Add a line in real-time

In this example, you will simulate adding new GPS coordinates one by one to create a real-time GPS trace. This is useful for tracking real-time data updates.

Real-time line code example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
```

```
<title>Add a line on the map in real-time</title>
<meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
<meta charset='utf-8'>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
<link rel='stylesheet' href='style.css' />
<script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>
<script src='main.js'></script>
</head>
<body>
<!-- Map container -->
<div id="map"></div>
<script>
  const apiKey = "<API_KEY>";
  const mapStyle = "Standard";
  const awsRegion = "eu-central-1";
  const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?key=${apiKey}`;

  const map = new maplibregl.Map({
    container: 'map',
    style: styleUrl,
    center: [-123.126575, 49.290585],
    zoom: 12.5
  });

  map.on('load', () => {
    const coordinates =
routeGeoJSON.features[0].geometry.coordinates;
    routeGeoJSON.features[0].geometry.coordinates = [coordinates[0],
coordinates[20]];

    map.addSource('vancouver-office-to-stanley-park-route', {
      type: 'geojson',
      data: routeGeoJSON
    });

    map.addLayer({
      id: 'vancouver-office-to-stanley-park-route',
      type: 'line',
      source: 'vancouver-office-to-stanley-park-route',
```

```

        layout: {
            'line-cap': 'round',
            'line-join': 'round'
        },
        paint: {
            'line-color': '#008296',
            'line-width': 2
        }
    });

    map.jumpTo({center: coordinates[0], zoom: 14});
    map.setPitch(30);

    let i = 0;
    const timer = window.setInterval(() => {
        if (i < coordinates.length) {

            routeGeoJSON.features[0].geometry.coordinates.push(coordinates[i]);
            map.getSource('vancouver-office-to-stanley-park-
route').setData(routeGeoJSON);
            map.panTo(coordinates[i]);
            i++;
        } else {
            window.clearInterval(timer);
        }
    }, 100);
});
</script>
</body>
</html>

```

style.css

```

body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }

```

main.js

```

const routeGeoJSON = {

```

```
    type: "FeatureCollection",
    features: [
      {
        type: "Feature",
        geometry: {
          type: "LineString",
          coordinates: [
            [-123.117011, 49.281306],
            [-123.11785, 49.28011],
            ...
            [-123.135735, 49.30106]
          ]
        },
        properties: {
          name: "Amazon YVR to Stanley Park",
          description: "An evening walk from Amazon office to Stanley
Park."
        }
      }
    ]
  };
```

Developer tips

Fitting bounds: You can fit the line to the map bounds by calculating the bounds of the line's coordinates:

```
const coordinates = routeGeoJSON.features[0].geometry.coordinates;
const bounds = coordinates.reduce((bounds, coord) => bounds.extend(coord), new
maplibregl.LngLatBounds(coordinates[0], coordinates[0]));
map.fitBounds(bounds, { padding: 20 });
```

How to add a polygon on the map

Amazon Location Service allows you to add polygons to the map. You can style the polygon based on your business needs, including adding fill and border styling.

Add a polygon

In this example, you will add a polygon to the map and style it with a fill color and a border.

Polygon code example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Overlay a Polygon on a Map</title>
    <meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>
    <script src='main.js'></script>
  </head>
  <body>
    <!-- Map container -->
    <div id="map"></div>
    <script>
      const apiKey = "<API_KEY>";
      const mapStyle = "Standard";
      const awsRegion = "eu-central-1";
      const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?key=${apiKey}`;

      const map = new maplibregl.Map({
        container: 'map',
        style: styleUrl,
        center: [-123.13203602550998, 49.28726257639254],
        zoom: 16
      });

      map.on('load', () => {
        map.addSource('barclayHeritageSquare', {
          type: 'geojson',
```

```
        data: barclayHeritageSquare
    });

    map.addLayer({
      id: 'barclayHeritageSquare-fill',
      type: 'fill',
      source: 'barclayHeritageSquare',
      layout: {},
      paint: {
        'fill-color': '#008296',
        'fill-opacity': 0.25
      }
    });

    map.addLayer({
      id: 'barclayHeritageSquare-outline',
      type: 'line',
      source: 'barclayHeritageSquare',
      layout: {},
      paint: {
        'line-color': '#008296',
        'line-width': 2
      }
    });
  });
</script>
</body>
</html>
```

style.css

```
body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }
```

main.js

```
const barclayHeritageSquare = {
  "type": "FeatureCollection",
  "features": [
```

```

    {
      "type": "Feature",
      "properties": {
        "park_id": 200,
        "park_name": "Barclay Heritage Square",
        "area_ha": 0.63255076039675,
        "park_url": "http://covapp.vancouver.ca/parkfinder/
parkdetail.aspx?inparkid=200"
      },
      "geometry": {
        "type": "MultiPolygon",
        "coordinates": [
          [
            [-123.1320948511985, 49.287379401361406],
            [-123.13179672786798, 49.2871908081159],
            [-123.13148154587924, 49.28739992437733],
            [-123.1313830551372, 49.28733617069321],
            [-123.13118648745055, 49.287208851500054],
            [-123.13128257706838, 49.28714532642171],
            [-123.13147941881572, 49.28727228533418],
            ...
            [-123.13177619357138, 49.28759081706052],
            [-123.1320948511985, 49.287379401361406]
          ]
        ]
      }
    }
  ]
};

```

How to add a popup to a map

Amazon Location Service allows you to add popups to the map. You can add simple popups, click-triggered popups on markers, hover-triggered popups, and popups for multiple markers.

Add your first popup

In this example, you will add your first popup.

First popup code example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Display a map</title>
    <meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>
  </head>
  <body>
    <!-- Map container -->
    <div id="map"></div>
    <script>
      const apiKey = "<API_KEY>";
      const mapStyle = "Standard";
      const awsRegion = "eu-central-1";
      const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?key=${apiKey}`;

      const map = new maplibregl.Map({
        container: 'map',
        style: styleUrl,
        center: [-96, 37.8],
        zoom: 2
      });

      const popup = new maplibregl.Popup({closeOnClick: false})
        .setLngLat([-96, 37.8])
        .setHTML('<h1>Hello USA!</h1>')
        .addTo(map);
    </script>
  </body>
</html>
```

style.css

```
body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }
```

Show popup on click on a marker

In this example, you will attach a popup to a marker and display it on click.

Popup on marker click example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Display a map</title>
    <meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>
  </head>
  <body>
    <!-- Map container -->
    <div id="map"></div>
    <script>
      const apiKey = "<API_KEY>";
      const mapStyle = "Standard";
      const awsRegion = "eu-central-1";
      const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?key=${apiKey}`;
```

```
const centralpark_nyc = [-73.966,40.781];
const map = new maplibregl.Map({
  container: 'map',
  style: styleUrl,
  center: centralpark_nyc,
  zoom: 13
});

const popup = new maplibregl.Popup({offset: 25}).setText(
  'Central Park, NY is one of the most filmed locations in the
world, appearing in over 240 feature films since 1908.'
);

new maplibregl.Marker()
  .setLngLat(centralpark_nyc)
  .setPopup(popup)
  .addTo(map);
</script>
</body>
</html>
```

style.css

```
body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }
```

Show popup on hover on a marker

In this example, you will attach a popup to a marker and display it on hover.

Popup on marker hover example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Display a map</title>
```

```
    <meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></
script>
  </head>
  <body>
    <!-- Map container -->
    <div id="map"></div>
    <script>
      const apiKey = "<API_KEY>";
      const mapStyle = "Standard";
      const awsRegion = "eu-central-1";
      const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/
styles/${mapStyle}/descriptor?key=${apiKey}`;

      const centralpark_nyc = [-73.966,40.781];
      const map = new maplibregl.Map({
        container: 'map',
        style: styleUrl,
        center: centralpark_nyc,
        zoom: 13
      });

      const marker = new maplibregl.Marker().setLngLat([-73.968285,
40.785091]).addTo(map);
      const popup = new maplibregl.Popup({ offset: 25 })
        .setHTML("<h3>Central Park</h3><p>Welcome to Central Park, NYC!
</p>");

      const markerElement = marker.getElement();
      markerElement.addEventListener('mouseenter', () => {
        popup.setLngLat([-73.968285, 40.785091]).addTo(map);
      });
      markerElement.addEventListener('mouseleave', () => {
        popup.remove();
      });
    </script>
  </body>
</html>
```

style.css

```
body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }
```

Show popup on click on multiple markers

In this example, you will attach a popup to multiple markers and display it on click.

Popup on click on multiple markers example

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Display a map</title>
    <meta property="og:description" content="Initialize a map in an HTML
element with MapLibre GL JS." />
    <meta charset='utf-8'>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/
maplibre-gl.css' />
    <link rel='stylesheet' href='style.css' />
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-
gl.js'></script>
  </head>
  <body>
    <!-- Map container -->
    <div id="map"></div>
    <script>
      const apiKey = "Your API Key";
      const mapStyle = "Monochrome";
      const awsRegion = "eu-central-1";
      const colorScheme = "Light";
      const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/
v2/styles/${mapStyle}/descriptor?color-scheme=${colorScheme}&key=${apiKey}`;
```



```
const map = new maplibregl.Map({
  container: 'map',
  style: styleUrl,
  center: [-123.126979, 49.2841563],
  zoom: 15,
  minZoom: 13,
  maxZoom: 17
});

const locations = [
  { id: 1, lat: 49.281108, lng: -123.117049, name: "Amazon -
YVR11 office" },
  { id: 2, lat: 49.285580, lng: -123.115806, name: "Amazon -
YVR20 office" },
  { id: 3, lat: 49.281661, lng: -123.114174, name: "Amazon -
YVR14 office" },
  { id: 4, lat: 49.280663, lng: -123.114379, name: "Amazon -
YVR26 office" },
  { id: 5, lat: 49.285343, lng: -123.129119, name: "Amazon -
YVR25 office" }
];

const geojson = {
  type: "FeatureCollection",
  features: locations.map(location => ({
    type: "Feature",
    properties: { id: location.id, name: location.name },
    geometry: {
      type: "Point",
      coordinates: [location.lng, location.lat]
    }
  })))
};

map.on('load', async () => {
  try {
    const image = await loadImage('https://
upload.wikimedia.org/wikipedia/commons/thumb/9/93/
Amazon_Web_Services_Logo.svg/1200px-Amazon_Web_Services_Logo.svg.png');
    map.addImage('aws', image);

    map.addSource('places', { type: 'geojson', data:
geojson });
```

```
        map.addLayer({
          'id': 'places',
          'type': 'symbol',
          'source': 'places',
          'layout': {
            'icon-image': 'aws',
            'icon-size': 0.025,
            'icon-allow-overlap': true
          }
        });

        const popup = new maplibregl.Popup({ closeButton: false,
closeOnClick: false });

        map.on('click', 'places', (e) => {
          map.getCanvas().style.cursor = 'pointer';
          const coordinates =
e.features[0].geometry.coordinates.slice();
          const name = e.features[0].properties.name;

          popup.setLngLat(coordinates).setHTML(name).addTo(map);
        });

        map.on('mouseleave', 'places', () => {
          map.getCanvas().style.cursor = '';
          popup.remove();
        });
      } catch (error) {
        console.error('Error loading image:', error);
      }
    });

    async function loadImage(url) {
      return new Promise((resolve, reject) => {
        const img = new Image();
        img.crossOrigin = 'anonymous';
        img.onload = () => resolve(img);
        img.onerror = (error) => reject(error);
        img.src = url;
      });
    }
  }
</script>
</body>
```

```
</html>
```

style.css

```
body { margin: 0; padding: 0; }  
html, body, #map { height: 100%; }
```

Show popup on hover on multiple markers

In this example, you will attach a popup to multiple markers and display it on hover.

Popup on hover on multiple markers example

index.html

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <title>Display a map</title>  
    <meta property="og:description" content="Initialize a map in an HTML element  
with MapLibre GL JS." />  
    <meta charset='utf-8'>  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <link rel='stylesheet' href='https://unpkg.com/maplibre-gl@4.x/dist/  
maplibre-gl.css' />  
    <link rel='stylesheet' href='style.css' />  
    <script src='https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js'></  
script>  
  </head>  
  <body>  
    <!-- Map container -->  
    <div id="map" style="width: 100%; height: 100vh;"></div>  
    <script>  
      const apiKey = "You API Key";  
      const mapStyle = "Monochrome"; // eg. Standard, Monochrome, Hybrid,  
Satellite  
      const awsRegion = "eu-central-1"; // eg. us-east-2, us-east-1, etc.  
      const colorScheme = "Light"; // eg Dark, Light (default)
```

```
const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/styles/
${mapStyle}/descriptor?color-scheme=${colorScheme}&key=${apiKey}`;

// Initialize the map
const map = new maplibregl.Map({
  container: 'map', // Container id
  style: styleUrl, // Style URL
  center: [-123.126979, 49.2841563], // Starting position [lng, lat]
  zoom: 15, // Starting zoom
  minZoom: 13, // Minimum zoom level
  maxZoom: 17 // Maximum zoom level
});

const locations = [
  office" },
  { id: 1, lat: 49.281108, lng: -123.117049, name: "Amazon - YVR11
office" },
  { id: 2, lat: 49.285580, lng: -123.115806, name: "Amazon - YVR20
office" },
  { id: 3, lat: 49.281661, lng: -123.114174, name: "Amazon - YVR14
office" },
  { id: 4, lat: 49.280663, lng: -123.114379, name: "Amazon - YVR26
office" },
  { id: 5, lat: 49.285343, lng: -123.129119, name: "Amazon - YVR25
office" }
];

// Convert locations to GeoJSON
const geojson = {
  type: "FeatureCollection",
  features: locations.map(location => ({
    type: "Feature",
    properties: {
      id: location.id,
      name: location.name // Use the name property for popup
    },
    geometry: {
      type: "Point",
      coordinates: [location.lng, location.lat] // GeoJSON uses
[lng, lat]
    }
  })))
};

// Add the GeoJSON source and layers when the map loads
```

```
map.on('load', async () => {
  try {
    // Load the PNG image for the marker
    const image = await loadImage('https://upload.wikimedia.org/
wikipedia/commons/thumb/9/93/Amazon_Web_Services_Logo.svg/1200px-
Amazon_Web_Services_Logo.svg.png'); // Ensure this URL points to a valid PNG
    map.addImage('aws', image);

    // Add a GeoJSON source
    map.addSource('places', {
      type: 'geojson',
      data: geojson
    });

    // Add a layer showing the places with custom icons
    map.addLayer({
      'id': 'places',
      'type': 'symbol',
      'source': 'places',
      'layout': {
        'icon-image': 'aws',
        'icon-size': 0.025, // Adjust as needed
        'icon-allow-overlap': true // Allow icons to overlap
      }
    });

    // Create a popup
    const popup = new maplibregl.Popup({
      closeButton: false,
      closeOnClick: false
    });

    // Event handlers for mouse enter and leave
    map.on('mouseenter', 'places', (e) => {
      map.getCanvas().style.cursor = 'pointer';
      const coordinates =
e.features[0].geometry.coordinates.slice();
      const name = e.features[0].properties.name;

      // Set popup content and position
      popup.setLngLat(coordinates).setHTML(name).addTo(map);
    });

    map.on('mouseleave', 'places', () => {
```

```
        map.getCanvas().style.cursor = '';
        popup.remove();
    });
} catch (error) {
    console.error('Error loading image:', error);
}
});

// Helper function to load the image
async function loadImage(url) {
    return new Promise((resolve, reject) => {
        const img = new Image();
        img.crossOrigin = 'anonymous'; // Enable CORS
        img.onload = () => resolve(img);
        img.onerror = (error) => reject(error);
        img.src = url;
    });
}
</script>
</body>
</html>
```

style.css

```
body { margin: 0; padding: 0; }
html, body, #map { height: 100%; }
```

How to set a preferred language for a map

Amazon Location Service enables you to set the preferred language at the client-side by updating the style descriptor for a specific language. You can set a preferred language and display content in that language where embedded. Otherwise, it will fall back to another language.

Note

For more information, see [the section called “Localization and internationalization”](#).

Set preferred language to Japanese and show map of Japan

In this example, you will set update style to show map labels in Japanese (ja).

Set preferred language to Japanese example

index.html

```
<html>
<head>
  <link href="https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.css"
rel="stylesheet" />
  <script src="https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js"></script>
  <link href="style.css" rel="stylesheet" />
  <script src="main.js"></script>
</head>
<body>
  <div id="map" />
  <script>
    const apiKey = "Add Your Api Key";
    const mapStyle = "Standard";
    const awsRegion = "eu-central-1";
    const initialLocation = [139.76694, 35.68085]; //Japan

    async function initializeMap() {
      // get updated style object for preferred language.
      const styleObject = await getStyleWithPreferredLanguage("ja");
      // Initialize the MapLibre map with the fetched style object
      const map = new maplibregl.Map({
        container: 'map',
        style: styleObject,
        center: initialLocation,
        zoom: 15,
        hash:true,
      });
      map.addControl(new maplibregl.NavigationControl(), "top-left");

      return map;
    }

    initializeMap();
  </script>
</body>
```

```
</html>
```

style.css

```
body { margin: 0; }  
#map { height: 100vh; }
```

main.js

```
async function getStyleWithPreferredLanguage(preferredLanguage) {  
  const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/styles/  
${mapStyle}/descriptor?key=${apiKey}`;  
  
  return fetch(styleUrl)  
    .then(response => {  
      if (!response.ok) {  
        throw new Error('Network response was not ok ' + response.statusText);  
      }  
      return response.json();  
    })  
    .then(styleObject => {  
      if (preferredLanguage !== "en") {  
        styleObject = setPreferredLanguage(styleObject, preferredLanguage);  
      }  
  
      return styleObject;  
    })  
    .catch(error => {  
      console.error('Error fetching style:', error);  
    });  
}  
  
const setPreferredLanguage = (style, language) => {  
  let nextStyle = { ...style };  
  
  nextStyle.layers = nextStyle.layers.map(l => {  
    if (l.type !== 'symbol' || !l?.layout?.['text-field']) return l;  
    return updateLayer(l, /^name:([A-Za-z\-\_\~]+)$/g, `name:${language}`);  
  });  
};
```



```
    return nextStyle;
  };

const updateLayer = (layer, prevPropertyRegex, nextProperty) => {
  const nextLayer = {
    ...layer,
    layout: {
      ...layer.layout,
      'text-field': recurseExpression(
        layer.layout['text-field'],
        prevPropertyRegex,
        nextProperty
      )
    }
  };
  return nextLayer;
};

const recurseExpression = (exp, prevPropertyRegex, nextProperty) => {
  if (!Array.isArray(exp)) return exp;
  if (exp[0] !== 'coalesce') return exp.map(v =>
    recurseExpression(v, prevPropertyRegex, nextProperty)
  );

  const first = exp[1];
  const second = exp[2];

  let isMatch =
    Array.isArray(first) &&
    first[0] === 'get' &&
    !!first[1].match(prevPropertyRegex)?.[0];

  isMatch = isMatch && Array.isArray(second) && second[0] === 'get';
  isMatch = isMatch && !exp?.[4];

  if (!isMatch) return exp.map(v =>
    recurseExpression(v, prevPropertyRegex, nextProperty)
  );

  return [
    'coalesce',
    ['get', nextProperty],
    ['get', 'name:en'],
  ];
};
```

```
    ['get', 'name']
  ];
};
```

Set preferred language based on end user's browser language

In this example, you will set update style to show map labels in user's device language.

Set preferred language based on browser language example

index.html

```
<html>
<head>
  <link href="https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.css"
rel="stylesheet" />
  <script src="https://unpkg.com/maplibre-gl@4.x/dist/maplibre-gl.js"></script>
  <link href="style.css" rel="stylesheet" />
  <script src="main.js"></script>
</head>
<body>
  <div id="map" />
  <script>
    const apiKey = "Add Your Api Key";
    const mapStyle = "Standard";
    const awsRegion = "eu-central-1";
    const initialLocation = [139.76694, 35.68085]; //Japan
    const userLanguage = navigator.language || navigator.userLanguage;
    const languageCode = userLanguage.split('-')[0];

    async function initializeMap() {
      const styleObject = await getStyleWithPreferredLanguage(languageCode);
      const map = new maplibregl.Map({
        container: 'map',
        style: styleObject,
        center: initialLocation,
        zoom: 15,
        hash:true,
      });
      map.addControl(new maplibregl.NavigationControl(), "top-left");
      return map;
    }
  </script>
</body>
</html>
```

```
    }

    initializeMap();
  </script>
</body>
</html>
```

style.css

```
body { margin: 0; }
#map { height: 100vh; }
```

main.js

```
async function getStyleWithPreferredLanguage(preferredLanguage) {
  const styleUrl = `https://maps.geo.${awsRegion}.amazonaws.com/v2/styles/
${mapStyle}/descriptor?key=${apiKey}`;

  return fetch(styleUrl)
    .then(response => {
      if (!response.ok) {
        throw new Error('Network response was not ok ' + response.statusText);
      }
      return response.json();
    })
    .then(styleObject => {
      if (preferredLanguage !== "en") {
        styleObject = setPreferredLanguage(styleObject, preferredLanguage);
      }

      return styleObject;
    })
    .catch(error => {
      console.error('Error fetching style:', error);
    });
}

const setPreferredLanguage = (style, language) => {
  let nextStyle = { ...style };

```

```
nextStyle.layers = nextStyle.layers.map(l => {
  if (l.type !== 'symbol' || !l?.layout?.['text-field']) return l;
  return updateLayer(l, /^name:([A-Za-z\-\_\~]+)$/g, `name:${language}`);
});

return nextStyle;
};

const updateLayer = (layer, prevPropertyRegex, nextProperty) => {
  const nextLayer = {
    ...layer,
    layout: {
      ...layer.layout,
      'text-field': recurseExpression(
        layer.layout['text-field'],
        prevPropertyRegex,
        nextProperty
      )
    }
  };
  return nextLayer;
};

const recurseExpression = (exp, prevPropertyRegex, nextProperty) => {
  if (!Array.isArray(exp)) return exp;
  if (exp[0] !== 'coalesce') return exp.map(v =>
    recurseExpression(v, prevPropertyRegex, nextProperty)
  );

  const first = exp[1];
  const second = exp[2];

  let isMatch =
    Array.isArray(first) &&
    first[0] === 'get' &&
    !!first[1].match(prevPropertyRegex)?.[0];

  isMatch = isMatch && Array.isArray(second) && second[0] === 'get';
  isMatch = isMatch && !exp?.[4];

  if (!isMatch) return exp.map(v =>
    recurseExpression(v, prevPropertyRegex, nextProperty)
  );
};
```

```
return [  
  'coalesce',  
  ['get', nextProperty],  
  ['get', 'name:en'],  
  ['get', 'name']  
];  
};
```

How to set the political view of a map

Amazon Location Service enables you to set the political view to ensure your application complies with local laws. You can set a political view and compare maps from different political perspectives.

Note

For more information, see [the section called “Localization and internationalization”](#).

Set political view and compare with international perspective

In this example, you will create and compare maps from two different political views: an international perspective and Turkey's view on Cyprus.

Political view comparison example

index.html

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <title>Display a map</title>  
    <meta property="og:description" content="Initialize  
a map in an HTML element with MapLibre GL JS." />  
    <meta charset='utf-8'>  
    <meta name="viewport" content="width=device-width,  
initial-scale=1">  
    <link rel='stylesheet' href='https://unpkg.com/  
maplibre-gl@4.x/dist/maplibre-gl.css' />  
    <link rel='stylesheet' href='style.css' />
```

```

maplibre-gl.js'></script>
maplibre-gl.js'></script>
<script src="https://unpkg.com/@mapbox/mapbox-gl-
sync-move@0.3.1"></script>
<script src='main.js'></script>
</head>
<body>
  <!-- Map container -->
  <div class="maps">
    <div id="internatinalView"></div>
    <div id="turkeyView"></div>
  </div>
  <script>

    const apiKey = "Add Your Api Key";
    const mapStyle = "Standard";
    const awsRegion = "eu-central-1";

    // International perspective without political-
view query parameter

    const internatinalViewMapStyleUrl = `https://
maps.geo.${awsRegion}.amazonaws.com/v2/styles/${mapStyle}/descriptor?key=${apiKey}`;

    // Turkey perspective with political-view query
parameter

    const turkeyViewMapStyleUrl = `https://maps.geo.
${awsRegion}.amazonaws.com/v2/styles/${mapStyle}/descriptor?political-view=TUR&key=
${apiKey}`;

    const internatinalViewMap = new maplibregl.Map({
      container: 'internatinalView', // container
      id
      style: internatinalViewMapStyleUrl, // style
      URL
      center: [33.0714561, 35.1052139], //
      starting position [lng, lat]
      zoom: 7.5,
    });

    const turkeyViewMap = new maplibregl.Map({
      container: 'turkeyView', // container id
      style: turkeyViewMapStyleUrl, // style URL
      center: [33.0714561, 35.1052139],
      zoom: 7.5,

```

```

    });

    // Sync map zoom and center
    syncMaps(internatinalViewMap, turkeyViewMap);

    // Informational popup for international view
    new maplibregl.Popup({closeOnClick: false})
      .setLngLat([33, 35.5])
      .setHTML('<h4>International view <br>
recognizes <br> Cyprus</h4>')
      .addTo(internatinalViewMap);

    // Informational popup for Turkey's view
    new maplibregl.Popup({closeOnClick: false})
      .setLngLat([33, 35.5])
      .setHTML('<h4>Turkey does not <br> recognize
<br> Cyprus</h4>')
      .addTo(turkeyViewMap);
  </script>
</body>
</html>

```

style.css

```

body { margin: 0; padding: 0; }
html, body { height: 100%; }
#internatinalView, #turkeyView { height: 100%; width:
100%; }

.maps {
  display: flex;
  width: 100%;
  height: 100%;
}

```

How to use static maps

These how-to topics offer step-by-step guidance for customizing static maps with overlays and visual elements, utilizing the mapping capabilities of Amazon Location Service. Each guide walks you through key tasks, such as adjusting map dimensions, choosing between zoom or radius, and

adding various geo-spatial elements like markers, polygons, lines, and proximity circles. These guides use both compact and GeoJSON formats to ensure flexibility in styling and configuration.

Whether you're optimizing maps for customer-facing applications or performing geospatial analysis, these topics provide clear instructions on tailoring your static maps to fit specific requirements. By following the detailed examples, you'll learn how to enhance map presentations and efficiently manage geographical data for a variety of business needs.

Topics

- [How to get a static map of a specific position](#)
- [How to get a static map of a specific dimension](#)
- [How to decide between radius and zoom for a static map](#)
- [How to add scale for a static map](#)
- [How to add a marker to a static map](#)
- [How to add a line to a static map](#)
- [How to add a route to a static map](#)

How to get a static map of a specific position

In this topic, you will learn how to retrieve static maps from Amazon Location Service based on specific parameters. You will learn how to generate a static map for a center position, a bounding box, and a set of bounded positions. The examples provided will help you customize the width, height, and style of the map.

Note

You must pass either `map` or `map@2x` when generating a static map.

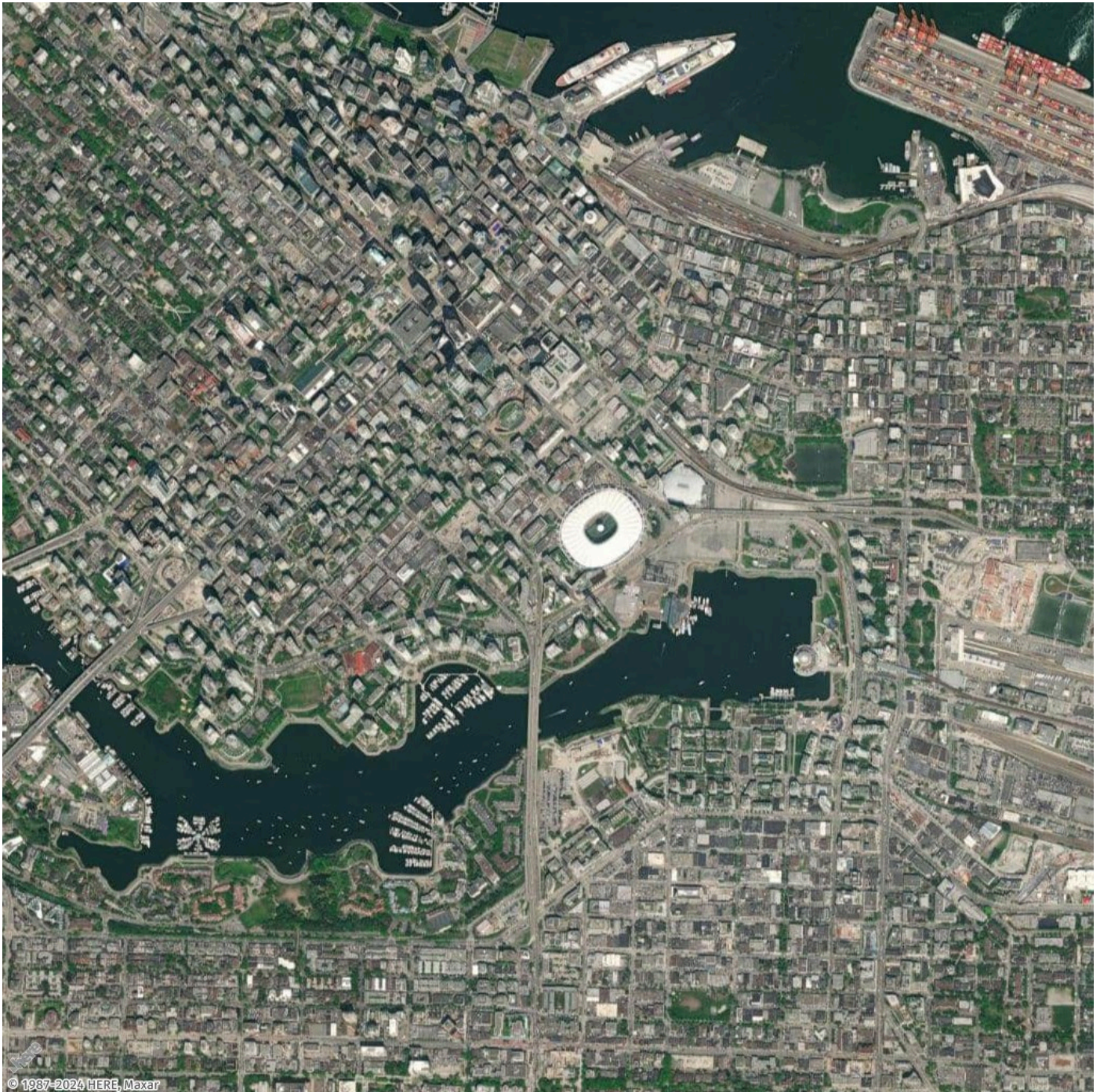
Get map image for a center position

In this example, you will create a map image with a width of 1024 and a height of 1024 with the center coordinates set at `-123.1143, 49.2763`, where `longitude=-123.1143` and `latitude=49.2763`, and a zoom level of 15.

Request URL

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?  
style=Satellite&width=1024&height=1024&zoom=15&center=-123.1156126,49.2767046&key=API_KEY
```

Response image



Get map image for bounding box

In this example, you'll generate a map image of Southeast Asia by setting the bounding box for the area.

The bbox format is {southwest_longitude}, {southwest_latitude}, {northeast_longitude}, {northeast_latitude}.

Request URL

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/  
map?style=Satellite&width=1024&height=1024&bounding-  
box=90.00,-21.94,146.25,31.95&key=API_KEY
```


Response image



Get map image for bounded positions

In this example, you will generate a map that covers several must-see places in Paris, each specified by its coordinates (longitude, latitude). The bounded positions include: Eiffel Tower,

Louvre Museum, Notre-Dame Cathedral, Champs-Élysées, Arc de Triomphe, Sacré-Cœur Basilica, Luxembourg Gardens, Musée d'Orsay, Place de la Concorde, and Palais Garnier.

The format for bounding positions is `{longitude1},{latitude1},{longitude2},{latitude2} ... {longitudeN},{latitudeN}`.

Request URL

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?
style=Satellite&width=1024&height=1024&bounded-
positions=2.2945,48.8584,2.3376,48.8606,2.3500,48.8529,2.3076,48.8698,2.2950,48.8738,2.3431,
```


Response image



How to get a static map of a specific dimension

In this topic, you will learn how to set the dimensions (height and width) for static maps using Amazon Location Service. Customizing the dimensions of a map image allows you to balance

performance, visual quality, and usability. The maximum values for both width and height are 1400 pixels, while the minimum values are 64 pixels. The maximum result size is 6 MB.

Additionally, you can use the `bbox` and `bounds` parameters along with padding to ensure that important map features near the edges are fully visible and not cut off.

Get map image with specific height and width

In this example, you will create a low-resolution and mid-resolution map image of Helsinki, Finland.

Request URL for low-resolution thumbnail

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?
style=Satellite&width=200&height=200&zoom=11.5&center=24.9460,60.1690&key=API_KEY
```

Response (Thumbnail 200x200)



Request URL for mid-resolution image

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?
style=Satellite&width=700&height=700&zoom=13&center=24.9460,60.1690&key=API_KEY
```

Response image (700x700)



Get map image with padding on all sides

In this example, you will generate a map using several must-see places in Helsinki, Finland, with their coordinates (longitude, latitude), both with and without padding.

How to decide between radius and zoom for a static map

In this topic, you will learn how to choose between using `radius` or `zoom` when generating static maps with Amazon Location Service. The `radius` parameter provides more precise control over the area of coverage, making it ideal for customer-facing applications where you know the exact coverage area. The `zoom` parameter is better suited for geospatial analysis when you want to adjust the level of detail displayed.

With radius

In this example, you will create a map image of Sri Lanka using the `radius` parameter with a center location.

Request URL

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?
style=Satellite&width=700&height=700&center=80.60596,7.76671&radius=235000&scale-
unit=KilometersMiles&key=API_KEY
```


Response image



With zoom

In this example, you will create a map image of Sri Lanka using the zoom parameter with a center location.

Request URL

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?
style=Satellite&width=700&height=700&zoom=8&center=80.60596,7.76671&scale-
unit=KilometersMiles&key=API_KEY
```

Response image



How to add scale for a static map

In this topic, you will learn how to display a scale on the bottom-right corner of a static map generated with Amazon Location Service. The scale can show a single unit, such as Miles or Kilometers, or both units, such as KilometersMiles or MilesKilometers, with one unit displayed at the top and the other at the bottom.

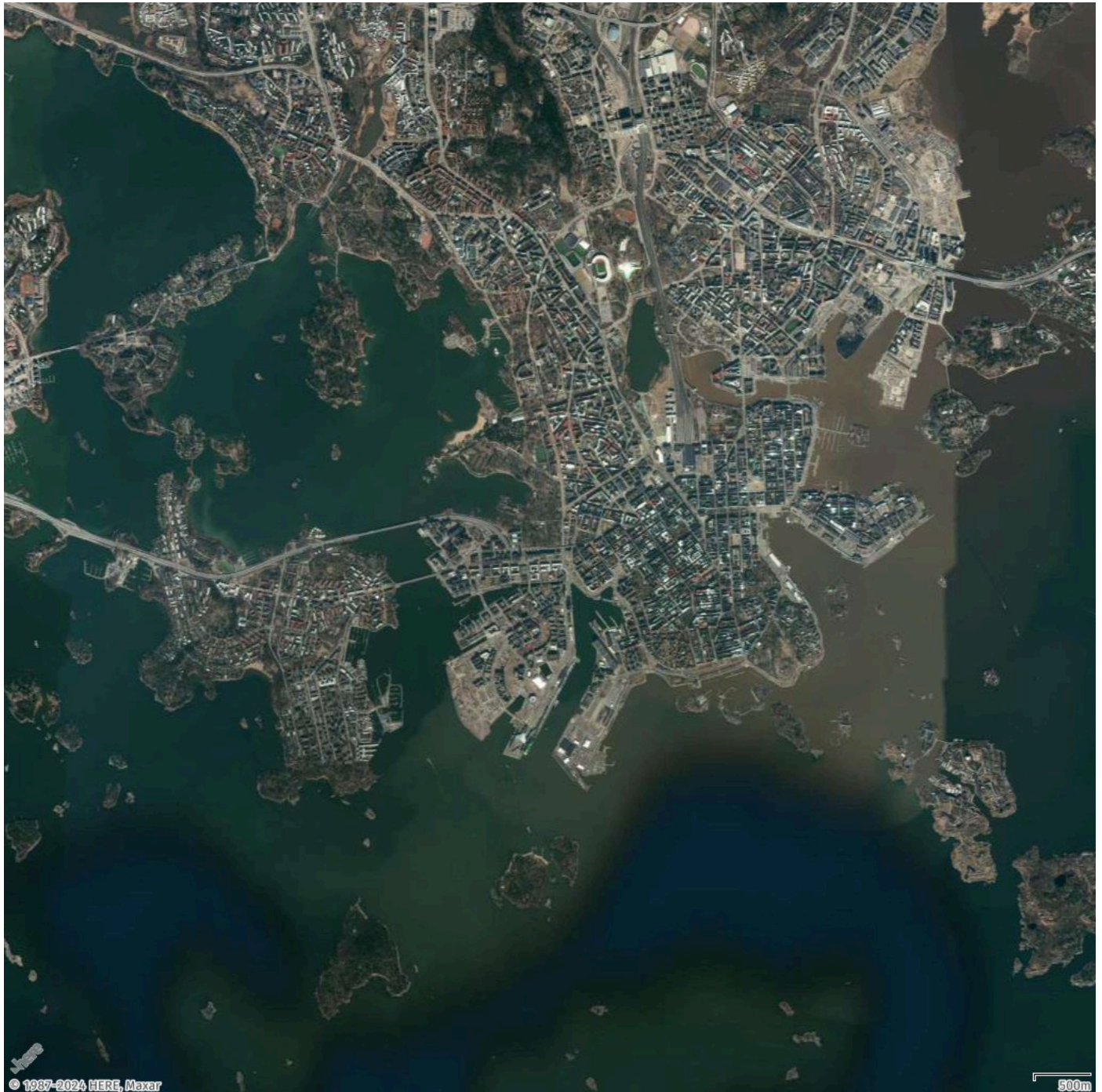
Add scale with single unit

In this example, you will display a map of Helsinki, Finland with the scale set to Kilometers in the bottom-right corner.

Request URL

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?
style=Satellite&width=1024&height=1024&zoom=13.5&center=24.9189564,60.1645772&scale-
unit=Kilometers&key=API_KEY
```


Response image



Add scale with both units

In this example, you will display a map of Helsinki, Finland with both Kilometers and Miles shown on the scale in the bottom-right corner, with one unit displayed above the other.

Request URL

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?  
style=Satellite&width=1024&height=1024&zoom=14&center=24.9189564,60.1645772&scale-  
unit=KilometersMiles&key=API_KEY
```

Response image



How to add a marker to a static map

In this topic, you will learn how to add markers to static maps generated with Amazon Location Service. You can customize the marker's icon, label, size, color, and other styling options. This allows you to highlight specific locations with visual cues that align with your map's purpose.

Marker styling

Name	Type	Default	Description
<code>style</code>	String	<code>marker</code>	The style of the Point geometry. To create a marker, set the value to <code>marker</code> or do not include the <code>style</code> attribute in the properties object of the GeoJSON.
<code>icon</code>	String	<code>pin</code>	The marker icon type. Available values include: <code>cp</code> (proximity circle), <code>diamond</code> , <code>pin</code> , <code>poi</code> , <code>square</code> , <code>triangle</code> , <code>bubble</code> .
<code>label</code>	String	<code><empty></code>	The text to display at the coordinates. To set automatic labels, use the following values: <code>\$alpha</code> (Latin alphabet) or <code>\$num</code> (Arabic numerals).
<code>color</code>	Color	Style-dependent	The icon color.
<code>outline-color</code>	Color	Style-dependent	The icon outline color.

Name	Type	Default	Description
text-color	Color	Style-dependent	The label text color.
text-outline-color	Color	Style-dependent	The text outline color.
outline-width	Integer	0 (turned off)	The text outline width.
size	String	medium	The label and icon size. Available values: small, medium, large.

Add a Marker to a Map Image

In this example, you will place a marker with a label on the map image of BC Place, Vancouver.

Geo JSON:

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [
          -123.11210,
          49.2767875
        ]
      },
      "properties": {
        "color": "#EE4B2B",
        "size": "large",
        "label": "BC Place, Vancouver",
        "text-color": "#0000FF"
      }
    }
  ]
}
```

```
}
```

Compact:

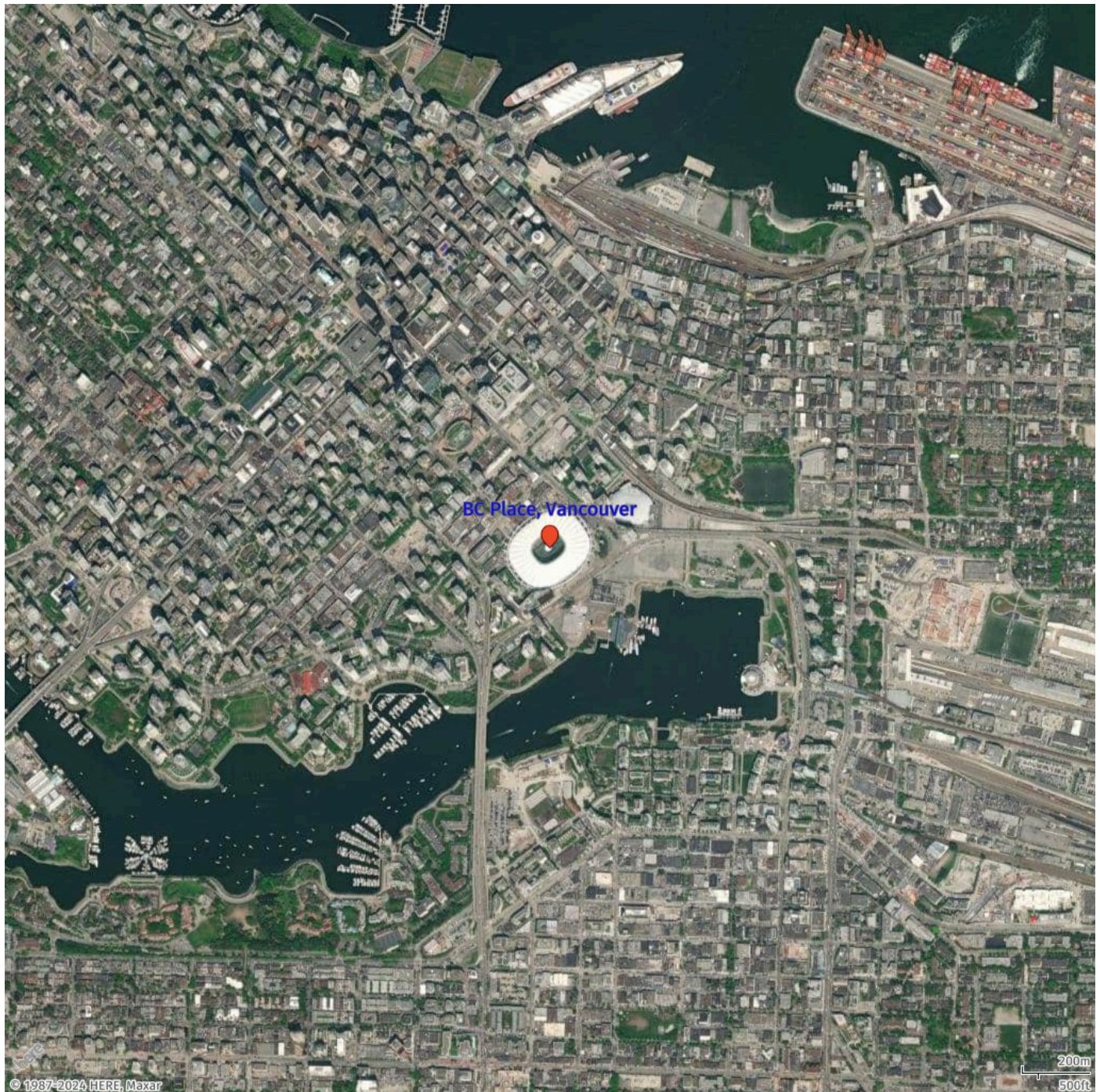
```
point: -123.11210,49.27678;  
label=BC Place, Vancouver;  
size=large;  
text-color=#0000FF;  
color=#EE4B2B
```

Request URL

With the GeoJSON overlay

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?  
style=Satellite&width=1024&height=1024&zoom=15&scale-unit=KilometersMiles&geojson-  
overlay=%7B%22type%22%3A%22FeatureCollection%22,%22features%22%3A%5B%7B  
%22type%22%3A%22Feature%22,%22properties%22%3A%7B%22color%22%3A%22%23EE4B2B  
%22,%22size%22%3A%22large%22,%22label%22%3A%22BC%20Place,%20Vancouver%22,  
%22text-color%22%3A%22%230000FF%22%7D,%22geometry%22%3A%7B%22coordinates%22%3A  
%5B-123.11210974152168,49.27678753813112%5D,%22type%22%3A%22Point%22%7D%7D%5D  
%7D&key=API_KEY
```

Response image



Add multiple markers to a map image

In this example, you will add markers for must-see places in Helsinki, Finland using their coordinates (longitude, latitude). You can also apply padding to ensure the map accommodates all markers properly.

Note

The available marker icon type include: cp for proximity circle, diamond, pin, poi, square, triangle, bubble.

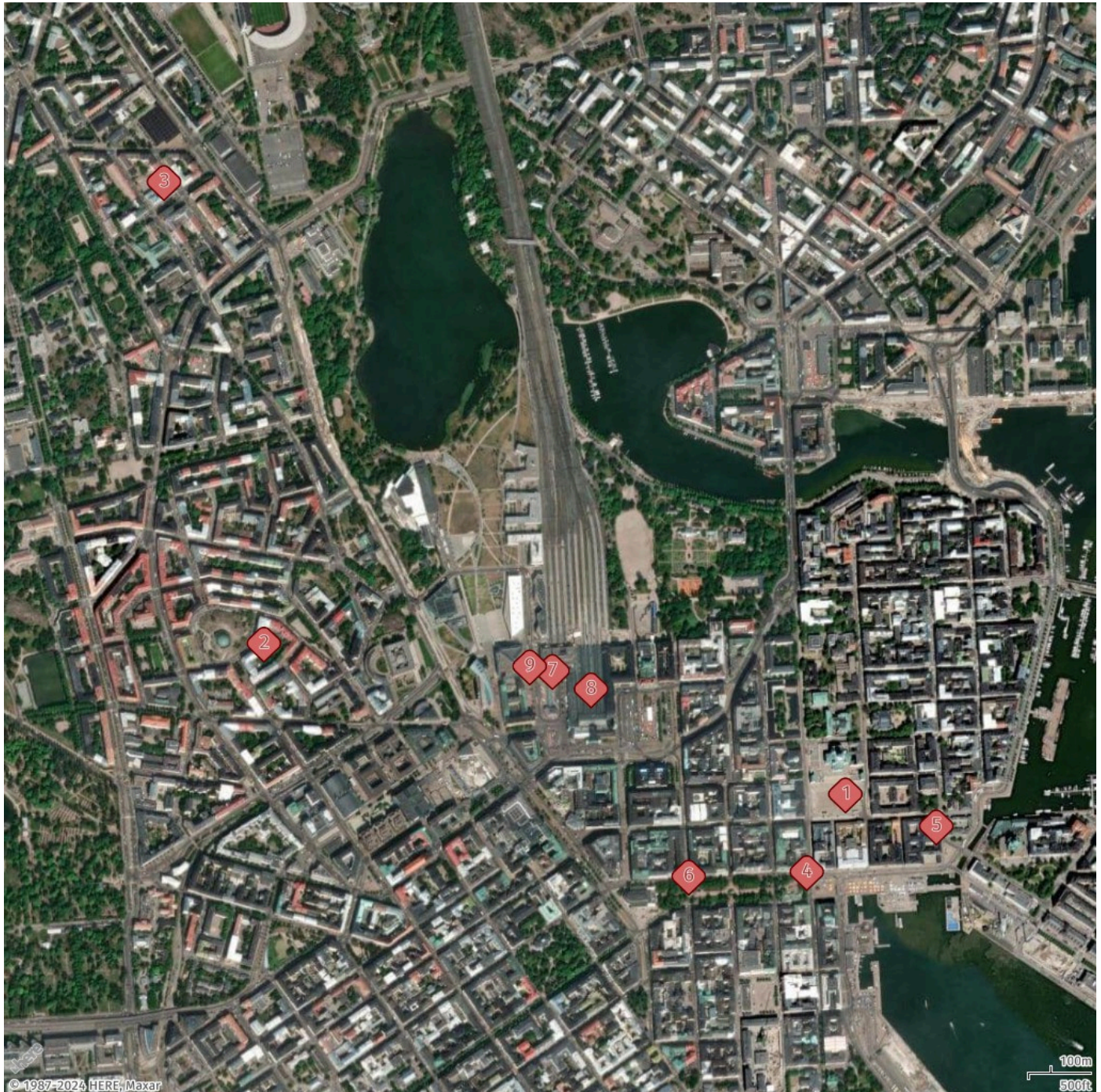
Geo JSON:

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "MultiPoint",
        "coordinates": [
          [24.9526, 60.1692],
          [24.9270, 60.1725],
          [24.9226, 60.1826],
          [24.9509, 60.1675],
          [24.9566, 60.1685],
          [24.9457, 60.1674],
          [24.9397, 60.1719],
          [24.9414, 60.1715],
          [24.9387, 60.1720]
        ]
      },
      "properties": {
        "icon": "diamond",
        "label": "$num",
        "size": "large",
        "text-color": "%23972E2B",
        "text-outline-color": "%23FFF",
        "text-outline-width": 2
      }
    }
  ]
}
```


Request URL

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?
style=Satellite&width=1024&height=1024&padding=150&geojson-overlay=%7B
%22type%22%3A%22FeatureCollection%22,%22features%22%3A%5B%7B%22type%22%3A
%22Feature%22,%22geometry%22%3A%7B%22type%22%3A%22MultiPoint%22,%22coordinates
%22%3A%5B%5B24.9526,60.1692%5D,%5B24.927,60.1725%5D,%5B24.9226,60.1826%5D,
%5B24.9509,60.1675%5D,%5B24.9566,60.1685%5D,%5B24.9457,60.1674%5D,
%5B24.9397,60.1719%5D,%5B24.9414,60.1715%5D,%5B24.9387,60.172%5D%5D%7D,%22properties
%22%3A%7B%22icon%22%3A%22diamond%22,%22label%22%3A%22%24num%22,%22size%22%3A%22large
%22,%22text-color%22%3A%22%23972E2B%22,%22text-outline-color%22%3A%22%23FFF%22,
%22text-outline-width%22%3A%22%7D%7D%5D%7D&key=API_KEY
```

Response image



Change color of marker in a map image

In this example, you will use bubble markers of different colors to represent cities across the world. You can customize the marker's color, label, and other properties to suit the context of your map.

Geo JSON:

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [85.1376, 25.5941]
      },
      "properties": {
        "label": "Patna",
        "icon": "bubble",
        "color": "#FF5722",
        "outline-color": "#D74D3D",
        "text-color": "#FFFFFF"
      }
    },
    .....redacted.....
    {
      "type": "Feature",
      "geometry": {
        "type": "Point",
        "coordinates": [2.3522, 48.8566]
      },
      "properties": {
        "label": "Paris, France",
        "icon": "bubble",
        "color": "#FF9800",
        "outline-color": "#D76B0B",
        "text-color": "#FFFFFF"
      }
    }
  ]
}
```

Request URL

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?
style=Satellite&width=1400&height=1024&padding=150&geojson-overlay=%7B
%22type%22%3A%22FeatureCollection%22,%22features%22%3A%5B%7B%22type%22%3A
%22Feature%22,%22geometry%22%3A%7B%22type%22%3A%22Point%22,%22coordinates
%22%3A%5B85.1376,25.5941%5D%7D,%22properties%22%3A%7B%22label%22%3A%22Patna
%22,%22icon%22%3A%22bubble%22,%22color%22%3A%22%23FF5722%22,%22outline-color
```

```
%22%3A%22%23D74D3D%22,%22text-color%22%3A%22%23FFFFFF%22%7D%7D,%7B%22type%22%3A%22Feature%22,%22geometry%22%3A%7B%22type%22%3A%22Point%22,%22coordinates%22%3A%5B105.8542,21.0285%5D%7D,%22properties%22%3A%7B%22label%22%3A%22Hanoi,%20Vietnam%22,%22icon%22%3A%22bubble%22,%22color%22%3A%22%232196F3%22,%22outline-color%22%3A%22%231A78C2%22,%22text-color%22%3A%22%23FFFFFF%22%7D%7D,%7B%22type%22%3A%22Feature%22,%22geometry%22%3A%7B%22type%22%3A%22Point%22,%22coordinates%22%3A%5B116.4074,39.9042%5D%7D,%22properties%22%3A%7B%22label%22%3A%22Beijing,%20China%22,%22icon%22%3A%22bubble%22,%22color%22%3A%22%23FF9800%22,%22outline-color%22%3A%22%23D76B0B%22,%22text-color%22%3A%22%23FFFFFF%22%7D%7D,%7B%22type%22%3A%22Feature%22,%22geometry%22%3A%7B%22type%22%3A%22Point%22,%22coordinates%22%3A%5B106.9101,47.918%5D%7D,%22properties%22%3A%7B%22label%22%3A%22Ulaanbaatar,%20Mongolia%22,%22icon%22%3A%22bubble%22,%22color%22%3A%22%239C27B0%22,%22outline-color%22%3A%22%237B1FA2%22,%22text-color%22%3A%22%23FFFFFF%22%7D%7D,%7B%22type%22%3A%22Feature%22,%22geometry%22%3A%7B%22type%22%3A%22Point%22,%22coordinates%22%3A%5B151.2093,-33.8688%5D%7D,%22properties%22%3A%7B%22label%22%3A%22Sydney,%20Australia%22,%22icon%22%3A%22bubble%22,%22color%22%3A%22%234CAF50%22,%22outline-color%22%3A%22%23388E3C%22,%22text-color%22%3A%22%23FFFFFF%22%7D%7D,%7B%22type%22%3A%22Feature%22,%22geometry%22%3A%7B%22type%22%3A%22Point%22,%22coordinates%22%3A%5B174.7633,-41.2865%5D%7D,%22properties%22%3A%7B%22label%22%3A%22Wellington,%20New%20Zealand%22,%22icon%22%3A%22bubble%22,%22color%22%3A%22%23FFC107%22,%22outline-color%22%3A%22%23FFA000%22,%22text-color%22%3A%22%23000000%22%7D%7D%5D%7D&key=API_KEY
```


Response image



How to add a line to a static map

In this topic, you'll learn how to add a line to a static map using Amazon Location Service. We'll cover the supported styling options, how to define a line using GeoJSON, and how to apply custom styles such as color, width, and outline. We'll also explore how to use different measurement units for properties like line width.

Supported styling

Name	Type	Default	Description
color	Color	style-dependent	The line color.

Name	Type	Default	Description
width	Integer/String	2	The line width. For more information, see the section called "Add a line to a static map" .
outline-color	Color	style-dependent	The line outline color.
outline-width	Integer/String	0 (turned off)	The width of the outline. For more information, see the section called "Add a line to a static map" .

Supported unit

Unit	Description
Integer, for example, 5	Pixels
String with no unit, for example "5"	Pixels
"px"	Pixels
"m"	Meters
"km"	Kilometers
"mi"	Miles
"ft"	Feet
"yd"	Yards
"%"	The percentage of the default value for a specific property, in pixels. For example, if the default value for width is 2 pixels, then 200%

Unit	Description
	is 4 pixels. % is a sensitive character that must be encoded in the request URL as %25.

Add a line

In this example, you will add a line from a place in Vancouver to Stanley Park. The line is created using a GeoJSON format with defined coordinates.

Request

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [-123.11813, 49.28246],
          [-123.11967, 49.28347],
          [-123.12108, 49.28439],
          [-123.12216, 49.28507],
          [-123.12688, 49.28812],
          [-123.1292, 49.28964],
          [-123.13216, 49.2916],
          [-123.13424, 49.29291],
          [-123.13649, 49.2944],
          [-123.13678, 49.29477],
          [-123.13649, 49.29569],
          [-123.13657, 49.29649],
          [-123.13701, 49.29715],
          [-123.13584, 49.29847],
          [-123.13579, 49.29935],
          [-123.13576, 49.30018],
          [-123.13574, 49.30097]
        ]
      },
      "properties": {
        "name": "To Stanley Park",
        "description": "An evening walk to Stanley Park."
      }
    }
  ]
}
```

```
    }  
  }  
]  
}
```

Request URL

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?  
style=Satellite&width=1024&height=1024&padding=200&scale-  
unit=KilometersMiles&geojson-overlay=%7B%22type%22%3A%22FeatureCollection  
%22,%22features%22%3A%5B%7B%22type%22%3A%22Feature%22,%22geometry%22%3A%7B  
%22type%22%3A%22LineString%22,%22coordinates%22%3A%5B%5B-123.11813,49.28246%5D,  
%5B-123.11967,49.28347%5D,%5B-123.12108,49.28439%5D,%5B-123.12216,49.28507%5D,  
%5B-123.12688,49.28812%5D,%5B-123.1292,49.28964%5D,%5B-123.13216,49.2916%5D,  
%5B-123.13424,49.29291%5D,%5B-123.13649,49.2944%5D,%5B-123.13678,49.29477%5D,  
%5B-123.13649,49.29569%5D,%5B-123.13657,49.29649%5D,%5B-123.13701,49.29715%5D,  
%5B-123.13584,49.29847%5D,%5B-123.13579,49.29935%5D,%5B-123.13576,49.30018%5D,  
%5B-123.13574,49.30097%5D%5D%7D,%22properties%22%3A%7B%22name%22%3A%22To%20Stanley  
%20Park%22,%22description%22%3A%22An%20evening%20walk%20to%20Stanley%20Park.%22%7D  
%7D%5D%7D&key=API_KEY
```


Response image



Add styling to the line

In this example, you will style the line added in the previous example. This includes defining the line's color, width, outline color, and outline width to customize the visual appearance of the line on the map.

Request

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [-123.11813, 49.28246],
          [-123.11967, 49.28347],
          [-123.12108, 49.28439],
          [-123.12216, 49.28507],
          [-123.12688, 49.28812],
          [-123.1292, 49.28964],
          [-123.13216, 49.2916],
          [-123.13424, 49.29291],
          [-123.13649, 49.2944],
          [-123.13678, 49.29477],
          [-123.13649, 49.29569],
          [-123.13657, 49.29649],
          [-123.13701, 49.29715],
          [-123.13584, 49.29847],
          [-123.13579, 49.29935],
          [-123.13576, 49.30018],
          [-123.13574, 49.30097]
        ]
      },
      "properties": {
        "color": "#6d34b3",
        "width": "9m",
        "outline-color": "#a8b9cc",
        "outline-width": "2px"
      }
    }
  ]
}
```

Request URL

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?
style=Satellite&width=1024&height=1024&padding=200&scale-
```

```
unit=KilometersMiles&geojson-overlay=%7B%22type%22%3A%22FeatureCollection%22,%22features%22%3A%5B%7B%22type%22%3A%22Feature%22,%22geometry%22%3A%22LineString%22,%22coordinates%22%3A%5B%5B-123.11813,49.28246%5D,%5B-123.11967,49.28347%5D,%5B-123.12108,49.28439%5D,%5B-123.12216,49.28507%5D,%5B-123.12688,49.28812%5D,%5B-123.1292,49.28964%5D,%5B-123.13216,49.2916%5D,%5B-123.13424,49.29291%5D,%5B-123.13649,49.2944%5D,%5B-123.13678,49.29477%5D,%5B-123.13649,49.29569%5D,%5B-123.13657,49.29649%5D,%5B-123.13701,49.29715%5D,%5B-123.13584,49.29847%5D,%5B-123.13579,49.29935%5D,%5B-123.13576,49.30018%5D,%5B-123.13574,49.30097%5D%5D%7D,%22properties%22%3A%7B%22color%22%3A%22%236d34b3%22,%22width%22%3A%2229m%22,%22outline-color%22%3A%22%23a8b9cc%22,%22outline-width%22%3A%222px%22%7D%7D%5D%7D&key=API_KEY
```


Response image



How to add a route to a static map

In this topic, you'll learn how to add a route to a static map using Amazon Location Service. You'll go through the steps to obtain a route using the `CalculateRoutes` API and then visualize it on a static map using GeoJSON and custom styling for points and lines.

Steps to add a route

1. Get routes from the `CalculateRoutes` API. Remove coordinates that fall on the same straight line to optimize the `LineString`, preventing the query string from reaching its limit.
2. Create a GeoJSON object based on the optimized set of coordinates.
3. Take the first and last points of the `LineString` and add `Point` geometries to mark the start and end locations.
4. Style the points and the `LineString` according to your business needs, adjusting properties like color, size, and labels.

Add a route using compact style

In this example, you will add a route with start and end points to the line created in [the section called "Add a line to a static map"](#). The route will be defined with custom styling, including color, size, and labels for the start and end points.

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [-123.11813, 49.28246],
          [-123.11967, 49.28347],
          [-123.12108, 49.28439],
          [-123.12216, 49.28507],
          [-123.12688, 49.28812],
          [-123.1292, 49.28964],
          [-123.13216, 49.2916],
          [-123.13424, 49.29291],
          [-123.13649, 49.2944],
          [-123.13678, 49.29477],
          [-123.13649, 49.29569],
          [-123.13657, 49.29649],
          [-123.13701, 49.29715],
          [-123.13584, 49.29847],
          [-123.13579, 49.29935],
          [-123.13576, 49.30018],
          [-123.13574, 49.30097]
        ]
      }
    }
  ]
}
```

```
    ]
  },
  "properties": {
    "color": "#000000",
    "width": "20m",
    "outline-color": "#a8b9cc",
    "outline-width": "2px"
  }
},
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [-123.11813, 49.28246]
  },
  "properties": {
    "label": "Pacific Centre",
    "icon": "bubble",
    "size": "large",
    "color": "#34B3A4",
    "outline-color": "#006400",
    "text-color": "#FFFFFF"
  }
},
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [-123.13574, 49.30097]
  },
  "properties": {
    "label": "Stanley Park",
    "icon": "bubble",
    "size": "large",
    "color": "#B3346D",
    "outline-color": "#FF0000",
    "text-color": "#FFFFFF"
  }
}
]
```

Request URL

```
https://maps.geo.eu-central-1.amazonaws.com/v2/static/map?
style=Satellite&width=1024&height=1024&padding=200&scale-
unit=KilometersMiles&geojson-overlay=%7B%22type%22%3A%22FeatureCollection
%22,%22features%22%3A%5B%7B%22type%22%3A%22Feature%22,%22geometry%22%3A%7B
%22type%22%3A%22LineString%22,%22coordinates%22%3A%5B%5B-123.11813,49.28246%5D,
%5B-123.11967,49.28347%5D,%5B-123.12108,49.28439%5D,%5B-123.12216,49.28507%5D,
%5B-123.12688,49.28812%5D,%5B-123.1292,49.28964%5D,%5B-123.13216,49.2916%5D,
%5B-123.13424,49.29291%5D,%5B-123.13649,49.2944%5D,%5B-123.13678,49.29477%5D,
%5B-123.13649,49.29569%5D,%5B-123.13657,49.29649%5D,%5B-123.13701,49.29715%5D,
%5B-123.13584,49.29847%5D,%5B-123.13579,49.29935%5D,%5B-123.13576,49.30018%5D,
%5B-123.13574,49.30097%5D%5D%7D,%22properties%22%3A%7B%22color%22%3A%22%23000000%22,
%22width%22%3A%2220m%22,%22outline-color%22%3A%22%23a8b9cc%22,%22outline-width
%22%3A%222px%22%7D%7D,%7B%22type%22%3A%22Feature%22,%22geometry%22%3A%22Point%22,
%22coordinates%22%3A%5B-123.11813,49.28246%5D%7D,%22properties%22%3A%7B%22label
%22%3A%22Pacific%20Centre%22,%22icon%22%3A%22bubble%22,%22size%22%3A%22large%22,
%22color%22%3A%22%2334B3A4%22,%22outline-color%22%3A%22%23006400%22,%22text-color
%22%3A%22%23FFFFFF%22%7D%7D,%7B%22type%22%3A%22Feature%22,%22geometry%22%3A%22Point
%22,%22coordinates%22%3A%5B-123.13574,49.30097%5D%7D,%22properties%22%3A%7B%22label
%22%3A%22Stanley%20park%22,%22icon%22%3A%22bubble%22,%22size%22%3A%22large%22,
%22color%22%3A%22%23B3346D%22,%22outline-color%22%3A%22%23FF0000%22,%22text-color
%22%3A%22%23FFFFFF%22%7D%7D%5D%7D&key=API_KEY
```

Response image



Manage costs and usage

As you continue learning about Amazon Location maps, it's important to understand how to manage service capacity, ensure you follow usage limits, and get the best results through quota

and API optimizations. By applying best practices for performance and accuracy, you can tailor your application to handle place-related queries efficiently and maximize your API requests.

Topics

- [Best practices for Amazon Location Service](#)
- [Maps pricing](#)
- [Map quotas and usage](#)

Best practices for Amazon Location Service

When working with Amazon Location Service, adhering to best practices ensures your maps are optimized for performance, accuracy, and user experience. This section outlines key considerations for working with static maps, geographical bounds, and GeoJSON data to enhance map functionality and visualization.

Dynamic maps

The following are a few best practices for working with dynamic maps in Amazon Location Service.

Rendering optimization with MapLibre

The following are few features of MapLibre which help optimize rendering for AWS map styles. For more information, see [the section called “Map styles”](#).

Skip validation of style

If you are using the AWS map style, set `validateStyle` to `false`. This will turn off load-time style validation, speeding up the initial map load. Style validation is not needed with AWS map styles, because they are pre-validated.

Example

```
const map = new maplibregl.Map({
  container: 'map', // ID of the div where the map will render
  style: 'https://maps.geo.${awsRegion}.amazonaws.com/v2/styles/${mapStyle}/
descriptor?key=${apiKey}', // Map style URL
  center: [0, 0], // Starting position [lng, lat]
  zoom: 2, // Starting zoom
  validateStyle: false, // Disable style validation for faster map load
});
```


Explanation

- `validateStyle: true`: This enables validation of the map style against the MapLibre GL style specification. If there are any issues in the style, they'll be logged in the console.
- If you set this to `false`, the map will skip the style validation process, which might result in faster loading, but without error checking.

Pre-warm the map

For single-page applications (SPAs) that may create and destroy the map many times as the user navigates through the app, the pre-warm function can reduce delays in re-creating the map after it has been destroyed.

This feature is only recommended for SPAs.

Static maps

Bounds, bounding box (bbox)

When working with maps and geographical data, defining the bounding box (bbox) and bounds parameters accurately is crucial, as they determine the geographic area of interest. Any inaccuracies can lead to undesirable results.

Ensure precise bounds

Ensure the specified bounds precisely represent the region you want to display. Even slight inaccuracies can crop or exclude portions of the desired area, defeating the visualization's purpose.

Verify appropriate zoom level

The map's zoom level is automatically calculated based on the specified bounds or bbox. Verify that the resulting zoom level provides appropriate detail and visibility for the entire area of interest. If the zoom is too high or low, the map may fail to convey the desired information effectively.

Check for custom overlay visibility

When using bbox or bounds with custom overlays like GeoJSON features, make sure the features' extent falls within the resulting map image. Features extending beyond the bounds may be clipped or omitted, leading to incomplete or misleading visualizations.

Use padding with bbox

Use the bbox along with the padding parameter to ensure map features near the edges are fully visible and not cut off.

By accurately defining the bbox and bounds parameters, you can ensure your maps represent the desired geographic area correctly, provide an appropriate level of detail, and effectively incorporate custom overlays or data layers.

GeoJSON

When using GeoJSON data, optimizing the query string by minifying the GeoJSON can help you stay within query string limits, especially for large datasets.

Maps pricing

Amazon Location Service offers competitive pricing for its Maps API based on the type of map request and the number of API calls made. This section provides an overview of the pricing structure for dynamic and static maps.

For detailed pricing information, see [Amazon Location Service pricing](#).

Dynamic maps

Pricing for the Maps API is based on the number of requests made to the GetTiles API.

Other map-related APIs, such as GetGlyphs, GetStyleDescriptor, and GetSprites are free of charge.

Static map images

Pricing for static map images is based on the number of requests made to the GetStaticMap API. Each request for a static map image is counted towards the pricing calculation.

Map quotas and usage

Amazon Location Service imposes specific service quotas and usage limits for both dynamic and static maps. These limits are put in place to ensure fair usage and performance efficiency across all users. Below are the service quotas and adjustable limits for each service.

Service quotas

Amazon Location Service sets default quotas for APIs to help manage service capacity, which can be viewed in the [AWS service quotas management console](#). You can request an increase in quotas through the [self-service console](#), for up to twice the default limit for each API.

For quota limits exceeding twice the default limit, request through the self service console and it will automatically submit a support ticket. Alternately, connect with your premium support team.

There are no direct charges for quota increase requests, but higher usage levels may lead to increased service costs based on the additional resources consumed. For more information, see [the section called "Manage quotas"](#).

Dynamic map

API name	Default	Max adjustable limit	More than Adjustable Max limit
GetTiles	2000	4000	Request on service quota console or contact support team

Static map

API name	Default	Max adjustable limit	More than Adjustable Max limit
GetStaticMap	50	100	Request on service quota console or contact support team

Usage limits

API name	Limit	Value
GetStyleDescriptor	Max requests, per second, per IP address.	5000

API name	Limit	Value
GetGlyphys	Max requests, per second, per IP address.	5000
GetSprites	Max requests, per second, per IP address.	5000
GetStaticMap	Response payload size after compression.	6MB
GetTiles	Response payload size after compression.	6MB

Terms

For more information, see [the section called “Terms of use and data attribution”](#).

Amazon Location Service Places



With Amazon Location Places, you can add location-based capabilities to your application. Using Places APIs, you can search or geocode locations by querying a comprehensive place database containing over 400 million addresses and points of interest (POIs) across 108 countries.

What is a Place?

A place refers to any specific location, including administrative areas, addresses, POIs, geographic areas, and more. Places have various associated information such as name, address, coordinates, type, and details like business hours, contacts, access points, and POI categories.

- **Address:** Includes specific point addresses (like offices, homes, apartments), street addresses, and interpolated addresses.
- **Points of Interest (POI):** Includes businesses (like restaurants, stores) and landmarks (like parks, monuments).
- **Administrative Areas:** Includes regions like countries, states, provinces, districts, and postal areas.
- **Geographic Areas:** Includes locations like cities, neighborhoods, and localities.

Features

Search APIs enable users to discover places, points of interest (POIs), and addresses by querying based on names, categories, or locations.

- **Geocode APIs:** Convert physical locations or addresses into geographic coordinates (longitude and latitude).
- **Intended use:** Places APIs allow users to retrieve and utilize location data for various applications, from one-time uses to storing data for future reference. By default, data is retrieved for single-use, but options are available to store data persistently for analysis or further use.

Use cases

Enhance your checkout experience

Provide real-time suggestions for valid street addresses as customers enter their location on websites or apps. This helps ensure accurate delivery or pickup locations, reducing errors and creating a seamless checkout experience. Enhance cart completion rates by validating addresses to improve user satisfaction and drive conversions.

For more information, refer to the related APIs: Autocomplete and Suggest.

Discover local places

Assist customers in finding the nearest, most relevant places. Real estate clients, for example, can locate nearby schools, grocery stores, and parks. Build lists of target businesses to market services and gain insights by searching by category and retrieving essential information like addresses and geocodes.

For more information, refer to the related APIs: SearchNearby and SearchText.

Provide customers with up-to-date business details

Deliver the latest information on local businesses. Enable users to search by business name or category within a specific area, enhancing customer experience and supporting business decision-making. For example, help gym members locate health food stores nearby, or ensure delivery drivers arrive during open hours.

For more information, refer to the related APIs: SearchNearby, SearchText, Suggest, and GetPlace.

Enrich customer addresses with detailed insights

Enhance existing customer address data with additional insights like postal codes, geo-coordinates, contact details, and business categories. Use the perpetual storage option to save this data in your database for strategic use in marketing and decision-making.

For more information, refer to the related APIs: Geocode, Reverse Geocode, SearchText, and SearchNearby.

Find geospatial coordinates of addresses to calculate routes and driving directions

Convert addresses or business names into geospatial coordinates and calculate optimal routes for tasks like food delivery or rideshare services. Use waypoint sequencing to create efficient paths for multiple stops.

For more information, refer to the related APIs: Suggest and SearchText.

Identify time zones for locations

Determine the correct time zone for specific locations (city, address, coordinates, or place ID). This feature is ideal for travel applications that require accurate time zones for itineraries and scheduling applications. Ensure that billing and compliance records use the appropriate timestamps.

For more information, refer to the related APIs: Geocode, Reverse Geocode, SearchText, and GetPlace.

Places APIs overview

API name	Description	Additional resources
Autocomplete	Autocomplete completes potential search places or addresses as the user types, based on the partial input. Autocomplete is not available in Japan.	the section called "Autocomplete"
Geocode	Geocode or forward geocode is converting a textual address or place into geographic coordinates.	the section called "Geocode"
Reverse Geocode	Reverse geocode is converting geographic coordinates into a human-readable address or place.	the section called "Reverse Geocode"
GetPlace	Get Place retrieves detailed information about a specific place, such as its address, opening hours, contacts, and other metadata.	the section called "GetPlace"
Suggest	Suggest provides intelligent predictions or recommendations based on the user's input or context, such as relevant places, points of interest or query term	the section called "Suggest"

API name	Description	Additional resources
Search Text	Search Text provides an ability to search for places, addresses or point of interest using textual input, and it returns information such as a place name, address, phone, category, food type, contact, opening hours.	the section called "Search Text"
Search Nearby	Search nearby provides an ability to search for points of interest within a specified radius or distance from a geographic coordinates.	the section called "Search Nearby"

Places concepts

The following topics provide foundational knowledge and best practices for using the Amazon Location Service Places API. These concepts are essential for effectively implementing the Places features within your application.

Topics

- [Places terminology](#)
- [Querying and biasing](#)
- [Filtering](#)
- [Localization and internationalization](#)
- [Contacts and opening hours](#)
- [Additional features](#)
- [IntendedUse](#)

Places terminology

This section provides essential definitions to understand core concepts within Amazon Location Service Places, such as location data types, geographic boundaries, and filtering options. These terms enable accurate use of the Places API capabilities.

Location

A specific point on Earth's surface, typically defined by geographic coordinates (longitude and latitude). Locations can represent any place or area, such as a city, building, or point of interest.

Places

Any location that includes administrative areas, addresses, points of interest (POI), geographic areas, and more. Places often have associated information, such as name, address, coordinates, types, business hours, contacts, and categories.

Address

Includes point-based addresses (like offices, homes), street addresses, and interpolated addresses, providing precise location data.

See the definition of *interpolation* below.

Point of Interest (POI)

A POI refers to notable locations, such as businesses (like restaurants, stores) or landmarks (like parks, monuments).

Administrative areas

Regions such as countries, provinces, states, districts, blocks, and postal areas that organize geographical data.

Geographic area

Areas including cities, localities, and neighborhoods, offering additional levels of granularity in geographical data.

Position

A precise set of coordinates (longitude and latitude) that pinpoints where a place is located on a map.

Bias position

A reference point in geographic data that helps prioritize search results closer to this location, enhancing relevance in searches.

Bounding box

A rectangular geographic area defined by southwest and northeast coordinates, used to narrow down searches or display content within the area on a map.

Place type

A classification for places based on function or characteristics. Types include country, region, locality, district, postal area, block, intersection, street address, point address, interpolated address, or POI.

Category

A grouping for businesses and landmarks based on the type of services or activities they offer, such as restaurants, hotels, schools, and parks. Categories make it easier for users to find specific types of POIs in searches.

Match score

An indicator of how closely a search result aligns with the input, aiding in determining relevance.

Interpolation

The method of estimating unknown addresses by using known address locations as reference points.

ISO 3166 country codes

Amazon Location Service Places use International Organization for Standardization (ISO) 3166 country codes for identifying countries or regions. Find the code for each country on the ISO Online Browsing Platform.

Querying and biasing

Amazon Location Service Places API offers querying and biasing options to retrieve and search location data.

Querying

A query refers to the input parameters used to retrieve and search location data. The way APIs returns results is determined by these queries types.

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
QueryText	Yes	No	Yes	N/A	Yes	No	Yes

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
Query component	Yes	No	No	N/A	No	No	No
Query Position	No	Yes	No	N/A	No	Yes	No
Query Radius	No	Yes	No	N/A	No	Yes	No
Query Id	No	No	No	N/A	No	No	No
Place Id	No	No	No	Yes	No	No	No

Biasing

The "bias position" is a location that influences search results, giving priority to places near biased position. It doesn't restrict results, but biases them toward the specified area. This feature prioritizes relevant location results when multiple places have similar names.

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
BiasPosition	Yes	No	Yes	N/A	Yes	No	Yes

Filtering

Amazon Location Service Places API offers filtering options to narrow down search results based on specific criteria. These parameters refine results by including or excluding criteria such as geographic constraints, administrative areas, business chains, POI categories, and food types.

This section details filtering parameters and options available in Amazon Location Service Places API. Each filter is explained below, including its impact on search results.

Geographic filters

Geographic filters restrict search results within specified geographic boundaries, such as circles or bounding boxes. This allows users to target specific areas in their search queries.

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
Circle	No	No	Yes	N/A	Yes	No	Yes
Bounding Box	No	No	Yes	N/A	Yes	Yes	Yes

Administrative area filters

These filters allow users to limit search results to specific administrative areas, such as countries or place types.

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
Include Countries	Yes	No	Yes	N/A	Yes	Yes	Yes
Include Place Types	Yes	Yes	Yes	N/A	No	No	No

POI category filters

POI category filters refine search results based on Points of Interest (POI) categories, such as specific business chains or food types.

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
Include Categories	No	No	No	N/A	No	Yes	No
Exclude Categories	No	No	No	N/A	No	Yes	No
Include Business Chains	No	No	No	N/A	No	Yes	No
Exclude Business Chains	No	No	No	N/A	No	Yes	No
Include Food Type	No	No	No	N/A	No	Yes	No
Exclude Food Type	No	No	No	N/A	No	Yes	No

Definitions of filters

Filtering for place types can be performed using the following criteria: FoodType, Category, and BusinessChainId.

Circle

A circular geographic area defined by a center point (longitude and latitude) and a radius. This filter restricts results to locations within the circle.

Bounding box

A rectangular geographic region defined by two sets of coordinates (southwest and northeast corners), restricting results to locations within this box.

Include countries

Limits results to specific countries, allowing only locations within the specified countries to be included.

Include place types

Limits results to specific types of places, such as country, city, or neighborhood.

Include and exclude categories

Refines results based on specific POI categories, such as restaurants or shops. You can include or exclude categories to adjust search results.

Include and exclude business chains

Refines results based on specific business chains, such as Starbucks or McDonald's. You can include or exclude business chains in your search criteria.

Include and exclude food type

Filters results by specific types of food-related places, such as Italian or Chinese cuisine. You can include or exclude food types as per your requirements.

Place type filters

These place types provide different levels of granularity in geographic searches, allowing users to refine results to return only specific types, such as country, region, street, or individual address level. Amazon Location Service supports the following Places types:

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
PostalCode	Yes	No	Yes	N/A	No	No	No
Locality	Yes	Yes	Yes	N/A	No	No	No

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
Intersection	Yes	No	No	N/A	No	No	No
Street	Yes	Yes	No	N/A	No	No	No
PointAddress	Yes	Yes	No	N/A	No	No	No
InterpolatedAddresses	Yes	Yes	No	N/A	No	No	No

Categories filters

This section details the various point of interest (POI) categories available within Amazon Location Service. These categories enable users to filter search results to specific types of places, enhancing relevance and user experience for location-based applications.

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
Categories	In Response only	In Response only	Not available	In Response only	In Response only	In Request and Response	In Response only

Amazon Location Service supports a variety of place categories, which allow applications to filter POIs according to user needs. The categories enhance search precision by narrowing down results based on the type of location, such as restaurants, hospitals, or schools.

To retrieve information on place categories, use the relevant Place API with filtering parameters for categories. For more information, refer to the Amazon Location Service [API documentation](#).

Category name	Category ID
Adult Entertainment	adult_entertainment
Adult Shop	adult_shop
Advertising-Marketing, PR and Market Research	advertising-marketing,_pr_and_market_research
Aerial Tramway	aerial_tramway
Airport	airport
Airport Cargo	airport_cargo
Airport Terminal	airport_terminal
Ambulance Services	ambulance_services
Amusement Park	amusement_park
Animal Park	animal_park
Apartment Rental-Flat Rental	apartment_rental-flat_rental
Aquarium	aquarium
Art Museum	art_museum
Arts and Crafts Supplies	arts_and_crafts_supplies
Ashram	ashram
ATM	atm
Attorney	attorney
Auto Parts	auto_parts
Automobile Club	automobile_club
Automobile Dealership-New Cars	automobile_dealership-new_cars

Category name	Category ID
Automobile Dealership-Used Cars	automobile_dealership-used_cars
Aviation	aviation
B2B Restaurant Services	b2b_restaurant_services
B2B Sales and Services	b2b_sales_and_services
Badminton	badminton
Bakery and Baked Goods Store	bakery_and_baked_goods_store
Bank	bank
Banquet Hall	banquet_hall
Bar or Pub	bar_or_pub
Barber	barber
Basketball	basketball
Bay-Harbor	bay-harbor
Beach	beach
Bed and Breakfast	bed_and_breakfast
Beer Garden	beer_garden
Bicycle and Bicycle Accessories Shop	bicycle_and_bicycle_accessories_shop
Bicycle Parking	bicycle_parking
Bicycle Service	bicycle_service
Bicycle Service and Maintenance	bicycle_service_and_maintenance
Bicycle Sharing Location	bicycle_sharing_location

Category name	Category ID
Bike Park	bike_park
Bill Payment Service	bill_payment_service
Billiards-Pool Hall	billiards-pool_hall
Bistro	bistro
Blood Bank	blood_bank
BMX Shop	bmx_shop
BMX Track	bmx_track
Boat Ferry	boat_ferry
Boating	boating
Body of Water	body_of_water
Body Piercing and Tattoos	body_piercing_and_tattoos
Bookstore	bookstore
Border Crossing	border_crossing
Bowling Center	bowling_center
Brewery	brewery
Building	building
Bus Rapid Transit	bus_rapid_transit
Bus Station	bus_station
Bus Stop	bus_stop
Business Facility	business_facility

Category name	Category ID
Business Service	business_service
Butcher	butcher
Cafeteria	cafeteria
Campground	campground
Camping-Hiking Shop	camping-hiking_shop
Campsite	campsite
Canal	canal
Canoe-Kayak Shop	canoe-kayak_shop
Car Repair	car_repair
Car Repair-Service	car_repair-service
Car Wash-Detailing	car_wash-detailing
Cargo Center	cargo_center
Cargo Transportation	cargo_transportation
Carshare Location	carshare_location
Casino	casino
Castle	castle
Casual Dining	casual_dining
Catering and Other Food Services	catering_and_other_food_services
Cellphone Parking Lot	cellphone_parking_lot
Cemetery	cemetery

Category name	Category ID
Check Cashing Service-Currency Exchange	check_cashing_service-currency_exchange
Children's Apparel	children's_apparel
Children's Museum	children's_museum
Chiropractor	chiropractor
Church	church
Cigar and Tobacco Shop	cigar_and_tobacco_shop
Cinema	cinema
City Hall	city_hall
Civic-Community Center	civic-community_center
Clothing and Accessories	clothing_and_accessories
Clubhouse	clubhouse
Coaching Institute	coaching_institute
Cocktail Lounge	cocktail_lounge
Coffee Shop	coffee_shop
Coffee-Tea	coffee-tea
Collective Community	collective_community
Commercial Services	commercial_services
Communication-Media	communication-media
Commuter Rail Station	commuter_rail_station
Commuter Train	commuter_train

Category name	Category ID
Complete Rest Area	complete_rest_area
Computer and Software	computer_and_software
Construction	construction
Consumer Electronics Store	consumer_electronics_store
Consumer Goods	consumer_goods
Consumer Services	consumer_services
Convenience Store	convenience_store
Convention-Exhibition Center	convention-exhibition_center
County Council	county_council
Couriers	couriers
Courthouse	courthouse
COVID-19 Testing Site	covid-19_testing_site
Crematorium	crematorium
Cross Country Ski Shop	cross_country_ski_shop
Customer Care-Service Center	customer_care-service_center
Dairy Goods	dairy_goods
Dancing	dancing
Deli	deli
Delivery Entrance	delivery_entrance
Dentist-Dental Office	dentist-dental_office

Category name	Category ID
Department Store	department_store
Discount Store	discount_store
Distillery	distillery
Diving Center	diving_center
Doughnut Shop	doughnut_shop
Drugstore	drugstore
Drugstore or Pharmacy	drugstore_or_pharmacy
Dry Cleaning and Laundry	dry_cleaning_and_laundry
Education Facility	education_facility
Electrical	electrical
Embassy	embassy
Emission Testing	emission_testing
Engineering and Scientific Services	engineering_and_scientific_services
Entertainment and Recreation	entertainment_and_recreation
Entertainment Electronics	entertainment_electronics
EV Battery Swap Station	ev_battery_swap_station
EV Charging Station	ev_charging_station
Event Spaces	event_spaces
Facilities	facilities
Family Restaurant	family_restaurant

Category name	Category ID
Family-General Practice Physicians	family-general_practice_physicians
Farming	farming
Fast Food	fast_food
Ferry Terminal	ferry_terminal
Finance and Insurance	finance_and_insurance
Financial Investment Firm	financial_investment_firm
Fine Arts	fine_arts
Fine Dining	fine_dining
Fire Department	fire_department
Fitness-Health Club	fitness-health_club
Floor and Carpet	floor_and_carpet
Florist	florist
Flowers and Jewelry	flowers_and_jewelry
Food Market-Stall	food_market-stall
Food Production	food_production
Food-Beverage Specialty Store	food-beverage_specialty_store
Forest, Heath or Other Vegetation	forest,_heath_or_other_vegetation
Fueling Station	fueling_station
Fulfillment and Distribution Center	fulfillment_and_distribution_center
Funeral Director	funeral_director

Category name	Category ID
Furniture Store	furniture_store
Gallery	gallery
Gambling-Lottery-Betting	gambling-lottery-betting
Garden	garden
Garden Center	garden_center
General Merchandise	general_merchandise
Gift, Antique and Art	gift,_antique_and_art
Glass and Window	glass_and_window
Golf Course	golf_course
Golf Practice Range	golf_practice_range
Golf Shop	golf_shop
Government Office	government_office
Government or Community Facility	government_or_community_facility
Grocery	grocery
Guest House	guest_house
Gurdwara	gurdwara
Hair and Beauty	hair_and_beauty
Hair Salon	hair_salon
Hardware, House and Garden	hardware,_house_and_garden
Healthcare and Healthcare Support Services	healthcare_and_healthcare_support_services

Category name	Category ID
Higher Education	higher_education
Highway Exit	highway_exit
Historical Monument	historical_monument
History Museum	history_museum
Hockey	hockey
Holiday Park	holiday_park
Home Improvement	home_improvement
Home Specialty Store	home_specialty_store
Hospital	hospital
Hospital Emergency Room	hospital_emergency_room
Hospital or Health Care Facility	hospital_or_health_care_facility
Hostel	hostel
Hot Spring	hot_spring
Hotel	hotel
Hotel or Motel	hotel_or_motel
Human Resources and Recruiting Services	human_resources_and_recruiting_services
Hunting-Fishing Shop	hunting-fishing_shop
Hydrogen Fuel Station	hydrogen_fuel_station
Ice Skating Rink	ice_skating_rink
Inclined Rail	inclined_rail

Category name	Category ID
Indoor Ski	indoor_ski
Indoor Sports	indoor_sports
Industrial Zone	industrial_zone
Interior and Exterior Design	interior_and_exterior_design
Internet Cafe	internet_cafe
Investigation Services	investigation_services
Island	island
IT and Office Equipment Services	it_and_office_equipment_services
Jazz Club	jazz_club
Jeweler	jeweler
Karaoke	karaoke
Kindergarten and Childcare	kindergarten_and_childcare
Lake	lake
Landmark-Attraction	landmark-attraction
Landscaping Services	landscaping_services
Language Studies	language_studies
Legal Services	legal_services
Leisure	leisure
Library	library
Lightrail	lightrail

Category name	Category ID
Live Entertainment-Music	live_entertainment-music
Loading Dock	loading_dock
Loading Zone	loading_zone
Local Transit	local_transit
Locksmiths and Security Systems Services	locksmiths_and_security_systems_services
Lodging	lodging
Lottery Booth	lottery_booth
Lumber	lumber
Maid Services	maid_services
Major Appliance	major_appliance
Management and Consulting Services	management_and_consulting_services
Manufacturing	manufacturing
Marina	marina
Market	market
Marriage and Match Making Services	marriage_and_match_making_services
Medical Services-Clinics	medical_services-clinics
Meeting Point	meeting_point
Men's Apparel	men's_apparel
Military Base	military_base
Mining, Quarrying and Other Extraction	mining,_quarrying_and_other_extraction

Category name	Category ID
Mobile Retailer	mobile_retailer
Mobile Service Center	mobile_service_center
Modeling Agencies	modeling_agencies
Money Transferring Service	money_transferring_service
Monorail	monorail
Mosque	mosque
Motel	motel
Motorcycle Accessories	motorcycle_accessories
Motorcycle Dealership	motorcycle_dealership
Motorcycle Service and Maintenance	motorcycle_service_and_maintenance
Motorcycle, Moped and Scooter Parking	motorcycle,_moped_and_scooter_parking
Motorway Service Rest Area	motorway_service_rest_area
Mountain or Hill	mountain_or_hill
Mountain Passes	mountain_passes
Mountain Peaks	mountain_peaks
Mover	mover
Museum	museum
Nail Salon	nail_salon
Named Intersection-Chowk	named_intersection-chowk
Natural and Geographical	natural_and_geographical

Category name	Category ID
Night Club	night_club
Nightlife-Entertainment	nightlife-entertainment
Non-Store Retailers	non-store_retailers
Nursing Home	nursing_home
Off Road Trailhead	off_road_trailhead
Off-Road Vehicle Area	off-road_vehicle_area
Office Supply and Services Store	office_supply_and_services_store
Optical	optical
Organizations and Societies	organizations_and_societies
Other Bookshop	other_bookshop
Other Library	other_library
Other Place of Worship	other_place_of_worship
Outdoor Area-Complex	outdoor_area-complex
Outdoor Service	outdoor_service
Outdoor-Recreation	outdoor-recreation
Pagoda	pagoda
Paint Store	paint_store
Park and Ride	park_and_ride
Park-Recreation Area	park-recreation_area
Parking	parking

Category name	Category ID
Parking and Restroom Only Rest Area	parking_and_restroom_only_rest_area
Parking Garage-Parking House	parking_garage-parking_house
Parking Lot	parking_lot
Parking Only Rest Area	parking_only_rest_area
Pawnshop	pawnshop
Performing Arts	performing_arts
Pet Care	pet_care
Pet Supply	pet_supply
Petrol-Gasoline Station	petrol-gasoline_station
Pharmacy	pharmacy
Photography	photography
Plumbing	plumbing
Police Box	police_box
Police Services-Security	police_services-security
Police Station	police_station
Post Office	post_office
Postal Collection Box	postal_collection_box
Power Equipment Dealer	power_equipment_dealer
Primary School	primary_school
Printing and Publishing	printing_and_publishing

Category name	Category ID
Property Management	property_management
Psychiatric Institute	psychiatric_institute
Public Administration	public_administration
Public Restroom-Toilets	public_restroom-toilets
Public Sports Airport	public_sports_airport
Public Transit Access	public_transit_access
Race Track	race_track
Racquetball Court	racquetball_court
Rail Ferry	rail_ferry
Rail Yard	rail_yard
Ranger Station	ranger_station
Real Estate Services	real_estate_services
Record, CD and Video	record,_cd_and_video
Recreation Center	recreation_center
Recycling Center	recycling_center
Registration Office	registration_office
Religious Place	religious_place
Rental and Leasing	rental_and_leasing
Rental Car Agency	rental_car_agency
Repair and Maintenance Services	repair_and_maintenance_services

Category name	Category ID
Repair Service	repair_service
Reservoir	reservoir
Residential Area-Building	residential_area-building
Rest Area	rest_area
Restaurant	restaurant
Rideshare Pickup	rideshare_pickup
River	river
Road Assistance	road_assistance
Roadside Station	roadside_station
Rugby	rugby
Running Track	running_track
Running-Walking Shop	running-walking_shop
RV Parks	rv_parks
Ryokan	ryokan
Scenic Overlook Rest Area	scenic_overlook_rest_area
Scenic Point	scenic_point
School	school
Science Museum	science_museum
Seaport-Harbour	seaport-harbour
Secondary School	secondary_school

Category name	Category ID
Shoes-Footwear	shoes-footwear
Shooting Range	shooting_range
Shopping Mall	shopping_mall
Short-Time Motel	short-time_motel
Shrine	shrine
Skate Shop	skate_shop
Ski Lift	ski_lift
Ski Resort	ski_resort
Ski Shop	ski_shop
Snowboard Shop	snowboard_shop
Soccer Club	soccer_club
Social Service	social_service
Specialty Clothing Store	specialty_clothing_store
Specialty Food Store	specialty_food_store
Specialty Store	specialty_store
Specialty Trade Contractors	specialty_trade_contractors
Sporting Goods Store	sporting_goods_store
Sporting Instruction and Camps	sporting_instruction_and_camps
Sports Activities	sports_activities
Sports Complex-Stadium	sports_complex-stadium

Category name	Category ID
Sports Facility-Venue	sports_facility-venue
Sports Field	sports_field
Squash Court	squash_court
Storage	storage
Student Housing	student_housing
Surf Shop	surf_shop
Sweet Shop	sweet_shop
Swimming Pool	swimming_pool
Synagogue	synagogue
Tack Shop	tack_shop
Tailor and Alteration	tailor_and_alteration
Take Out and Delivery Only	take_out_and_delivery_only
Tanning Salon	tanning_salon
Taqueria	taqueria
Tax Service	tax_service
Taxi Stand	taxi_stand
Tea House	tea_house
Telephone Service	telephone_service
Temple	temple
Tennis Court	tennis_court

Category name	Category ID
Theatre, Music and Culture	theatre,_music_and_culture
Therapist	therapist
Tire Repair	tire_repair
Tollbooth	tollbooth
Tourist Attraction	tourist_attraction
Tourist Information	tourist_information
Towing Service	towing_service
Toy Store	toy_store
Trailhead	trailhead
Train Station	train_station
Training and Development	training_and_development
Translation and Interpretation Services	translation_and_interpretation_services
Transportation Service	transportation_service
Travel Agent-Ticketing	travel_agent-ticketing
Truck Dealership	truck_dealership
Truck Parking	truck_parking
Truck Repair	truck_repair
Truck Stop-Plaza	truck_stop-plaza
Truck Wash	truck_wash
Truck-Semi Dealer-Services	truck-semi_dealer-services

Category name	Category ID
Underground Train-Subway	underground_train-subway
Undersea Feature	undersea_feature
Urgent Care Center	urgent_care_center
Used-Second Hand Merchandise Stores	used-second_hand_merchandise_stores
Utilities	utilities
Vaccination Site	vaccination_site
Van Repair	van_repair
Vaping Store	vaping_store
Variety Store	variety_store
Veterinarian	veterinarian
Video and Game Rental	video_and_game_rental
Video Arcade-Game Room	video_arcade-game_room
Warehouse	warehouse
Waste and Sanitary	waste_and_sanitary
Water Park	water_park
Water Transit	water_transit
Waterfall	waterfall
Wedding Services and Bridal Studio	wedding_services_and_bridal_studio
Weigh Station	weigh_station
Wellness Center and Services	wellness_center_and_services

Category name	Category ID
Wholesale Store	wholesale_store
Wild Animal Park	wild_animal_park
Wildlife Refuge	wildlife_refuge
Wine and Liquor	wine_and_liquor
Winery	winery
Women's Apparel	women's_apparel
Zoo	zoo

Food Type filters

This section provides an overview of supported food type filters in Amazon Location Service. These filters allow you to narrow down search results for places, focusing specifically on types of cuisine or food-related points of interest (POIs), enhancing location-based searches for users seeking specific dining options.

The food type filter feature in Amazon Location Service helps refine search results by specific types of cuisine or food establishments, supporting user preferences for targeted location searches. Below are the supported food types and their IDs.

To apply food type filters, use the Place APIs with the `foodType` parameter set to the desired cuisine. For detailed guidance on implementing these filters, see the Amazon Location Service [API documentation](#).

Name	FoodType ID
American	american
American-Barbecue/Southern	american-barbecue/southern
American-Cajun	american-cajun
American-Californian	american-californian

Name	FoodType ID
American-Creole	american-creole
American-Native American	american-native_american
American-Soul Food	american-soul_food
American-Southern	american-southern
American-Southwestern	american-southwestern
Argentinean	argentinean
Armenian	armenian
Asian	asian
Australian	australian
Austrian	austrian
Azerbaijani	azerbaijani
Balkan	balkan
Baltic	baltic
Belgian	belgian
Belorussian	belorussian
Bistro	bistro
Bohemian	bohemian
Brazilian	brazilian
Brazilian-Baiana	brazilian-baiana
Brazilian-Bakery	brazilian-bakery

Name	FoodType ID
Brazilian-Capixaba	brazilian-capixaba
Brazilian-Mineira	brazilian-mineira
Breakfast	breakfast
BrewPub	brewpub
British Isles	british_isles
Brunch	brunch
Bruneian	bruneian
Burgers	burgers
Burmese	burmese
Cambodian	cambodian
Canadian	canadian
Caribbean	caribbean
Caucasian	caucasian
Chicken	chicken
Chilean	chilean
Chinese	chinese
Chinese-Beijing	chinese-beijing
Chinese-Cantonese	chinese-cantonese
Chinese-Guangxi	chinese-guangxi
Chinese-Hot Pot	chinese-hot_pot

Name	FoodType ID
Chinese-Hunan/Hubei	chinese-hunan/hubei
Chinese-Inner Mongolian	chinese-inner_mongolian
Chinese-Islamic	chinese-islamic
Chinese-Jiangsu/Zhejiang	chinese-jiangsu/zhejiang
Chinese-Jiangxi	chinese-jiangxi
Chinese-Northeastern	chinese-northeastern
Chinese-Northwestern	chinese-northwestern
Chinese-Porridge	chinese-porridge
Chinese-Shandong	chinese-shandong
Chinese-Shanghai	chinese-shanghai
Chinese-Szechuan	chinese-szechuan
Chinese-Taiwanese	chinese-taiwanese
Chinese-Yunnan/Guizhou	chinese-yunnan/guizhou
Continental	continental
Creperie	creperie
Cuban	cuban
Danish	danish
Dinner	dinner
Dutch	dutch
East European	east_european

Name	FoodType ID
Egyptian	egyptian
Ethiopian	ethiopian
European	european
Fast Food	fast_food
Filipino	filipino
Finnish	finnish
Fondue	fondue
French	french
French-Alsatian	french-alsatian
French-Auvergnate	french-auvergnate
French-Basque	french-basque
French-Corse	french-corse
French-Lyonnaise	french-lyonnaise
French-Provencale	french-provencale
French-Sud-ouest	french-sud-ouest
Fusion	fusion
German	german
Greek	greek
Grill	grill
Halal	halal

Name	FoodType ID
Hawaiian/Polynesian	hawaiian/polynesian
Hot Dogs	hot_dogs
Hungarian	hungarian
Ice Cream	ice_cream
Indian	indian
Indian-Bengali	indian-bengali
Indian-Goan	indian-goan
Indian-Gujarati	indian-gujarati
Indian-Hyderabadi	indian-hyderabadi
Indian-Jain	indian-jain
Indian-Konkani	indian-konkani
Indian-Maharashtrian	indian-maharashtrian
Indian-Malvani	indian-malvani
Indian-Mughlai	indian-mughlai
Indian-North Indian	indian-north_indian
Indian-Parsi	indian-parsi
Indian-Punjabi	indian-punjabi
Indian-Rajasthani	indian-rajasthani
Indian-South Indian	indian-south_indian
Indian-Tandoori	indian-tandoori

Name	FoodType ID
Indonesian	indonesian
International	international
Irish	irish
Italian	italian
Japanese	japanese
Japanese-Chanko	japanese-chanko
Japanese-Curry	japanese-curry
Japanese-Fish/Other Seafood	japanese-fish/other_seafood
Japanese-Gyoza	japanese-gyoza
Japanese-Hibachi	japanese-hibachi
Japanese-Izakaya	japanese-izakaya
Japanese-Jingisukan	japanese-jingisukan
Japanese-Kaiseki	japanese-kaiseki
Japanese-Okonomiyaki	japanese-okonomiyaki
Japanese-Ramen	japanese-ramen
Japanese-Rice Bowl	japanese-rice_bowl
Japanese-Shabushabu	japanese-shabushabu
Japanese-Sukiyaki	japanese-sukiyaki
Japanese-Sushi	japanese-sushi
Japanese-Sushi Train	japanese-sushi_train

Name	FoodType ID
Japanese-Takoyaki	japanese-takoyaki
Japanese-Tempura/Other Fried Food	japanese-tempura/other_fried_food
Japanese-Tonkatsu	japanese-tonkatsu
Japanese-Udon/Other Noodles	japanese-udon/other_noodles
Japanese-Unagi/Anago	japanese-unagi/anago
Japanese-Yakiniku	japanese-yakiniku
Japanese-Yakitori/Chicken	japanese-yakitori/chicken
Jewish/Kosher	jewish/kosher
Kebab	kebab
Korean	korean
Latin American	latin_american
Lebanese	lebanese
Lunch	lunch
Malaysian	malaysian
Maltese	maltese
Mediterranean	mediterranean
Mexican	mexican
Mexican-Oaxaquena	mexican-oaxaquena
Mexican-Poblana	mexican-poblana
Mexican-Veracruzana	mexican-veracruzana

Name	FoodType ID
Mexican-Yucateca	mexican-yucateca
Middle Eastern	middle_eastern
Moroccan	moroccan
Natural/Healthy	natural/healthy
Noodles	noodles
North African	north_african
Norwegian	norwegian
Oceanic	oceanic
Organic	organic
Pakistani	pakistani
Pastries	pastries
Peruvian	peruvian
Pizza	pizza
Polish	polish
Portuguese	portuguese
Romanian	romanian
Russian	russian
Sandwich	sandwich
Scandinavian	scandinavian
Seafood	seafood

Name	FoodType ID
Seychellois	seychellois
Sicilian	sicilian
Singaporean	singaporean
Snacks and Beverages	snacks_and_beverages
Soup	soup
South African	south_african
South American	south_american
Southeast Asian	southeast_asian
Spanish	spanish
Spanish-Tapas	spanish-tapas
Sri Lankan	sri_lankan
Steak House	steak_house
Surinamese	surinamese
Swedish	swedish
Swiss	swiss
Thai	thai
Tibetan	tibetan
Turkish	turkish
Ukrainian	ukrainian
Vegan	vegan

Name	FoodType ID
Vegetarian	vegetarian
Venezuelan	venezuelan
Vietnamese	vietnamese
Yemeni	yemeni

Business Chains filter

This section explains how to filter search results using business chain identifiers within Amazon Location Service's Places API. This feature allows users to refine location searches based on well-known business chains, helping enhance the search experience for brand-specific requests.

The business chains filter enables focused search results by allowing users to specify one or more chain identifiers, useful for finding specific brand locations (for example, coffee shops, gas stations).

To apply business chain filters, use the Place APIs with the `businessChain` parameter set to the desired chain identifier. For implementation details, see the Amazon Location Service [API documentation](#).

Localization and internationalization

Localization (L10n) and Internationalization (I18n) are key processes in adapting software, content, or applications for different languages, regions, and views. Here's an overview in the context of specific factors:

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
Language	Yes	Yes	Yes	N/A	Yes	Yes	Yes
Political View	Yes	Yes	Yes	N/A	Yes	Yes	Yes

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
Time Zone	Yes	Yes	No	N/A	Yes	Yes	Yes
Phonemes	No	No	No	N/A	Yes	Yes	Yes

Language

The feature allows you to select a preferred response language from BCP47-compliant codes. It detects the query language based on name variants and uses the preferred language for unmatched tokens and ambiguous cases. If no requested language, Places API provides results in the official country language, but it prioritizes the regional language in regions where it differs.

As a fallback strategy, Places APIs return addresses in the default language if some address elements are unavailable in the requested language.

Political view

Places APIs allow you to adapt results to reflect local political sensitivities or guidelines of the local government. By default, Places APIs show an international perspective on disputed political boundaries.

- **AR:** Argentina's view on the Southern Patagonian Ice Field and Tierra Del Fuego, including the Falkland Islands, South Georgia, and South Sandwich Islands.
- **EG:** Egypt's view on Bir Tawil.
- **IN:** India's view on Gilgit-Baltistan.
- **KE:** Kenya's view on the Ilemi Triangle.
- **MA:** Morocco's view on Western Sahara.
- **RU:** Russia's view on Crimea.
- **SD:** Sudan's view on the Halaib Triangle.
- **RS:** Serbia's view on Kosovo, Vukovar, and Sarengrad Islands.
- **SR:** Suriname's view on the Courantyne Headwaters and Lawa Headwaters.
- **SY:** Syria's view on the Golan Heights.
- **TR:** Turkey's view on Cyprus and Northern Cyprus.

- **TZ:** Tanzania's view on Lake Malawi.
- **UY:** Uruguay's view on Rincon de Artigas.
- **VN:** Vietnam's view on the Paracel Islands and Spratly Islands.

Time zone

The feature provides additional time zone information. This includes time zone names and UTC offsets.

Phonemes

The feature provides additional phoneme information on how to pronounce the various components of the address or place.

Contacts and opening hours

Amazon Location Service provides details on contacts and opening hours for various points of interest (POIs), enabling applications to offer comprehensive information about business operations. This section covers contact and business hours information and how to retrieve and interpret these details effectively.

Amazon Location Service can provide structured data on the contact details and business hours of points of interest (POIs), enabling applications to display accurate information. This data is valuable for applications where real-time availability or business operations influence customer experience.

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
Contacts	No	No	No	Yes	Yes	Yes	No
Opening Hours	No	No	No	Yes	Yes	Yes	No

Contact details

Contact details for a place include structured information such as phone numbers, email addresses, and websites. These details are available for businesses listed in place data.

Phone Number

The primary contact number for a business. This field may include international dialing codes to ensure accurate connection, regardless of the caller's location.

Email Address

The primary contact email for inquiries. It may be available for businesses that provide an email address in their public listing.

Website

A link to the official website, providing users with additional information or access to services such as online booking or support.

Opening hours

Opening hours indicate the regular business hours for a location, providing users with insights into availability. This information is crucial for applications where users need to know when a business is open or closed.

Regular Hours

The standard weekly opening hours, typically provided as daily ranges (e.g., Monday to Friday, 9 AM to 5 PM). These indicate the usual operating schedule.

Special Hours

Exceptional hours for holidays or special events, provided as overrides to regular hours. For example, holiday hours or closures can be indicated to inform users of temporary changes in schedule.

Open Now

An indicator of whether a location is currently open, based on the local time. This information is helpful for users looking for businesses that are open at the time of their query.

For further details, refer to the Amazon Location Service [API documentation](#) to explore options for retrieving contact details and opening hours for points of interest.

Additional features

This section provides an overview of additional features supported by the Place APIs. These features provide additional location information by including requested details such as contact information, accessibility, and phonemes. You need to request additional feature by adding `AdditionalFeatures: [<Value1>, <Value 2>, ..]` in your request payload.

Note

Review [the section called “Places pricing”](#) for additional information about cost.

Valid Values	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
Core	Default	Default	Yes	Default	Default	Default	Yes
Time Zone	Yes	Yes	No	Yes	Yes	Yes	Yes
Phonemes	No	No	No	Yes	Yes	Yes	Yes
Contacts	No	No	No	Yes	Yes	Yes	No
Opening Hours	No	No	No	Yes	Yes	Yes	No
Access Points	No	Yes	No	Yes	Yes	Yes	Yes

To use these additional features, set the `additionalFeatures` parameter in the Place API requests. Refer to the Amazon Location Service [API documentation](#) for details.

For additional information on Time Zone and Phonemes, see [the section called “Localization and internationalization”](#).

For additional information on Contacts and Opening Hours, see [the section called “Contacts and opening hours”](#).

Definitions

Time Zone

Displays the time zone of a given place.

Phonemes

Offers phonetic representations for place names, aiding pronunciation.

Contacts

Provides contact details such as phone numbers and email addresses for places.

Opening Hours

An indicator of whether a location is currently open, based on the local time. This information is helpful for users looking for businesses that are open at the time of their query.

Access Points

Includes access point information, such as entrances.

IntendedUse

Note

If results are stored, then they will be billed at the higher storage pricing tier. Use request parameter `IntendedUse` to specify whether the results are for single use or storage. See [the section called "Places pricing"](#) to understand costs associated with stored results.

When you call a place API, specify `IntendedUse` by setting the value to be either `SingleUse` or `Storage`, based on the intended use of the results. If you are going to store the results (even for caching purposes), you must choose the *storage* option, not the *single use* option.

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
SingleUse	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Filter Type	Geocode	Reverse Geocode	Autocomplete	Get Place	Search Text	Search Nearby	Suggest
Storage	Yes	Yes	No	Yes	Yes	Yes	No

Places APIs

Places enable applications to search, find, and retrieve details about points of interest, addresses, and specific locations. These capabilities enhance location-based services by providing context and improving user experience in search functions.

- **Geocode:** Converts addresses or place names into geographic coordinates (longitude, latitude), supporting applications that require address-to-location transformation for mapping and spatial analysis. For more information, see [the section called "Geocode"](#).
- **Reverse Geocode:** Converts geographic coordinates to the nearest address or place name, providing context for a location. For more information, See [the section called "Reverse Geocode"](#).
- **Autocomplete:** Suggests potential completions for user-entered text, improving efficiency in search input. For more information, See [the section called "Autocomplete"](#).
- **GetPlace:** Retrieves detailed information about a specified place, including attributes like address, contact details, and opening hours. For more information, See [the section called "GetPlace"](#).
- **SearchNearby:** Finds places within a specified radius of a given geographic point, suitable for "near me" searches. For more information, See [the section called "Search Nearby"](#).
- **SearchText:** Allows text-based searching for places or points of interest based on a keyword or phrase, ideal for finding locations by name or description. For more information, See [the section called "Search Text"](#).
- **Suggest:** Provides search term suggestions as users type, enhancing search relevance and user experience. For more information, See [the section called "Suggest"](#).

The following table presents a number of business use cases that are best solved with Places APIs.

Places use cases

The following section presents a number of business use cases that are best solved with Places APIs.

Business need	Useful API	Examples
<p>Add a neighbourhood business information in an application</p> <p>Also returns business information, such as categories and food types.</p>	SearchNearBy	
<p>Proximity Search for business chain</p> <p>Also returns business information, such as categories, food types, and access points.</p>	Search Text Suggest	
<p>Search by phone name</p> <p>Also returns business information, such as categories, food types, and access points.</p>	Search Text Suggest	
<p>Search by name of place, POI, category</p> <p>Also returns business information, such as categories, food types, and access points.</p>	Search Text Suggest	<p>the section called “Search for a place, POI, or business using a name”</p>
<p>Predict suggestions as user key query</p> <p>Also returns business information, such as categories</p>	Suggest	<p>the section called “Predict suggestions based on input”</p>

Business need	Useful API	Examples
s, food types, and access points.		
Autocomplete or suggest for autofill an address on checkout page Normalizes and standardizes addresses.	Autocomplete, Suggest	the section called “How to complete an address”
Convert a specific address to longitude and latitude coordinates Normalizes and standardizes addresses.	Geocode	
Convert longitude and latitude coordinates into a corresponding address Normalizes and standardizes addresses.	Reverse Geocode	the section called “How to reverse geocode for a position”
Get Timezone of a City Supports UTC offset and time zone name.	Geocode	
Get Timezone for longitude and latitude Supports UTC offset and time zone name.	Reverse Geocode	

Business need	Useful API	Examples
<p>Get place by place id</p> <p>Returns business information, such as categories, contacts, opening hours, and access points.</p>	<p>Get Place</p>	<p>the section called “Get results for a place ID”</p>
<p>Get name, contacts, and opening hours of a point of interest</p> <p>Returns business information, such as categories, food types, and access points.</p>	<p>Search Text, Search Nearby, Suggestion</p>	
<p>Provide places type for a place name</p> <p>Supports pre-filtering.</p>	<p>Geocode Autocomplete</p>	
<p>Provide places type for a latitude/longitude coordinates</p> <p>Supports pre-filtering.</p>	<p>Reverse Geocode</p>	
<p>Provide poi categories (such as hospital, store, museum and 500 more) for an address or POI</p> <p>Supports pre-filtering.</p>	<p>Search Text, Search Nearby, Suggestion</p>	

Business need	Useful API	Examples
<p>Add the type-ahead search behavior for completion or predictions</p> <p>Supports cost efficient, label-only, and address component.</p>	Autocomplete Suggest	the section called “Predict suggestions based on input”
<p>Visualize Places search and/or geocode result on a map</p> <p>All APIs return geocoordinates, except autocomplete.</p>	GetTile and GetStyled escriptor with rendering engine (MapLibre) with Places API	
<p>Enhance, clean, normalize and standardize your address database</p> <p>Supports address label, components, timezone, and more.</p>	Geocode, Reverse Geocode	

Geocode

Note

Stored results incur higher Storage pricing. Use the `IntendedUse` parameter to indicate single-use or storage. Refer to the [the section called “Places pricing”](#) to understand cost implications for stored results.

Geocoding transforms textual addresses or place names into geographic coordinates, along with detailed address components and additional information. This API supports flexible queries, including both free-form text and structured queries with street names, postal codes, and regions. Optional features include time zone data and political view adjustments.

Use cases

- **Add supplementary data to customer addresses:** Improve address records by including zip codes, coordinates, and utilizing persistent storage to support informed business and marketing decisions.
- **Address Data Standardization:** Apply geocoding in data pipelines or batch processes to standardize address data, utilizing persistent storage for ongoing reference.
- **Determine Time Zones:** Identify the time zone for cities or addresses to provide accurate timestamps for applications like travel, scheduling, and invoicing.

Understand the request

The request accepts optional parameters, such as `AdditionalFeatures`, `BiasPosition`, and `Filter`, to refine the search results. Additional options, like `Language`, `MaxResults`, and `PoliticalView`, provide further customization of the response. The required parameter is `Query`, which can be provided as free-form text or structured as `QueryComponents`. For more information, refer to the [Geocode API Reference](#).

The request includes the following key parameters:

Authentication

For authentication, the `Key` parameter is optional if other methods are in use.

- `Key`: Optional parameter for authentication.

For more information, see [Authentication](#)

Querying and biasing

Parameters used for querying and geographic biasing of results.

- `QueryText`: Free-form text query for searching locations.
- `QueryComponents`: Structured components such as address number, country, locality, or postal code for precise searching.
- `BiasPosition`: Geographic position to bias search results towards.

For more information, see [the section called "Querying and biasing"](#)

Refining Results

Apply filters to refine the results.

- **Filter:** Filter to include specific countries or place types. For more information, see [the section called “Filtering”](#).

Internationalization and localization

Specify language and apply political view for localized results.

- **Language:** Language for the results.
- **PoliticalView:** Applies a political view reflecting territorial claims.

For more information, see [the section called “Localization and internationalization”](#)

Additional Features

Request additional information like time zone details.

- **AdditionalFeatures:** Option to request additional details such as time zone. For more information, see [the section called “Additional features”](#).

Understand the response

The response provides `ResultItems`, which contain detailed location data such as `Address`, `PlaceId`, `Position`, and other relevant attributes. Additional features like `TimeZone` information or match scores for each query component may also be included. Each `ResultItem` represents a matched location or geocoding result based on the specified request parameters. For further details, see the [Geocode API Reference](#).

The response includes the following key parameters:

Address and related details

Details about the returned location, including address components.

- **Address:** Full address including country, region, postal code, and street details.
- **PostalCodeDetails:** Additional information related to postal codes.
- **StreetComponents:** Street details including base name and type.

Place types and categories

Information on the type and category of place returned.

- **Categories:** List of categories describing the place, for example *Restaurants*, *Schools*.
- **PlaceType:** Specifies the type of place, such as city, address, or region.

Result analysis

Scores indicating how closely results match the input query.

- `MatchScores`: Scores for matching precision.

Additional details

Includes extra location-related information as needed.

- `AccessPoints`: Geographic coordinates representing access points.
- `TimeZone`: Time zone information for the location.

Reverse Geocode

Note

If results are stored, they will be billed at the higher Storage pricing tier. Use the `IntendedUse` parameter to specify whether the results are for single use or storage. Refer to [the section called “Places pricing”](#) for cost implications associated with stored results.

Reverse Geocode converts geographic coordinates into a human-readable address or place. It provides detailed address components, place type, category, and street information, with filtering options based on place type to refine results. The API can also include additional features, such as time zone information and political view adjustments.

Use cases

- **Add supplementary data to customer position data:** Improve position data records by including zip codes, coordinates, and utilizing persistent storage to support informed business and marketing decisions.
- **Position data standardization:** Apply geocoding in data pipelines or batch processes to standardize position data, utilizing persistent storage for ongoing reference.
- **Determine time zones:** Identify the time zone for cities or addresses to provide accurate timestamps for applications like travel, scheduling, and invoicing.

Understand the request

The Reverse Geocode API request accepts a combination of required and optional parameters to customize results. The required parameter `QueryPosition` (longitude and latitude) specifies the coordinates to reverse geocode. Optional parameters include `AdditionalFeatures` for extra data, `Filter` to refine results, and `MaxResults` to limit the number of returned results. Additional options like `Language`, `PoliticalView`, and `IntendedUse` allow for further customization. For more details, refer to the [Reverse Geocode API Reference](#).

The request accepts the following key parameters:

Authentication

The `Key` parameter is optional if other authentication methods are used.

- `Key`: Optional parameter for authentication.

For more information, see [Authentication](#).

Querying

Parameters used for defining the geographic location and search radius.

- `QueryPosition`: Specifies the longitude and latitude for the reverse geocode request.
- `QueryRadius`: Defines the search radius around the coordinates.

For more information, see [the section called "Querying and biasing"](#).

Refining results

Apply filters to limit the search results to specific countries or place types.

- `Filter`: Filter for results based on country or place type.

For more details, see [the section called "Filtering"](#).

Internationalization and localization

Options to customize the language and apply a political view to the results.

- `Language`: Specifies the language of the results.
- `PoliticalView`: Applies a political view reflecting territorial claims.

For more information, see [the section called "Localization and internationalization"](#).

Additional features

Requests extra data, such as time zone information.

- `AdditionalFeatures`: Option to request additional data such as time zone details.

For more information, see [the section called “Additional features”](#).

Understand the response

The response from the Reverse Geocode API provides detailed information about locations at the specified coordinates. The `ResultItems` array includes a list of place objects, each containing address details, categories, and geographic position. Additional data such as `TimeZone`, `FoodTypes`, and `PostalCodeDetails` offer further context. The response also includes bounding box information for mapping and the distance from the query position. For more details, see the Reverse Geocode API Reference.

The response includes the following key data:

Address and related details

Detailed address information for the returned location.

- `Address`: Complete address information, including country, region, postal code, and street details.
- `PostalCodeDetails`: Additional details related to postal codes, such as classifications and postal authorities.
- `StreetComponents`: Additional details about the street, including base name and type.

Place types and categories

Describes the type and category of the returned place.

- `Categories`: Categories describing the place, such as *Restaurants* or *Schools*.
- `PlaceType`: Specifies the type of place, such as a city, address, or region.

Result analysis

Information on how closely each result matches the input query.

- `MatchScores`: Scores indicating the accuracy of each match to the input query.

Additional details

Provides extra location-related data.

- **TimeZone:** Time zone information for the location.
- **AccessPoints:** Geographic coordinates representing entry points to the location.

Autocomplete

Note

By default, Autocomplete returns only the ID and Title fields, providing a cost-effective option. Additional address components and highlights can be requested by setting `additionalFeatures` to `Core`. Refer to [the section called “Places pricing”](#) for cost details associated with stored results. Note that Autocomplete does not return geocodes, and this feature is currently unavailable in Japan.

The Autocomplete API completes potential places and addresses as the user types, based on partial input. It enhances the efficiency and accuracy of address entry by completing queries with valid address information after a few keystrokes. Additionally, Autocomplete supports result filtering based on geographic location, country, or specific place types and can be tailored using optional parameters such as language and political view.

Use cases

- **Enhance checkout experience:** Provide real-time address completion as customers enter their location on websites or apps. Ensure that delivery or pickup locations match known addresses to streamline address validation, reduce errors, and create a seamless checkout experience.
- **Support customer services:** Offer real-time address suggestions for customer services such as contact centers or emergency services, streamlining the process of locating accurate addresses and improving user satisfaction by reducing the time needed to obtain correct address information.

Understand the request

The Autocomplete request uses optional parameters to refine search results. For more details, refer to the Autocomplete API Reference.

The request accepts the following key parameters:

Authentication

The Key parameter is optional if other authentication methods are used.

- **Key:** Optional parameter for authentication.

For more information, see [Authentication](#)

Querying

Defines the partial text query and geographic bias for result suggestions.

- **QueryText:** The free-form text query used to find location results, this parameter is required.
- **BiasPosition:** Suggests results closest to specified longitude and latitude coordinates, this parameter is optional.

For more information, see [the section called "Querying and biasing"](#)

Refine results

Applies filters to restrict results to specific countries or place types.

- **Filter:** Allows filtering based on specified countries or place types.

For more details, see [the section called "Filtering"](#)

Understand the response

The Autocomplete API returns results as `ResultItems`, each representing a matched location based on the request parameters. For further details, refer to the Autocomplete API Reference.

The response includes the following key data:

Address and related details

Contains detailed address components for each location.

- **Address:** Provides full address components, including street, postal code, and country.

Result analysis

Provides analysis data for each result in relation to the input query.

- **Distance:** Distance from the specified bias position to the result.
- **Highlights:** Highlights portions of the query within the result for clarity.

GetPlace

Note

If results are stored, they will be billed at the higher Storage pricing tier. Use the `IntendedUse` parameter to specify whether the results are for single use or storage. Refer to [the section called “Places pricing”](#) for cost details associated with stored results.

`GetPlace` retrieves detailed information about a place using its unique `PlaceId`. The request supports optional parameters for additional features, such as time zone data or phonemes, and language localization. The API returns information like address, coordinates, business details, contact methods, and time zone information.

Use cases

- **Refresh stored data:** Regularly refresh stored data to ensure your application provides accurate, current information. Use the `GetPlace` API to update details about businesses and places, ensuring users have the latest information. For example, refresh opening hours, contact details, and user reviews for local restaurants or shops to enhance user trust and application functionality.
- **Retrieve place details after typeahead suggestions:** Improve user experience by integrating the `GetPlace` API with autocomplete or suggestion features. As users type an address, business name, or point of interest, suggestions provide relevant places, and `GetPlace` enables access to more detailed information once a selection is made.

Understand the request

The `GetPlace` API request requires specific parameters, such as the `PlaceId`, and supports optional parameters for additional features or language localization. These parameters help tailor the results based on language, intended use, and geographic context. For more details, refer to the `GetPlace` API Reference.

The request accepts the following key parameters:

Authentication

The `Key` parameter is optional if other authentication methods are used.

- **Key:** API key for authorization.

For more information, see [Authentication](#).

Querying

Defines the unique identifier for retrieving place information.

- **PlaceId:** The unique identifier for the place. (required)

For more information, see [the section called "Querying and biasing"](#).

Internationalization and Localization

Options to customize the language and apply a political view to the results.

- **Language:** Specifies the language of the results.
- **PoliticalView:** Applies a political view reflecting territorial claims.

For more information, see [the section called "Localization and internationalization"](#).

Additional Features

Requests additional data, such as time zone details.

- **AdditionalFeatures:** Option to request additional information like time zone or phoneme details.

For more information, see [the section called "Additional features"](#).

Understand the response

The response from the GetPlace API includes basic data, such as the address, coordinates, and business details, as well as additional data based on the request parameters. Information like access restrictions, time zones, and phoneme details for pronunciation can also be provided. For further details, refer to the GetPlace API Reference.

The response includes the following key data:

Address and Related Details

Provides complete address details for the location.

- **Address:** Full address information, including country, region, and postal code.

- `PostalCodeDetails`: Additional information about postal codes and authorities.

Place Types and Categories

Describes the type and category of the place.

- `BusinessChains`: Information on any associated business chains.
- `Categories`: Categories to which the place belongs, such as food or retail.
- `PlaceType`: Type of the place, such as point of interest or locality.

Additional Details

Provides additional data about the place, as specified in the request.

- `Contacts`: Contact methods, including phone numbers, emails, and websites.
- `OpeningHours`: Business hours or access times.
- `AccessPoints`: Geographic coordinates (longitude and latitude) of access points.
- `Phonemes`: Pronunciations for various address components.
- `TimeZone`: Time zone information, including offset.

Search Nearby

Note

If results are stored, they will be billed at the higher Storage pricing tier. Use the `IntendedUse` parameter to specify whether the results are for single use or storage. Refer to [the section called “Places pricing”](#) for cost implications associated with stored results.

The Search Nearby API allows developers to query for points of interest within a specified radius from central coordinates. The API provides place results with optional filters, including categories, business chains, food types, and more. Response details include place name, address, phone, category, food type, contact, and opening hours. Additional features, such as phonemes, time zones, and others, are available based on requested parameters.

Use cases

- **Real Estate Applications**: Provide potential homebuyers or renters with a list of amenities and services near a property, such as schools, hospitals, and shopping centers.

- **Local Business Discovery:** Enable users to locate businesses, restaurants, or shops within a specific radius from their current location. For example, a food delivery app can display nearby restaurants based on real-time location.
- **Travel and Tourism Apps:** Help travelers explore attractions, landmarks, hotels, and services near their location or a destination. A travel app might suggest nearby points of interest such as museums, parks, and historic sites.
- **Emergency Services:** Assist users in finding the nearest hospitals, police stations, or pharmacies during emergencies, especially valuable for apps focused on health, safety, or disaster response.
- **Transportation and Logistics:** Help users find the nearest gas stations, EV charging points, or public transit stops. Ride-hailing services can display nearby available drivers.

Understand the request

The Search Nearby API request includes required parameters such as `QueryPosition` and optional parameters like filters to refine the search. Additional options, like `Language` and `PoliticalView`, customize results for specific needs. For more information, refer to the [Search Nearby API Reference](#).

The request includes the following key parameters:

Authentication

The `Key` parameter is optional if other authentication methods are used.

- `Key`: API key for authorization.

For more information, see [Authentication](#)

Querying

Defines the location and search radius for the query.

- `QueryPosition`: Specifies the longitude and latitude coordinates.
- `QueryRadius`: Defines the search radius around the specified coordinates.

For more information, see [the section called "Querying and biasing"](#)

Refining Results

Filters results to narrow down search criteria.

- `Filter`: Applies filters such as categories, business chains, or food types.

For more details, see [the section called "Filtering"](#)

Internationalization and Localization

Options to customize the language and apply a political view to the results.

- `Language`: Specifies the language of the results.
- `PoliticalView`: Applies a political view reflecting territorial claims.

For more information, see [the section called "Localization and internationalization"](#)

Additional Features

Requests additional data, such as time zone information.

- `AdditionalFeatures`: Option to request additional details like time zone information.

For more information, see [the section called "Additional features"](#)

Understand the response

The response from the Search Nearby API provides detailed results matching the query, including location, address details, business chains, contacts, phonemes, time zone, and opening hours. For more details, refer to the Search Nearby API Reference.

The response includes the following key data:

Address and Related Details

Comprehensive address information for the returned location.

- `Address`: Includes full address details, such as country and street.

Place Types and Categories

Describes the type and category of the returned place.

- `Categories`: Categories describing the place, such as *Restaurants* or *Schools*.
- `FoodTypes`: Types of food available at restaurants or eateries.
- `BusinessChains`: Indicates any associated business chains.

Additional Details

Additional information related to the place, as requested.

- `Contacts`: Lists emails, phone numbers, and websites.
- `OpeningHours`: Specifies operational hours for the place.

- **Phonemes:** Provides phonetic representations of address components.
- **TimeZone:** Includes time zone information for the place.

Search Text

Note

If results are stored, they will be billed at the higher Storage pricing tier. Use the `IntendedUse` parameter to specify whether the results are for single use or storage. Refer to [the section called “Places pricing”](#) for cost implications associated with stored results.

Search Text allows you to query location data using a single free-form text input, returning place results with optional filters like country, geographic circle, or bounding box. The API also enables searching by `QueryId` returned from the Suggest API. The response includes details such as place name, address, phone number, category, food type, contact information, and opening hours. Optional features like phonemes, time zones, and more are also available based on specified parameters.

Use cases

- **Neighborhood search by text:** Help customers locate the nearest relevant place. Customers can find the closest address, business, or attraction such as museums, shops, or parks.
- **Business discovery:** Allow users to find specific businesses or types of places by entering keywords like "Italian restaurant" or "coffee shop." This feature is particularly useful for food delivery or restaurant recommendation apps.
- **Place search in travel apps:** Enable users to search for locations, landmarks, or businesses using free-form text, such as "hotels near Eiffel Tower," to receive relevant suggestions quickly.
- **Educational institutions search:** Allow users to search for schools, colleges, or training centers (for example, "elementary schools" or "universities"), a helpful tool for parents or students.

Understand the request

The Search Text API request allows various parameters to refine the search. Optional parameters enable additional features, location biasing, filtering criteria, language preferences, and result limits. For more details, refer to the Search Text API Reference.

The request includes the following key parameters:

Authentication

The Key parameter is optional if other authentication methods are used.

- **Key**: API key for authorization.

For more information, see [Authentication](#).

Querying

Parameters used for specifying the text search and location bias.

- **QueryText**: Free-form text for searching places.
- **QueryId**: Enables completion of suggested queries from the Suggest API.
- **BiasPosition**: Prioritizes results near a specific longitude and latitude.

For more information, see [the section called "Querying and biasing"](#).

Refining Results

Filters results to narrow down search criteria.

- **Filter**: Allows filtering by bounding box or circular area to limit search results.

For more details, see [the section called "Filtering"](#).

Internationalization and Localization

Options for customizing the language and applying a political view to the results.

- **Language**: Specifies the language of the results.
- **PoliticalView**: Applies a political view reflecting territorial claims.

For more information, see [the section called "Localization and internationalization"](#).

Additional Features

Requests extra data such as time zone information.

- **AdditionalFeatures**: Option to request additional data such as time zone details.

For more information, see [the section called "Additional features"](#).

Understand the response

The response from the Search Text API includes detailed results that match the query, with information on location, address details, business chains, contacts, phonemes, time zone, and opening hours. For more details, refer to the Search Text API Reference.

The response includes the following key data:

Address and Related Details

Comprehensive address information for the returned location.

- **Address:** Includes country, street, and other address details.

Place Types and Categories

Describes the type and category of the place.

- **Categories:** A list of categories describing the place, such as *Restaurants* or *Schools*.
- **PlaceType:** Specifies the type of place, such as a city, address, or region.
- **BusinessChains:** Indicates any associated business chains.

Additional Details

Additional data related to the place, as requested.

- **Contacts:** Provides emails, phone numbers, and websites.
- **OpeningHours:** Operational hours of the place.
- **AccessPoints:** Geographic coordinates associated with the place.
- **Phonemes:** Phonetic representations of address components.
- **TimeZone:** Time zone information, including offset.

Suggest

Note

By default, the Suggest API returns only ID and Title fields, providing a cost-effective option. Additional address components and highlights can be requested by setting `additionalFeatures` to `Core`. Refer to [the section called "Places pricing"](#) for cost implications related to stored results.

Suggest provides predictions or recommendations based on user input or context, such as relevant places, points of interest, query terms, or search categories. It assists users in finding places, points of interest, or identifying follow-up queries based on incomplete or misspelled input. The API returns a list of possible matches or refinements, which can be used to formulate a more accurate query. Users can select the appropriate suggestion for further searching. The API supports filtering results by location and other attributes and offers additional features like phonemes and time zones. The response includes refined query terms and detailed place information.

Use cases

- **Ride-Hailing Services:** Provide real-time suggestions to quickly complete addresses for pick-up and drop-off locations, ensuring accuracy and faster ride booking.
- **Travel and Navigation Services:** Offer real-time predictions for locations or landmarks, even for misspelled or partially entered terms, such as "Eifel" for "Eiffel Tower." The API refines suggestions to offer relevant nearby points of interest, helping users locate places accurately.
- **Restaurant Search Assistance:** Anticipate users' interests in restaurants and suggest nearby dining options, enhancing the search experience.

Understand the request

The Suggest API request uses parameters to generate suggestions based on user input. Optional parameters allow for refined search results using location bias and filtering criteria. For more details, refer to the Suggest API Reference.

The request includes the following key parameters:

Authentication

The Key parameter is optional if other authentication methods are used.

- **Key:** API key for authorization.

For more information, see [Authentication](#)

Querying

Defines the free-text search and location bias.

- **QueryText:** Free-form text for generating suggestions. (required)
- **BiasPosition:** Prioritizes suggestions near a specific longitude and latitude.

For more information, see [the section called “Querying and biasing”](#)

Refining results

Filters results to narrow down search criteria.

- `Filter`: Allows filtering by bounding box or circular area to limit search results.

For more details, see [the section called “Filtering”](#)

Internationalization and Localization

Options for customizing the language and applying a political view to the results.

- `Language`: Specifies the language of the results.
- `PoliticalView`: Applies a political view reflecting territorial claims.

For more information, see [the section called “Localization and internationalization”](#)

Additional features

Requests extra data, such as time zone information.

- `AdditionalFeatures`: Option to request additional details such as time zone or phonetic data.

For more information, see [the section called “Additional features”](#)

Limiting results

Sets limits on the number of results or query refinements returned.

- `MaxQueryRefinements`: Limits the number of query refinement terms returned.
- `MaxResults`: Limits the number of suggestions returned.

Understand the response

The response provides suggested addresses or places based on the input query, with attributes such as location, address details, business chains, contacts, phonemes, time zone, and opening hours. The API returns `ResultItems`, representing possible matches for completing the input query. There are two types of results, identified by `SuggestResultItemType`: results of type `Query` suggest a follow-up category or chain query, which can be used to obtain focused results for a specified category by passing the `QueryID` to the `SearchText` API. Results of type `Place` provide a final result with an address and additional information about the place. For further details, refer to the API Reference for `Suggest` API.

The response includes the following key data:

Result analysis

Provides information on refining the input query.

- **QueryRefinements:** Terms that can be used to refine the search query.
- **Highlights:** Highlights parts of the address or title that match the query.

Place types and categories

Describes the type and category of the place.

- **Categories:** Categories describing the place, such as *Restaurants* or *Schools*.
- **PlaceType:** Specifies the type of place, such as a city, address, or region.
- **BusinessChains:** Indicates any associated business chains.

Additional details

Additional information about the place, as specified in the request.

- **Contacts:** Provides emails, phone numbers, and websites.
- **OpeningHours:** Operational hours of the place.
- **AccessPoints:** Geographic coordinates associated with the place.
- **Phonemes:** Phonetic representations of address components.
- **TimeZone:** Time zone information, including offset.

How to

This section provides step-by-step instructions for performing specific tasks with the Places APIs. These guides are designed to help you quickly and effectively implement key functionality. Each topic focuses on a distinct use case, ensuring you have the practical knowledge needed to integrate Places API features into your applications.

Examples

- [How to use Geocode](#)
- [How to use ReverseGeocode](#)
- [How to use Autocomplete](#)

- [How to use Get Place](#)
- [How to use SearchNearby](#)
- [How to use SearchText](#)
- [How to use Suggest](#)

How to use Geocode

This section contains a variety of how to guides and examples for how to use Geocode APIs.

Examples

- [How to geocode an administrative and postal area](#)
- [How to geocode an address](#)

How to geocode an administrative and postal area

The Geocode API allows you to perform geocoding for a geographic area using a query text input, such as the name of a country, region (state or province), or city. The API response includes location details like geographic coordinates, bounding boxes for map visualizations, and match scores indicating the result's relevance to the query.

Potential use cases

- **Obtain coordinates for an administrative area:** Use coordinates as a bias position or center in other Places APIs.
- **Visualize information on a map:** Geocoded coordinates can be used to display data visually on a map.

Examples

Geocode a Country

Sample request

```
{
  "QueryText": "Canada"
}
```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId": "AQAAADgAxLuIjs1R50umRP-
z1Xm2qBkNSPJS30qtFNxlBi89q16BPU7n4FZ1jGwn8jVkmSy6wC8WS57BmgoenFEUKbEn9MNBitVKEwBGpGpZgVbY_ux",
      "PlaceType": "Country",
      "Title": "Canada",
      "Address": {
        "Label": "Canada",
        "Country": {
          "Code2": "CA",
          "Code3": "CAN",
          "Name": "Canada"
        }
      },
      "Position": [-75.69122, 45.42177],
      "MapView": [-141.00271, 41.67659, -52.61901, 83.11062],
      "MatchScores": {
        "Overall": 1,
        "Components": {
          "Address": { "Country": 1 }
        }
      }
    }
  ]
}
```

cURL

```
curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/geocode?key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
    "QueryText": "Canada"
  }'
```

AWS CLI

```
aws geo-places geocode --key YOUR_API_KEY --query-text "Canada"
```

Geocode a Region

Sample request

```
{
  "QueryText": "BC"
}
```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId": "AQAAADgAgmWdm7J1hyZyJyi__90eghzvK0jVGxtF66m6PrEohbBHDP-
eMqk3Poh1c6Pz6hIwhcakKpmffJizGgpFRMgykyoFQBxnMN7I3mB10_0NSQ_HSy4QUwhYog",
      "PlaceType": "Region",
      "Title": "BC, Canada",
      "Address": {
        "Label": "BC, Canada",
        "Country": {
          "Code2": "CA",
          "Code3": "CAN",
          "Name": "Canada"
        },
        "Region": {
          "Code": "BC",
          "Name": "British Columbia"
        }
      },
      "Position": [-123.36445, 48.42854],
      "MapView": [-139.04941, 48.22478, -114.05201, 60.00043],
      "MatchScores": {
        "Overall": 1,
        "Components": {
          "Address": { "Region": 1 }
        }
      }
    }
  ]
}
```

cURL

```
curl --request POST \  
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/geocode?key=Your_Key' \  
  --header 'Content-Type: application/json' \  
  --data '{  
    "QueryText": "BC"  
  }'
```

AWS CLI

```
aws geo-places geocode --key ${YourAPIKey} --query-text "BC"
```

Geocode a City

Sample request

```
{  
  "QueryText": "Vancouver"  
}
```

Sample response

```
{  
  "ResultItems": [  
    {  
      "PlaceId": "AQAAADgAv5UG-  
H662VWgYRpC6bkL20kbW8PCPmweGcfkJ50JkGUrzoB6QBYePP_nmHprt7JQzkCx31uA_P35-  
YppSnEPQPgyHlI7GMaorJpfvkCI0A0T6sVsyMQfGQ",  
      "PlaceType": "Locality",  
      "Title": "Vancouver, BC, Canada",  
      "Address": {  
        "Label": "Vancouver, BC, Canada",  
        "Country": {  
          "Code2": "CA",  
          "Code3": "CAN",  
          "Name": "Canada"  
        },  
        "Region": {  
          "Code": "BC",  
          "Name": "British Columbia"  
        }  
      }  
    }  
  ]  
}
```

```

    },
    "SubRegion": {
      "Name": "Metro Vancouver"
    },
    "Locality": "Vancouver",
    "PostalCode": "V5Y"
  },
  "Position": [-123.11336, 49.26038],
  "MapView": [-123.26754, 49.19891, -123.02301, 49.33557],
  "MatchScores": {
    "Overall": 1,
    "Components": {
      "Address": { "Locality": 1 }
    }
  }
}
]
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/geocode?key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
    "QueryText": "Vancouver"
  }'

```

AWS CLI

```

aws geo-places geocode --key ${YourAPIKey} --query-text "Vancouver"

```

Geocode a Postal Code

You can geocode a postal code. Use `IncludePlaceTypes` with `["PostalCode"]` for more precise results.

Sample request

```

{
  "QueryText": "800006",
  "Filter": { "IncludePlaceTypes": ["PostalCode"] }
}

```

```
}

```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId": "AQAAADgA6mZNpk60x-nT_P9zmmqWD8jsYAe-
xSwoupPbwVcnP0Gix9pnctWQMuGf7FE4IAxP7KfldWvTin0zXoe2dNEcmQ0v7eveQbevZDy6M0izWoqFgdkcHe6gWA",
      "PlaceType": "PostalCodeArea",
      "Title": "800006, Patna, Bihar, India",
      "Address": {
        "Label": "800006, Patna, Bihar, India",
        "Country": {
          "Code2": "IN",
          "Code3": "IND",
          "Name": "India"
        },
        "Region": {
          "Code": "BR",
          "Name": "Bihar"
        },
        "SubRegion": { "Name": "Patna" },
        "Locality": "Patna",
        "PostalCode": "800006"
      },
      "Position": [85.18048, 25.61532],
      "MapView": [85.16599, 25.60054, 85.19103, 25.6221],
      "MatchScores": {
        "Overall": 1,
        "Components": {
          "Address": { "PostalCode": 1 }
        }
      }
    }
  ]
}
```

cURL

```
curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/geocode?key=Your_Key' \
  --header 'Content-Type: application/json' \
```

```
--data '{
  "QueryText": "800006",
  "Filter": { "IncludePlaceTypes": ["PostalCode"] }
}'
```

AWS CLI

```
aws geo-places geocode --key ${YourAPIKey} --query-text "800006" --filter
'{"IncludePlaceTypes": ["PostalCode"]}'
```

Developer tips

Use filters like `IncludeCountries` and `IncludePlaceTypes` for more targeted results. For example, to ensure results from Vancouver in the USA, set `"IncludeCountries": ["USA"]`. For more details, see .

```
{
  "QueryText": "Vancouver",
  "Filter": { "IncludeCountries": ["USA"] }
}
```

How to geocode an address

The Geocode API enables you to geocode a specific point address, interpolated address, or street. The API response contains location information, including geographical coordinates and match scores that indicate how accurately the result aligns with the query.

Potential use cases

- **Clean address database:** Enhance data quality by identifying and correcting errors in address records.
- **Normalize and standardize addresses:** Ensure consistent address formatting across datasets for improved data interoperability.
- **Enrich addresses with additional information:** Add geographic coordinates and other relevant details to address records to support location-based analytics and insights.

Examples

Use query text

Sample request

```
{
  "QueryText": "510 W Georgia St, Vancouver, BC"
}
```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId":
        "AQAAAGEADn1UqahfGsH9UyMN4IuUV410UdpbnAtstz7BmlgMJsDwdhsNUGmIarD5C0UY80NE7Wex2YE7Ut1YQh89M9nOP0rH2chHTLyW2YwHNhyGZJdJ43eBYRGRICLFPNdYey2p40IUCB3MZ7wk7o6hwx3rxziXT",
      "PlaceType": "PointAddress",
      "Title": "510 W Georgia St, Vancouver, BC V6B 0M3, Canada",
      "Address": {
        "Label": "510 W Georgia St, Vancouver, BC V6B 0M3, Canada",
        "Country": {
          "Code2": "CA",
          "Code3": "CAN",
          "Name": "Canada"
        },
        "Region": {
          "Code": "BC",
          "Name": "British Columbia"
        },
        "SubRegion": {
          "Name": "Metro Vancouver"
        },
        "Locality": "Vancouver",
        "District": "Downtown Vancouver",
        "PostalCode": "V6B 0M3",
        "Street": "W Georgia St",
        "StreetComponents": [
          {
            "BaseName": "Georgia",
            "Type": "St",
            "TypePlacement": "AfterBaseName",

```



```
        "TypeSeparator": " ",
        "Prefix": "W",
        "Language": "en"
    }
  ],
  "AddressNumber": "510"
},
"Position": [
  -123.11694,
  49.28126
],
"MapView": [
  -123.11832,
  49.28036,
  -123.11556,
  49.28216
],
"AccessPoints": [
  {
    "Position": [
      -123.11656,
      49.28151
    ]
  }
],
"MatchScores": {
  "Overall": 1,
  "Components": {
    "Address": {
      "Region": 1,
      "Locality": 1,
      "Intersection": [
        1
      ],
      "AddressNumber": 1
    }
  }
}
}
```

cURL

```
curl --request POST \  
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/geocode?key=Your_Key' \  
  --header 'Content-Type: application/json' \  
  --data '{"QueryText": "510 W Georgia St, Vancouver, BC"}'
```

AWS CLI

```
aws geo-places geocode --key #{YourAPIKey} --query-text "510 W Georgia St,  
Vancouver, BC"
```

Use query components

Sample request

```
{  
  "QueryComponents": {  
    "AddressNumber": "510",  
    "Locality": "Vancouver",  
    "Region": "BC",  
    "Country": "Canada",  
    "Street": "Georgia"  
  }  
}
```

Sample response

```
{  
  "ResultItems": [  
    {  
      "PlaceId":  
        "AQAAAGEADn1UqahfGsH9UyMN4IuUV410UdpbnAtstz7BmlgMJsDwdhsNUGmIarD5C0UY80NE7Wex2YE7Ut1YQh89M9  
nOP0rH2chHTLyW2YwHNhyGZJdJ43eBYRGRICLFPNdYey2p40IUCB3MZ7wk7o6hwx3rxziXT",  
      "PlaceType": "PointAddress",  
      "Title": "510 W Georgia St, Vancouver, BC V6B 0M3, Canada",  
      "Address": {  
        "Label": "510 W Georgia St, Vancouver, BC V6B 0M3, Canada",  
        "Country": {  
          "Code2": "CA",  
          "Code3": "CAN",
```

```
        "Name": "Canada"
    },
    "Region": {
        "Code": "BC",
        "Name": "British Columbia"
    },
    "SubRegion": {
        "Name": "Metro Vancouver"
    },
    "Locality": "Vancouver",
    "District": "Downtown Vancouver",
    "PostalCode": "V6B 0M3",
    "Street": "W Georgia St",
    "StreetComponents": [
        {
            "BaseName": "Georgia",
            "Type": "St",
            "TypePlacement": "AfterBaseName",
            "TypeSeparator": " ",
            "Prefix": "W",
            "Language": "en"
        }
    ],
    "AddressNumber": "510"
},
"Position": [
    -123.11694,
    49.28126
],
"MapView": [
    -123.11832,
    49.28036,
    -123.11556,
    49.28216
],
"AccessPoints": [
    {
        "Position": [
            -123.11656,
            49.28151
        ]
    }
],
"MatchScores": {
```

```

    "Overall": 0.99,
    "Components": {
      "Address": {
        "Country": 1,
        "Region": 1,
        "Locality": 1,
        "Intersection": [
          0.78
        ],
        "AddressNumber": 1
      }
    }
  },
  {
    "PlaceId":
    "AQAAAFQANa0W3hvUXNNkNzaqab9MbQ7C83Gmr5H6-3m67A8jLZcPBY2BENV5g82MUfbPIF75MevYN_a36qwmivU6K
    JmX_z_P7g6ZQcmgLLmi8jw2Zq-qHu1MUIEJa4Gx1FZrClfQ",
    "PlaceType": "InterpolatedAddress",
    "Title": "510 E Georgia St, Vancouver, BC V6A 1Z9, Canada",
    "Address": {
      "Label": "510 E Georgia St, Vancouver, BC V6A 1Z9, Canada",
      "Country": {
        "Code2": "CA",
        "Code3": "CAN",
        "Name": "Canada"
      },
      "Region": {
        "Code": "BC",
        "Name": "British Columbia"
      },
      "SubRegion": {
        "Name": "Metro Vancouver"
      },
      "Locality": "Vancouver",
      "District": "Strathcona",
      "PostalCode": "V6A 1Z9",
      "Street": "E Georgia St",
      "StreetComponents": [
        {
          "BaseName": "Georgia",
          "Type": "St",
          "TypePlacement": "AfterBaseName",
          "TypeSeparator": " ",

```

```
        "Prefix": "E",
        "Language": "en"
      }
    ],
    "AddressNumber": "510"
  },
  "Position": [
    -123.0932,
    49.27829
  ],
  "MapView": [
    -123.09458,
    49.27739,
    -123.09182,
    49.27919
  ],
  "AccessPoints": [
    {
      "Position": [
        -123.0932,
        49.27842
      ]
    }
  ],
  "MatchScores": {
    "Overall": 0.99,
    "Components": {
      "Address": {
        "Country": 1,
        "Region": 1,
        "Locality": 1,
        "Intersection": [
          0.78
        ],
        "AddressNumber": 1
      }
    }
  }
}
```

cURL

```
curl --request POST \  
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/geocode?key=Your_Key' \  
  --header 'Content-Type: application/json' \  
  --data '{"QueryComponents": {"AddressNumber": "510", "Locality": "Vancouver",  
"Region": "BC", "Country": "Canada", "Street": "Georgia"}}'
```

AWS CLI

```
./aws geo-places geocode --key ${YourAPIKey} --query-components '{  
"AddressNumber" : "510",  
"Locality": "vancouver",  
"Region": "BC",  
"Country": "Canada",  
"Street": "Georgia"}
```

Use hybrid query

Sample request

```
{  
  "QueryText": "W. 6th St",  
  "QueryComponents": {  
    "AddressNumber": "415",  
    "Locality": "Vancouver"  
  }  
}
```

Sample response

```
{  
  "ResultItems": [  
    {  
      "PlaceId":  
"AQAAAGAAsauU_5L3Wl03mFFqy3PP0JkK0w802Los0CT1z3hUbmLL3lsBpMgPBGGNKsKaolz-  
xNs9wgRcBAvpbz-  
Bev1C4ntCVbAognFsL6tgRqYc1o2zJwvSKpkS0xL2cdx88z0xHoB9PuJ7ma0KPX8mlrt-6UA4ddYetJ6GMMCRXWurxfg  
      "PlaceType": "PointAddress",  
      "Title": "415 W 6th St, Vancouver, WA 98660-3375, United States",  
      "Address": {
```

```
"Label": "415 W 6th St, Vancouver, WA 98660-3375, United States",
"Country": {
  "Code2": "US",
  "Code3": "USA",
  "Name": "United States"
},
"Region": {
  "Code": "WA",
  "Name": "Washington"
},
"SubRegion": {
  "Name": "Clark"
},
"Locality": "Vancouver",
"District": "Esther Short",
"PostalCode": "98660-3375",
"Street": "W 6th St",
"StreetComponents": [
  {
    "BaseName": "6th",
    "Type": "St",
    "TypePlacement": "AfterBaseName",
    "TypeSeparator": " ",
    "Prefix": "W",
    "Language": "en"
  }
],
"AddressNumber": "415"
},
"Position": [
  -122.67543,
  45.62527
],
"MapView": [
  -122.67672,
  45.62437,
  -122.67414,
  45.62617
],
"AccessPoints": [
  {
    "Position": [
      -122.67543,
      45.62506
    ]
  }
]
```

```

    ]
  }
],
"MatchScores": {
  "Overall": 1,
  "Components": {
    "Address": {
      "Locality": 1,
      "Intersection": [
        1
      ],
      "AddressNumber": 1
    }
  }
}
]
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/geocode?key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{"QueryText": "W. 6th St", "QueryComponents": {"AddressNumber": "415",
"Locality": "Vancouver"}}'

```

AWS CLI

```

./aws geo-places geocode -key ${YourAPIKey} --query-text "W. 6th St" \
--query-components '{"AddressNumber" : "415", "Locality": "Vancouver"}'

```

Developer Tips

Use filters like `IncludeCountries` and `IncludePlaceTypes` to obtain accurate results. For instance, if you need Vancouver from the USA, apply `"IncludeCountries": ["USA"]` to prioritize results in the USA.

```

{
  "QueryText": "Vancouver",
  "Filter": {

```



```
"IncludeCountries": ["USA"],
"IncludePlaceTypes": ["City"]
}
}
```

How to use ReverseGeocode

This section contains a variety of how to guides and examples for how to use ReverseGeocode APIs.

Topics

- [How to reverse geocode for a position](#)
- [How to Reverse Geocode with a Political View](#)
- [How to Reverse Geocode for Right Result](#)

How to reverse geocode for a position

The Reverse Geocode API allows you to convert a geocode to a geographic area based on a position query. The API response includes place details, providing information about the location associated with specific coordinates.

Potential use cases

- **Store place information:** Add place details to a datastore containing geocoordinates.
- **Map visualization:** Use place information to display data on a map.
- **User location detection:** Identify the user's location based on their device position.

Reverse geocode a position

Sample request

```
{
  "QueryPosition": [
    -123.11694,
    49.28126
  ]
}
```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId": "AQAAAGEAD...",
      "PlaceType": "PointAddress",
      "Title": "510 W Georgia St, Vancouver, BC V6B 0M3, Canada",
      "Address": {
        "Label": "510 W Georgia St, Vancouver, BC V6B 0M3, Canada",
        "Country": { "Code2": "CA", "Code3": "CAN", "Name": "Canada" },
        "Region": { "Code": "BC", "Name": "British Columbia" },
        "SubRegion": { "Name": "Metro Vancouver" },
        "Locality": "Vancouver",
        "District": "Downtown Vancouver",
        "PostalCode": "V6B 0M3",
        "Street": "W Georgia St",
        "StreetComponents": [
          { "BaseName": "Georgia", "Type": "St", "TypePlacement":
"AfterBaseName", "TypeSeparator": " ", "Prefix": "W", "Language": "en" }
        ],
        "AddressNumber": "510"
      },
      "Position": [-123.11694, 49.28126],
      "Distance": 0,
      "MapView": [-123.11813, 49.27786, -123.11076, 49.28246],
      "AccessPoints": [
        { "Position": [-123.11656, 49.28151] }
      ]
    }
  ]
}
```

cURL

```
curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/reverse-geocode?
key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
    "QueryPosition": [
      -123.11694,
      49.28126
    ]
  }'
```

```
]
}'
```

AWS CLI

```
aws geo-places reverse-geocode --key ${YourKey} --query-position
"-123.11694,49.28126"
```

Reverse geocode a position with multiple results

Sample request

```
{
  "QueryPosition": [
    -123.11694,
    49.28126
  ],
  "MaxResults": "3"
}
```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId": "AQAAAGEAD...",
      "PlaceType": "PointAddress",
      "Title": "510 W Georgia St, Vancouver, BC V6B 0M3, Canada",
      "Address": { /* Address details */ },
      "Position": [-123.11694, 49.28126],
      "Distance": 0,
      "MapView": [/* Map view details */],
      "AccessPoints": [/* Access point details */]
    },
    {
      "PlaceId": "AQAAAGIA...",
      "PlaceType": "PointOfInterest",
      "Title": "ChargePoint",
      "Address": { /* Address details */ },
      "Position": [-123.11663, 49.28116],
      "Distance": 25,
    }
  ]
}
```

```

        "Categories": [/* Category details */],
        "AccessPoints": [/* Access point details */]
    },
    {
        "PlaceId": "AQAAAFUA9m...",
        "PlaceType": "PointOfInterest",
        "Title": "Zipcar",
        "Address": { /* Address details */ },
        "Position": [-123.11715, 49.28094],
        "Distance": 29,
        "Categories": [/* Category details */],
        "AccessPoints": [/* Access point details */]
    }
]
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/reverse-geocode?
key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
    "QueryPosition": [
      -123.11694,
      49.28126
    ],
    "MaxResults": "3"
  }'

```

AWS CLI

```

aws geo-places reverse-geocode --key ${YourKey} --query-position
"-123.11694,49.28126" --max-results "3"

```

Developer tips

For targeted results, use `IncludePlaceTypes` in the filter.

```

{
  "QueryPosition": [

```

```
-123.11694,  
49.28126  
],  
"Filter": { "IncludePlaceTypes": ["PointAddress"] }  
}
```

How to Reverse Geocode with a Political View

The Amazon Location Service allows you to specify a political view to ensure your application aligns with local regulations. This feature is useful when geocoding locations in disputed areas where place names or borders may vary depending on the political view.

Potential Use Cases

- **Adhere to local policies:** Ensure that place names and borders comply with legal requirements of the selected political view.

Examples

Reverse Geocode a Disputed Position

Sample Request

```
{  
  "QueryPosition": [  
    33.95876,  
    45.46824  
  ],  
  "PoliticalView": "RUS"  
}
```

Sample Response

```
{  
  "ResultItems": [  
    {  
      "PlaceId": "AQAAAD...",  
      "PlaceType": "Locality",  
      "Title": "Первомайский район, Южный федеральный округ, Россия",  
      "Address": {  
        "Label": "Первомайский район, Южный федеральный округ, Россия",
```

```
    "Country": {
      "Code2": "RU",
      "Code3": "RUS",
      "Name": "Россия"
    },
    "Region": {
      "Name": "Южный федеральный округ"
    },
    "SubRegion": {
      "Name": "Республика Крым"
    },
    "Locality": "Первомайский район"
  },
  "Position": [
    33.85939,
    45.7142
  ],
  "Distance": 0,
  "MapView": [
    33.52692,
    45.34303,
    34.12277,
    45.80953
  ],
  "PoliticalView": "RUS"
}
]
```

cURL

```
curl --request POST \  
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/reverse-geocode?\  
key=Your_Key' \  
  --header 'Content-Type: application/json' \  
  --data '{  
    "QueryPosition": [  
      33.95876,  
      45.46824  
    ],  
    "PoliticalView": "RUS"  
  }'
```

AWS CLI

```
aws geo-places reverse-geocode --key ${YourKey} --query-position 33.95876 45.46824
--political-view "RUS"
```

How to Reverse Geocode for Right Result

This guide provides methods to refine reverse geocoding results, ensuring that the returned data aligns closely with specific business needs. Using filters, users can narrow down results to more precisely match types like address points, streets, or localities.

Potential Use Cases

- **Restrict results for specific needs:** Use filters to retrieve only the most relevant information, such as exact addresses or broader locality data, based on business requirements.

Examples

Filter for a Point Address

By filtering for `PointAddress`, you can retrieve specific street addresses, enhancing location accuracy.

Sample Request

```
{
  "QueryPosition": [
    -97.721, 30.404
  ],
  "Filter": {
    "IncludePlaceTypes": [
      "PointAddress"
    ]
  }
}
```

Sample Response

```
{
  "ResultItems": [
    {
```

```
"PlaceId": "AQAAAGQA...",
"PlaceType": "PointAddress",
"Title": "11721 Domain Blvd, Austin, TX 78758-0051, United States",
"Address": {
  "Label": "11721 Domain Blvd, Austin, TX 78758-0051, United States",
  "Country": {
    "Code2": "US",
    "Code3": "USA",
    "Name": "United States"
  },
  "Region": {
    "Code": "TX",
    "Name": "Texas"
  },
  "SubRegion": {
    "Name": "Travis"
  },
  "Locality": "Austin",
  "District": "North Burnet",
  "PostalCode": "78758-0051",
  "Street": "Domain Blvd",
  "StreetComponents": [
    {
      "BaseName": "Domain",
      "Type": "Blvd",
      "TypePlacement": "AfterBaseName",
      "TypeSeparator": " ",
      "Language": "en"
    }
  ],
  "AddressNumber": "11721"
},
"Position": [
  -97.72087,
  30.404
],
"Distance": 5,
"MapView": [
  -97.72219,
  30.40273,
  -97.72057,
  30.40649
]
}
```



```
]
}
```

cURL

```
curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/reverse-geocode?
key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
  "{
    "QueryPosition": [
      -97.721, 30.404
    ],
    "Filter": {
      "IncludePlaceTypes": [
        "PointAddress"
      ]
    }
  }'
```

AWS CLI

```
aws geo-places reverse-geocode --key ${YourKey} --query-position -97.721 30.404 --
filter '{"IncludePlaceTypes": ["PointAddress"]}'
```

Filter for a Street

By filtering for `Street`, the API returns street-level data without specific address numbers.

Sample Request

```
{
  "QueryPosition": [
    -97.721, 30.404
  ],
  "Filter": {
    "IncludePlaceTypes": [
      "Street"
    ]
  }
}
```

Sample Response

```
{
  "ResultItems": [
    {
      "PlaceId": "AQAAAEkA...",
      "PlaceType": "Street",
      "Title": "Domain Blvd, Austin, TX 78758, United States",
      "Address": {
        "Label": "Domain Blvd, Austin, TX 78758, United States",
        "Country": {
          "Code2": "US",
          "Code3": "USA",
          "Name": "United States"
        },
        "Region": {
          "Code": "TX",
          "Name": "Texas"
        },
        "SubRegion": {
          "Name": "Travis"
        },
        "Locality": "Austin",
        "District": "North Burnet",
        "PostalCode": "78758",
        "Street": "Domain Blvd",
        "StreetComponents": [
          {
            "BaseName": "Domain",
            "Type": "Blvd",
            "TypePlacement": "AfterBaseName",
            "TypeSeparator": " ",
            "Language": "en"
          }
        ]
      },
      "Position": [
        -97.72103,
        30.40399
      ],
      "Distance": 3,
      "MapView": [
        -97.72219,
        30.40273,

```

```

        -97.72057,
        30.40649
    ]
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/reverse-geocode?
key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
  "{
    "QueryPosition": [
      -97.721, 30.404
    ],
    "Filter": {
      "IncludePlaceTypes": [
        "Street"
      ]
    }
  }'

```

AWS CLI

```

aws geo-places reverse-geocode --key ${YourKey} --query-position -97.721 30.404 --
filter '{"IncludePlaceTypes": ["Street"]}'

```

Filter for a Locality

By filtering for `Locality`, you can retrieve broader location data, including city names.

Sample Request

```

{
  "QueryPosition": [
    -97.721, 30.404
  ],
  "Filter": {
    "IncludePlaceTypes": [
      "Locality"
    ]
  }
}

```

```
    ]
  }
}
```

Sample Response

```
{
  "ResultItems": [
    {
      "PlaceId": "AQAAAD...",
      "PlaceType": "Locality",
      "Title": "Austin, TX, United States",
      "Address": {
        "Label": "Austin, TX, United States",
        "Country": {
          "Code2": "US",
          "Code3": "USA",
          "Name": "United States"
        },
        "Region": {
          "Code": "TX",
          "Name": "Texas"
        },
        "SubRegion": {
          "Name": "Travis"
        },
        "Locality": "Austin",
        "PostalCode": "78701"
      },
      "Position": [
        -97.74299,
        30.26759
      ],
      "Distance": 0,
      "MapView": [
        -98.06484,
        30.06592,
        -97.55914,
        30.51965
      ]
    }
  ]
}
```

cURL

```
curl --request POST \  
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/reverse-geocode?key=Your_Key' \  
  --header 'Content-Type: application/json' \  
  --data '{  
    "{  
      "QueryPosition": [  
        -97.721, 30.404  
      ],  
      "Filter": {  
        "IncludePlaceTypes": [  
          "Locality"  
        ]  
      }  
    }'  
'
```

AWS CLI

```
aws geo-places reverse-geocode --key ${YourKey} --query-position -97.721 30.404 --filter '{"IncludePlaceTypes": ["Locality"]}'
```

How to use Autocomplete

This topic demonstrates how to use the Autocomplete API to efficiently retrieve address suggestions based on user input. It covers two key applications: basic address auto completion, which suggests addresses as users type, and filtered auto completion, allowing users to narrow results by criteria such as region or specific location types. These examples provide practical guidance for integrating Autocomplete into applications to enhance search relevance and user experience.

Topics

- [How to complete an address](#)
- [How to complete an address with filters](#)

How to complete an address

The Autocomplete API enables you to complete partially typed addresses, providing standardized input for end users and improving efficiency during data entry.

Potential use cases

- **Complete an address during checkout:** Facilitate accurate and fast address input as customers type into a checkout form.
- **Country-specific address completion:** Restrict suggestions to a specific country for compliance or relevance to the user's location.

Examples

Complete address with minimal data

In this example, minimal data is returned to keep the response concise and cost-effective. Complete place details can be retrieved later using the PlaceId with the GetPlace API.

Sample request

```
{
  "QueryText": "100 McCullum Rd"
}
```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId": "AQAAAGEAr...",
      "PlaceType": "PointAddress",
      "Title": "United Kingdom, E3 5JB, London, 100 McCullum Road"
    }
  ],
  "QueryRefinements": []
}
```

cURL

```
curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/autocomplete?key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
    "QueryText": "100 McCullum Rd"
  }'
```

```
}'
```

AWS CLI

```
aws geo-places autocomplete --key ${YourKey} --query-text "100 McCullum Rd"
```

Complete address with address component

This example returns multiple address suggestions, allowing users to select a standardized address format for populating form fields accurately.

Sample request

```
{
  "QueryText": "100 McCullum Rd",
  "AdditionalFeatures": [
    "Core"
  ]
}
```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId": "AQAAAGE....",
      "PlaceType": "PointAddress",
      "Title": "United Kingdom, E3 5JB, London, 100 McCullum Road",
      "Address": {
        "Label": "100 McCullum Road, London, E3 5JB, United Kingdom",
        "Country": {
          "Code2": "GB",
          "Code3": "GBR",
          "Name": "United Kingdom"
        },
        "Region": {
          "Name": "England"
        },
        "SubRegion": {
          "Code": "LDN",
          "Name": "London"
        }
      },
    },
  ],
}
```

```
    "Locality": "London",
    "District": "Bow",
    "PostalCode": "E3 5JB",
    "Street": "McCullum Road",
    "StreetComponents": [
      {
        "BaseName": "McCullum",
        "Type": "Road",
        "TypePlacement": "AfterBaseName",
        "TypeSeparator": " ",
        "Language": "en"
      }
    ],
    "AddressNumber": "100"
  },
  "Language": "en",
  "Highlights": {
    "Title": [
      {
        "StartIndex": 32,
        "EndIndex": 35,
        "Value": "100"
      },
      {
        "StartIndex": 36,
        "EndIndex": 49,
        "Value": "McCullum Road"
      }
    ],
    "Address": {
      "Label": [
        {
          "StartIndex": 0,
          "EndIndex": 3,
          "Value": "100"
        },
        {
          "StartIndex": 4,
          "EndIndex": 17,
          "Value": "McCullum Road"
        }
      ],
      "Street": [
        {
```



```

        "StartIndex": 0,
        "EndIndex": 13,
        "Value": "McCullum Road"
      }
    ],
    "AddressNumber": [
      {
        "StartIndex": 0,
        "EndIndex": 3,
        "Value": "100"
      }
    ]
  }
},
...
],
"QueryRefinements": []
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/autocomplete?key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
    "QueryText": "100 McCullum Rd",
    "AdditionalFeatures": [
      "Core"
    ]
  }'

```

AWS CLI

```

aws geo-places autocomplete --key ${YourKey} --query-text "100 McCullum Rd" --
additional-features "Core"

```

How to complete an address with filters

The Autocomplete API enables you to complete partially typed addresses, facilitating quick and standardized address input for end users. By using the `AdditionalFeatures` parameter, you can customize the information provided in the response to meet specific requirements.

Potential use cases

- **Minimize data for cost efficiency:** Reduce response size and data transfer costs by requesting only essential address components when a follow-up query is anticipated.
- **Include necessary details for direct use:** Retrieve comprehensive address information to eliminate the need for additional queries.

Examples

Complete address with country filter

This example applies a country filter to refine results, enabling a standardized form of the user-inputted address to populate form fields accurately.

Sample request

```
{
  "QueryText": "100 McCullum Rd",
  "Filter": {
    "IncludeCountries": [
      "GBR"
    ]
  }
}
```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId": "AQAAAGEA...",
      "PlaceType": "PointAddress",
      "Title": "United Kingdom, E3 5JB, London, 100 McCullum Road",
      "Address": {
        "Label": "100 McCullum Road, London, E3 5JB, United Kingdom",

```

```
"Country": {
  "Code2": "GB",
  "Code3": "GBR",
  "Name": "United Kingdom"
},
"Region": {
  "Name": "England"
},
"SubRegion": {
  "Code": "LDN",
  "Name": "London"
},
"Locality": "London",
"District": "Bow",
"PostalCode": "E3 5JB",
"Street": "McCullum Road",
"StreetComponents": [
  {
    "BaseName": "McCullum",
    "Type": "Road",
    "TypePlacement": "AfterBaseName",
    "TypeSeparator": " ",
    "Language": "en"
  }
],
"AddressNumber": "100"
},
"Language": "en",
"Highlights": {
  "Title": [
    {
      "StartIndex": 32,
      "EndIndex": 35,
      "Value": "100"
    },
    {
      "StartIndex": 36,
      "EndIndex": 49,
      "Value": "McCullum Road"
    }
  ]
},
"Address": {
  "Label": [
    {
```

```

        "StartIndex": 0,
        "EndIndex": 3,
        "Value": "100"
      },
      {
        "StartIndex": 4,
        "EndIndex": 17,
        "Value": "McCullum Road"
      }
    ],
    "Street": [
      {
        "StartIndex": 0,
        "EndIndex": 13,
        "Value": "McCullum Road"
      }
    ],
    "AddressNumber": [
      {
        "StartIndex": 0,
        "EndIndex": 3,
        "Value": "100"
      }
    ]
  }
}
],
"QueryRefinements": []
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/autocomplete?key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
  "QueryText": "100 McCullum Rd",
  "Filter": {
    "IncludeCountries": [
      "GBR"
    ]
  }
}'

```

```
}  
}'
```

AWS CLI

```
aws geo-places autocomplete --key ${YourKey} --query-text "100 McCullum Rd" \  
--filter '{"IncludeCountries": ["GBR"]}'
```

Complete address with additional data

This example returns additional data, allowing the returned address details to be used without a follow-up query. The Core set of additional features is sufficient for this use case.

Sample request

```
{  
  "QueryText": "100 McCullum Rd",  
  "Filter": {  
    "IncludeCountries": [  
      "GBR"  
    ]  
  },  
  "AdditionalFeatures": [  
    "Core"  
  ]  
}
```

Sample response

```
{  
  "ResultItems": [  
    {  
      "PlaceId": "AQAAAGEA...",  
      "PlaceType": "PointAddress",  
      "Title": "United Kingdom, E3 5JB, London, 100 McCullum Road",  
      "Address": {  
        "Label": "100 McCullum Road, London, E3 5JB, United Kingdom",  
        "Country": {  
          "Code2": "GB",  
          "Code3": "GBR",  
          "Name": "United Kingdom"  
        }  
      }  
    },  
  ],  
}
```

```
    "Region": {
      "Name": "England"
    },
    "SubRegion": {
      "Code": "LDN",
      "Name": "London"
    },
    "Locality": "London",
    "District": "Bow",
    "PostalCode": "E3 5JB",
    "Street": "McCullum Road",
    "StreetComponents": [
      {
        "BaseName": "McCullum",
        "Type": "Road",
        "TypePlacement": "AfterBaseName",
        "TypeSeparator": " ",
        "Language": "en"
      }
    ],
    "AddressNumber": "100"
  },
  "Language": "en",
  "Highlights": {
    "Title": [
      {
        "StartIndex": 32,
        "EndIndex": 35,
        "Value": "100"
      },
      {
        "StartIndex": 36,
        "EndIndex": 49,
        "Value": "McCullum Road"
      }
    ],
    "Address": {
      "Label": [
        {
          "StartIndex": 0,
          "EndIndex": 3,
          "Value": "100"
        }
      ]
    }
  }
}
```

```

        "StartIndex": 4,
        "EndIndex": 17,
        "Value": "McCullum Road"
      }
    ],
    "Street": [
      {
        "StartIndex": 0,
        "EndIndex": 13,
        "Value": "McCullum Road"
      }
    ],
    "AddressNumber": [
      {
        "StartIndex": 0,
        "EndIndex": 3,
        "Value": "100"
      }
    ]
  }
}
],
"QueryRefinements": []
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/autocomplete?key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
    "QueryText": "100 McCullum Rd",
    "Filter": {
      "IncludeCountries": [
        "GBR"
      ]
    },
    "AdditionalFeatures": [
      "Core"
    ]
  }'

```

AWS CLI

```
aws geo-places autocomplete --key ${YourKey} --query-text "100 McCullum Rd" \  
--additional-features "Core" \  
--filter '{"IncludeCountries": ["GBR"]}'
```

How to use Get Place

This section contains a variety of how to guides and examples for how to use Get Place APIs.

Examples

- [How to get results for a place ID](#)
- [How to get contacts for a place ID](#)

How to get results for a place ID

The GetPlace API retrieves detailed information about a place using its unique PlaceId, allowing applications to access comprehensive details about specified locations.

Potential use cases

- **Retrieve store locations:** Use PlaceId to obtain details about various store locations for improved user experience and efficient resource management.
- **Access related place details:** Fetch details of places that were identified using other PlaceIds returned by previous queries.

Examples

Get place with a place ID

Sample request

```
https://places.geo.eu-central-1.amazonaws.com/v2/place/  
AQAAAFUAcRFHu947JATTY9gIGcfN1NVzD3UftkkI9ayJjtquaC7IquYz-_  
FFnJnzJSQ7JePd-  
sY0MSPA64V0w4aXlc-  
1B2fZLJKk6uoAMSgtwvwxyzg1fvPxFM9zXsx77EaLXar17F4gSPTyQ6fiEnj0b0ip0Xpn0oIsP?  
key=Your_Key
```


Sample response

```
{
  "PlaceId": "AQAAAFUAcFHu947JATTY9gIGcfNlNVzD3UftkkI9ayJjtquaC7IquYz-
_FFnJnzJSQ7JePd-sY0MSpA64V0w4aXlc-
1B2fZLJk6uoAMSgtwvwxyzg1fvPxFM9zXsx77EaLXarl17F4gSPTyQ6fiEnj0b0ip0Xpn0oIsP",
  "PlaceType": "PointOfInterest",
  "Title": "London Eye",
  "Address": {
    "Label": "London Eye, County Hall, Westminster Bridge Road, London, SE1 7,
United Kingdom",
    "Country": {
      "Code2": "GB",
      "Code3": "GBR",
      "Name": "United Kingdom"
    },
    "Region": {
      "Name": "England"
    },
    "SubRegion": {
      "Code": "LDN",
      "Name": "London"
    },
    "Locality": "London",
    "District": "Waterloo",
    "PostalCode": "SE1 7",
    "Street": "County Hall, Westminster Bridge Road",
    "StreetComponents": [
      {
        "BaseName": "County Hall, Westminster Bridge Road",
        "Language": "en-GB"
      }
    ]
  },
  "Position": [
    -0.11953,
    51.50336
  ],
  "Categories": [
    {
      "Id": "tourist_attraction",
      "Name": "Tourist Attraction",
      "LocalizedName": "Tourist Attraction",
      "Primary": true
    }
  ]
}
```

```

    },
    {
      "Id": "landmark-attraction",
      "Name": "Landmark-Attraction",
      "LocalizedName": "Landmark-Attraction",
      "Primary": false
    },
    {
      "Id": "residential_area-building",
      "Name": "Residential Area-Building",
      "LocalizedName": "Residential Area/Building",
      "Primary": false
    }
  ]
}

```

cURL

```

curl --request GET \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/place/
AQAAAFUAc rFHu947JATTY9gIGcfNlNVzD3UftkkI9ayJjtquaC7IquYz-
_FFnJnzJSQ7JePd-
sY0MSpA64V0w4aXLc-
lB2fZLJkk6uoAMSgtwvwxyzg1fvPxFM9zXsx77EaLXarl7F4gSPTyQ6fiEnj0b0ip0Xpn0oIsP?
key=Your_Key&language=en'

```

AWS CLI

```

export PLACEID=AQAAAFUAc rFHu947JATTY9gIGcfNlNVzD3UftkkI9ayJjtquaC7IquYz-
_FFnJnzJSQ7JePd-sY0MSpA64V0w4aXLc-
lB2fZLJkk6uoAMSgtwvwxyzg1fvPxFM9zXsx77EaLXarl7F4gSPTyQ6fiEnj0b0ip0Xpn0oIs
aws geo-places get-place --key ${YourKey} --place-id ${PLACEID}

```

How to get contacts for a place ID

The GetPlace API allows retrieval of contact information associated with a specific PlaceId, providing users with details such as phone numbers and website URLs.

Potential use cases

- **Obtain additional contact information for a saved PlaceId:** Enhance stored place details by accessing contact data.

- **Retrieve contact details for addresses from Autocomplete:** Use the PlaceId from the Autocomplete API to fetch specific contact details.

Examples

Get contacts for a place ID

Sample request

```
https://places.geo.eu-central-1.amazonaws.com/v2/place/
AQAAAFUAc rFHu947JATTY9gIGcfN1NVzD3UftkkI9ayJjtquaC7IquYz-
_FFnJnzJSQ7JePd-
sY0MSPa64V0w4aXLc-
1B2fZLJkk6uoAMSgtwvwxyzg1fvPxFM9zXsx77EaLXarl7F4gSPTyQ6fiEnj0b0ip0Xpn0oIsP?
additional-features=Contact&key=Your_Key
```

Sample response

```
{
  "PlaceId": "AQAAAFUAc rFHu947JATTY9gIGcfN1NVzD3UftkkI9ayJjtquaC7IquYz-
_FFnJnzJSQ7JePd-sY0MSPa64V0w4aXLc-
1B2fZLJkk6uoAMSgtwvwxyzg1fvPxFM9zXsx77EaLXarl7F4gSPTyQ6fiEnj0b0ip0Xpn0oIsP",
  "PlaceType": "PointOfInterest",
  "Title": "London Eye",
  "Address": {
    "Label": "London Eye, County Hall, Westminster Bridge Road, London, SE1 7,
United Kingdom",
    "Country": {
      "Code2": "GB",
      "Code3": "GBR",
      "Name": "United Kingdom"
    },
    "Region": {
      "Name": "England"
    },
    "SubRegion": {
      "Code": "LDN",
      "Name": "London"
    },
    "Locality": "London",
    "District": "Waterloo",
    "PostalCode": "SE1 7",
    "Street": "County Hall, Westminster Bridge Road",
```

```
    "StreetComponents": [
      {
        "BaseName": "County Hall, Westminster Bridge Road",
        "Language": "en-GB"
      }
    ],
  },
  "Position": [
    -0.11953,
    51.50336
  ],
  "Categories": [
    {
      "Id": "tourist_attraction",
      "Name": "Tourist Attraction",
      "LocalizedName": "Tourist Attraction",
      "Primary": true
    },
    {
      "Id": "landmark-attraction",
      "Name": "Landmark-Attraction",
      "LocalizedName": "Landmark-Attraction",
      "Primary": false
    },
    {
      "Id": "residential_area-building",
      "Name": "Residential Area-Building",
      "LocalizedName": "Residential Area/Building",
      "Primary": false
    }
  ],
  "Contacts": {
    "Phones": [
      {
        "Value": "+44870500600"
      },
      {
        "Value": "+448709908883"
      }
    ],
    "Websites": [
      {
        "Value": "http://www.londoneye.com"
      }
    ]
  }
}
```

```
    ]  
  }  
}
```

cURL

```
curl --request GET \  
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/place/  
AQAAAFUAcIFHu947JATTY9gIGcfN1NVzD3UftkkI9ayJjtquaC7IquYz-  
_FFnJnzJSQ7JePd-  
sY0MSpA64V0w4aXLC-  
1B2fZLJkk6uoAMSgtwvwxyzg1fvPxFM9zXsx77EaLXar17F4gSPTyQ6fiEnj0b0ip0Xpn0oIsP?  
key=Your_Key&additional-features=Contact&language=en'
```

AWS CLI

```
export PLACEID=AQAAAFUAcIFHu947JATTY9gIGcfN1NVzD3UftkkI9ayJjtquaC7IquYz-  
_FFnJnzJSQ7JePd-sY0MSpA64V0w4aXLC-  
1B2fZLJkk6uoAMSgtwvwxyzg1fvPxFM9zXsx77EaLXar17F4gSPTyQ6fiEnj0b0ip0Xpn0oIsP  
aws geo-places get-place --key ${YourKey} --place-id ${PLACEID} --additional-  
features "Contact" --language "en"
```

How to use SearchNearby

This section contains a variety of how to guides and examples for how to use SearchNearby.

Examples

- [How to search nearby from a position](#)
- [How to search nearby places based on category](#)

How to search nearby from a position

The SearchNearby API enables querying for all nearby places and points of interest (POI) without entering any specific text. Users can explore neighborhoods, discover POIs, and more using this API. It requires a `QueryPosition`, which can represent a device's location, IP-based position, or the map viewport center. Alternatively, users can specify a city or place to bias results based on the geocoordinates of that location.

Potential use cases

- **Explore nearby POIs:** View all points of interest near the current position.
- **Explore nearby places:** View all locations or places near a given position.

Examples

Search nearby from a position

In this example, the search is conducted from a position in Dubai with latitude 25.26951 and longitude 55.30884.

Sample request

```
{
  "QueryPosition": [
    55.30884,
    25.26951
  ]
}
```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId":
      "AQAAAFUAUFIBxnlf3pSzBCdPj9dEL5bz8EgiVH_BLPEUybERuzP-9en0Ejf8MXfy_k47Xy5hstAlwY7T-
      jPmx8aV_acceKFqjV818CfpZIV5Cm15RbedrQq3KeeXoWjdrVAX4r97aXB0Iq2zZ0arnN0Cu5CfRX1c-
      zfL",
      "PlaceType": "PointOfInterest",
      "Title": "###",
      "Address": {
        "Label": "##### ,### #### ,##### ## ##### #### ,###
##### ",
        "Country": {
          "Code2": "AE",
          "Code3": "ARE",
          "Name": "##### "
        },
        "SubRegion": {
          "Name": "###"
        }
      }
    }
  ]
}
```

```

    },
    "Locality": "###",
    "District": "####",
    "Street": "##### ## ##### ####",
    "StreetComponents": [
      {
        "BaseName": "##### ## #####",
        "Type": "####",
        "TypePlacement": "BeforeBaseName",
        "TypeSeparator": " ",
        "Language": "ar"
      }
    ]
  },
  "Position": [
    55.30884,
    25.26951
  ],
  "Distance": 0,
  "Categories": [
    {
      "Id": "department_store",
      "Name": "Department Store",
      "LocalizedName": "##### ###",
      "Primary": true
    }
  ],
  "BusinessChains": [
    {
      "Name": "HEMA",
      "Id": "HEMA"
    }
  ]
},
{
  "PlaceId": "AQAAADgAxiVfwtSjVviV4nQSzJ-gbGj8-
AuSr5wjVge1ZfCgDnghr1r2CofXYy_0ew9TteEeGxFC5U1vj7DCXdJX2X156PaHhBWiABhNkCTx3Rc3PQTbHzyo5Az7_
",
  "PlaceType": "Locality",
  "Title": "##### ##### ##### ,###",
  "Address": {
    "Label": "##### ##### ##### ,###",
    "Country": {
      "Code2": "AE",
      "Code3": "ARE",

```

```

        "Name": "#####"
      },
      "SubRegion": {
        "Name": "###"
      },
      "Locality": "###"
    },
    "Position": [
      55.30884,
      25.26951
    ],
    "Distance": 0,
    "MapView": [
      54.64906,
      24.62308,
      55.7371,
      25.36995
    ]
  ]
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/search-nearby?
key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
  "QueryPosition": [
    55.30884,
    25.26951
  ],
  "MaxResults": 2
}'

```

AWS CLI

```

aws geo-places search-nearby --key ${YourKey} \
--query-position 55.30884 25.26951 \
--max-results 2

```


How to search nearby places based on category

The SearchNearby API enables querying for points of interest (POI) with the inclusion or exclusion of specified categories. This can help users explore neighborhoods, discover local POIs, and more. The API requires a `QueryPosition`, which can be based on a device's location, IP position, or the center of the map viewport. Alternatively, users can specify a city or place, and the application will bias results based on that location's coordinates.

To learn more about supported categories, see [the section called "Categories filters"](#).

Potential use cases

- **Explore local facilities:** Find available facilities within a neighborhood.
- **Discover tourist attractions:** Identify tourist spots within a city.
- **Plan travel in a city:** Organize travel around different POIs within a chosen city.

Examples

Include category

Sample request

```
{
  "QueryPosition": [
    4.35609,
    50.84439
  ],
  "Filter": {
    "IncludeCategories": ["airport"]
  }
}
```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId":
        "AQAAAFUAXV4cfKzNk3Tgth2C1JdH2LLZYb8Z3FSoZKoxkgNzR71Vn4IFc1s9xECqUqYk8JJpG5ZPmaq-31ycT8U066JaG00DcUYk9h-CBi3awt5_W83b56ZKsDpCGM1oEV",
      "PlaceType": "PointOfInterest",
    }
  ]
}
```

```

    "Title": "Brussels Airport",
    "Address": {
      "Label": "Brussels Airport, A201, 1930 Zaventem, België",
      "Country": {
        "Code2": "BE",
        "Code3": "BEL",
        "Name": "België"
      },
      "Region": {
        "Code": "VLG",
        "Name": "Vlaanderen"
      },
      "SubRegion": {
        "Name": "Vlaams Brabant"
      },
      "Locality": "Zaventem",
      "PostalCode": "1930",
      "Street": "A201",
      "StreetComponents": [
        {
          "BaseName": "A201",
          "Language": "nl"
        }
      ]
    },
    "Position": [
      4.47767,
      50.89452
    ],
    "Distance": 10191,
    "Categories": [
      {
        "Id": "airport",
        "Name": "Airport",
        "LocalizedName": "Luchthaven",
        "Primary": true
      }
    ]
  },
  {
    "PlaceId":
    "AQAAAFUAbgGbaTvrkQtGzse0FSHl3FWyDI14vsBhcZzXwa_I0rhEJN00qMnftJGbqdyKHZ3svn-
    oCyBgpVtwpKlhhkrYmXa0tzf4TASZFMXkFXhf9LScE8xRl3gBJWyZ4E4erzebS_YhFFUztZUG0zCB-8zQUHvLHJya",
    "PlaceType": "PointOfInterest",

```

```
    "Title": "Internationale Luchthaven Antwerpen",
    "Address": {
      "Label": "Internationale Luchthaven Antwerpen, Luchthavenlei 1, 2100
Antwerpen, België",
      "Country": {
        "Code2": "BE",
        "Code3": "BEL",
        "Name": "België"
      },
      "Region": {
        "Code": "VLG",
        "Name": "Vlaanderen"
      },
      "SubRegion": {
        "Name": "Antwerpen"
      },
      "Locality": "Antwerpen",
      "District": "Deurne",
      "PostalCode": "2100",
      "Street": "Luchthavenlei",
      "StreetComponents": [
        {
          "BaseName": "Luchthaven",
          "Type": "lei",
          "TypePlacement": "AfterBaseName",
          "TypeSeparator": "",
          "Language": "nl"
        }
      ],
      "AddressNumber": "1"
    },
    "Position": [
      4.45083,
      51.18867
    ],
    "Distance": 38852,
    "Categories": [
      {
        "Id": "airport",
        "Name": "Airport",
        "LocalizedName": "Luchthaven",
        "Primary": true
      }
    ]
  ]
}
```

```

    }
  ]
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/search-nearby?
key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
  "QueryPosition": [
    4.35609,
    50.84439
  ],
  "Filter": {
    "IncludeCategories": ["airport"]
  },
  "MaxResults": 2
}'

```

AWS CLI

```

aws geo-places search-nearby --key ${YourKey} \
  --query-position 4.35609 50.84439 \
  --filter '{"IncludeCategories": ["airport"]}' \
  --max-results 2

```

Exclude category

Sample request

```

{
  "QueryPosition": [
    4.35609,
    50.84439
  ],
  "Filter": {
    "ExcludeCategories": ["airport"]
  }
}

```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId": "AQAAAFUAdkKiPCNZ7PPUoB5NzbozQAGZ42fmhikKkrAM44iFYQWkWQ-
D00QuRL-3na48xwwgZThaPVSbhrE4X83TqfF1PDnSNRYJQ4MgL-
pH7g2uoBkP8t43NDLyhVQw0ZwbW_6VEyMtM2q5DQ3f1vrEJiQm8A0teY1YT",
      "PlaceType": "PointOfInterest",
      "Title": "Socialbrands Module 2",
      "Address": {
        "Label": "Socialbrands Module 2, Albertinaplein, 1000 Brussel,
België",
        "Country": {
          "Code2": "BE",
          "Code3": "BEL",
          "Name": "België"
        },
        "Region": {
          "Code": "BRU",
          "Name": "Brussel"
        },
        "SubRegion": {
          "Name": "Brussel"
        },
        "Locality": "Brussel",
        "District": "Koningswijk",
        "PostalCode": "1000",
        "Street": "Albertinaplein",
        "StreetComponents": [
          {
            "BaseName": "Albertina",
            "Type": "plein",
            "TypePlacement": "AfterBaseName",
            "TypeSeparator": "",
            "Language": "nl"
          }
        ]
      },
      "Position": [
        4.35609,
        50.84439
      ],
      "Distance": 0,
    }
  ]
}
```

```

    "Categories": [
      {
        "Id": "commercial_services",
        "Name": "Commercial Services",
        "LocalizedName": "Commerciële diensten",
        "Primary": true
      }
    ],
    "Contacts": {
      "Websites": [
        {
          "Value": "https://oneread.net"
        }
      ]
    },
    "AccessPoints": [
      {
        "Position": [
          4.35609,
          50.84439
        ]
      }
    ]
  },
  {
    "PlaceId": "AQAAAFUAuUkEuMtd2jvHT-
I_4lvx5qlsh5i1gsqw0pN3aAvCnQX7IdDnbDqDMNo3Ia7b7iNJYJhlge266p1pXUWwz4yDZpRkPIiIBUE9Rv7P_iuGqj",
    "PlaceType": "PointOfInterest",
    "Title": "Barman Privé",
    "Address": {
      "Label": "Barman Privé, Albertinaplein, 1000 Brussel, België",
      "Country": {
        "Code2": "BE",
        "Code3": "BEL",
        "Name": "België"
      },
      "Region": {
        "Code": "BRU",
        "Name": "Brussel"
      },
      "SubRegion": {
        "Name": "Brussel"
      },
      "Locality": "Brussel",

```

```
    "District": "Koningswijk",
    "PostalCode": "1000",
    "Street": "Albertinaplein",
    "StreetComponents": [
      {
        "BaseName": "Albertina",
        "Type": "plein",
        "TypePlacement": "AfterBaseName",
        "TypeSeparator": "",
        "Language": "nl"
      }
    ]
  },
  "Position": [
    4.35609,
    50.84439
  ],
  "Distance": 0,
  "Categories": [
    {
      "Id": "catering_and_other_food_services",
      "Name": "Catering and Other Food Services",
      "LocalizedNames": "Catering- en horecadiensten",
      "Primary": true
    }
  ],
  "Contacts": {
    "Phones": [
      {
        "Value": "+32476891634"
      }
    ]
  },
  "OpeningHours": [
    {
      "Display": [
        "ma-zo: 00:00 - 24:00"
      ],
      "OpenNow": true,
      "Components": [
        {
          "OpenTime": "T000000",
          "OpenDuration": "PT24H00M",
          "Recurrence": "FREQ:DAILY;BYDAY:MO,TU,WE,TH,FR,SA,SU"
        }
      ]
    }
  ]
}
```

```

    ]
  }
],
"AccessPoints": [
  {
    "Position": [
      4.35609,
      50.84439
    ]
  }
]
}
]
}
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/search-nearby?
key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
  "QueryPosition": [
    4.35609,
    50.84439
  ],
  "Filter": {
    "ExcludeCategories": ["airport"]
  }
}'

```

AWS CLI

```

aws geo-places search-nearby --key ${YourKey} \
--query-position 4.35609 50.84439 \
--filter '{"ExcludeCategories": ["airport"]}' \
--max-results 2

```

How to use SearchText

This section contains a variety of how to guides and examples for how to use SearchText APIs.

Examples

- [How to search for a place, POI, or business using a name](#)
- [How to search for a place using contact information](#)

How to search for a place, POI, or business using a name

The SearchText API allows users to search for a place, POI, or business by name, using free text input. Results can be refined by setting a bias position, which may be based on device location, IP position, or map viewport center. Alternatively, users can provide a specific city or place, with results biased based on the provided geocoordinates.

Potential use cases

- **Locate a place by name:** Retrieve locations based on place names, such as "Gas Town".
- **Find points of interest (POIs):** Search for places of interest by name, such as "Stanley Park".
- **Search by business name:** Locate businesses by name, like "Starbucks".

Examples

Search by place name

Sample request

```
{
  "QueryText": "Gas Town",
  "BiasPosition": [
    -123.11336,
    49.26038
  ]
}
```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId": "AQAAADgAUkZ5N7UW6ZWuIFWY0CuY-
I5EnunJHbMaHP0JWtQWhBn305gbx2SQT1Ub6k0J6vpo2tjY31Idqfyw5tJrBkCGDZLiS7uDVI0YpW0bN49A7wcvFyk5-
R7_iw",
```

```

    "PlaceType": "District",
    "Title": "Gastown, Vancouver, BC, Canada",
    "Address": {
      "Label": "Gastown, Vancouver, BC, Canada",
      "Country": {
        "Code2": "CA",
        "Code3": "CAN",
        "Name": "Canada"
      },
      "Region": {
        "Code": "BC",
        "Name": "British Columbia"
      },
      "SubRegion": {
        "Name": "Metro Vancouver"
      },
      "Locality": "Vancouver",
      "District": "Gastown",
      "PostalCode": "V6B"
    },
    "Position": [
      -123.10647,
      49.28363
    ],
    "Distance": 2633,
    "MapView": [
      -123.11193,
      49.28141,
      -123.10217,
      49.28648
    ]
  ]
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/search-text?key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
    "QueryText": "Gas Town",

```

```
"BiasPosition": [
  -123.11336,
  49.26038
]
```

AWS CLI

```
aws geo-places search-text --key YOUR_KEY --query-text "Gas Town" --bias-position
-123.11336 49.26038
```

Search by POI name

Sample request

```
{
  "QueryText": "Stanley Park",
  "BiasPosition": [
    -123.11336,
    49.26038
  ]
}
```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId": "AQAAAFUA-fkBnqPFDENTCLU83ocqzf4hnC2ZGVoPoW_5dYABzbjEVtmq3uUeB3lGhru8Rf9BvwPQiozf85NBYqcoGfezbd1PVMkQF",
      "PlaceType": "PointOfInterest",
      "Title": "Stanley Park",
      "Address": {
        "Label": "Stanley Park, Stanley Park Dr, Vancouver, BC V6G, Canada",
        "Country": {
          "Code2": "CA",
          "Code3": "CAN",
          "Name": "Canada"
        },
        "Region": {
          "Code": "BC",
          "Name": "British Columbia"
        }
      }
    }
  ]
}
```

```
    },
    "SubRegion": {
      "Name": "Metro Vancouver"
    },
    "Locality": "Vancouver",
    "District": "Stanley Park",
    "PostalCode": "V6G",
    "Street": "Stanley Park Dr",
    "StreetComponents": [
      {
        "BaseName": "Stanley Park",
        "Type": "Dr",
        "TypePlacement": "AfterBaseName",
        "TypeSeparator": " ",
        "Language": "en"
      }
    ]
  },
  "Position": [
    -123.13593,
    49.29716
  ],
  "Distance": 4405,
  "Categories": [
    {
      "Id": "park-recreation_area",
      "Name": "Park-Recreation Area",
      "LocalizedName": "Park-Recreation Area",
      "Primary": true
    },
    {
      "Id": "tourist_attraction",
      "Name": "Tourist Attraction",
      "LocalizedName": "Tourist Attraction",
      "Primary": false
    }
  ]
}
]
```

cURL

```
curl --request POST \  
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/search-text?key=Your_Key'  
 \  
  --header 'Content-Type: application/json' \  
  --data '{  
    "QueryText": "Stanley Park",  
    "BiasPosition": [  
      -123.11336,  
      49.26038  
    ]  
  }'
```

AWS CLI

```
aws geo-places search-text --key YOURKEY --query-text "Stanley Park" --bias-  
position -123.11336 49.26038
```

Search by business name

Sample request

```
{  
  "QueryText": "Amazon YVR11",  
  "BiasPosition": [  
    -123.11336,  
    49.26038  
  ]  
}
```

Sample response

```
{  
  "ResultItems": [  
    {  
      "PlaceId":  
"AQAAAFUAMMThb0tL3UEi8q0XK2If43Wty4dB7u6NwnBeskCf4AK7jSvY48P6FGnDWT8aLCdG3FW3csX1p_6P64Rng0  
ijl3eMMoF",  
      "PlaceType": "PointOfInterest",  
      "Title": "Amazon YVR11",
```

```
"Address": {
  "Label": "Amazon YVR11, 510 W Georgia St, Vancouver, BC V6B 0M3,
Canada",
  "Country": {
    "Code2": "CA",
    "Code3": "CAN",
    "Name": "Canada"
  },
  "Region": {
    "Code": "BC",
    "Name": "British Columbia"
  },
  "SubRegion": {
    "Name": "Metro Vancouver"
  },
  "Locality": "Vancouver",
  "District": "Downtown Vancouver",
  "PostalCode": "V6B 0M3",
  "Street": "W Georgia St",
  "StreetComponents": [
    {
      "BaseName": "Georgia",
      "Type": "St",
      "TypePlacement": "AfterBaseName",
      "TypeSeparator": " ",
      "Prefix": "W",
      "Language": "en"
    }
  ],
  "AddressNumber": "510"
},
"Position": [
  -123.11694,
  49.28126
],
"Distance": 2336,
"Categories": [
  {
    "Id": "business_facility",
    "Name": "Business Facility",
    "LocalizedName": "Business Facility",
    "Primary": true
  }
],
```

```

    "AccessPoints": [
      {
        "Position": [
          -123.11656,
          49.28151
        ]
      }
    ]
  }
]
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/search-text?key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
    "QueryText": "Amazon YVR11",
    "BiasPosition": [
      -123.11336,
      49.26038
    ]
  }'

```

AWS CLI

```

aws geo-places search-text --key ${YourKey} --query-text "Amazon YVR11" --bias-
position -123.11336 49.26038

```

How to search for a place using contact information

The SearchText API allows users to search for a place using a phone number, supporting both international and local formats. Users can bias results by setting a position based on device location, IP address, or map viewport center, or by specifying a city or place to refine results based on geocoordinates.

Potential use cases

- **Locate a place with a phone number:** Find a POI by using its contact number to retrieve its address.

Examples

Search using phone number

In this example, the Vancouver Aquarium is searched using its phone number "+1 778-655-9554" from a bias position in Vancouver, BC.

Sample request

```
{
  "QueryText": "+1 778-655-9554",
  "BiasPosition": [
    -123.11336,
    49.26038
  ]
}
```

Sample response

```
{
  "ResultItems": [
    {
      "PlaceId":
      "AQAAAFUA_ujqM62qn8Y6qsmfrcP93sw2H0fCtQs0AxqPx5TpYZrU4h1bJTGsCF_YfPKzGlt8KHdz_ict0bel68ueNf",
      "PlaceType": "PointOfInterest",
      "Title": "Vancouver Aquarium",
      "Address": {
        "Label": "Vancouver Aquarium, 834 Avison Way, Vancouver, BC V6G,
Canada",
        "Country": {
          "Code2": "CA",
          "Code3": "CAN",
          "Name": "Canada"
        },
        "Region": {
          "Code": "BC",
          "Name": "British Columbia"
        }
      }
    }
  ]
}
```



```
    },
    "SubRegion": {
      "Name": "Metro Vancouver"
    },
    "Locality": "Vancouver",
    "District": "Stanley Park",
    "PostalCode": "V6G",
    "Street": "Avison Way",
    "StreetComponents": [
      {
        "BaseName": "Avison",
        "Type": "Way",
        "TypePlacement": "AfterBaseName",
        "TypeSeparator": " ",
        "Language": "en"
      }
    ],
    "AddressNumber": "834"
  },
  "Position": [
    -123.13049,
    49.30013
  ],
  "Distance": 4591,
  "Categories": [
    {
      "Id": "aquarium",
      "Name": "Aquarium",
      "LocalizedName": "Aquarium",
      "Primary": true
    },
    {
      "Id": "tourist_attraction",
      "Name": "Tourist Attraction",
      "LocalizedName": "Tourist Attraction",
      "Primary": false
    }
  ]
]
}
```

cURL

```
curl --request POST \  
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/search-text?key=Your_Key'  
 \  
  --header 'Content-Type: application/json' \  
  --data '{  
    "QueryText": "+1 778-655-9554",  
    "BiasPosition": [  
      -123.11336,  
      49.26038  
    ]  
  }'
```

AWS CLI

```
aws geo-places search-text --key #{YourKey} --query-text "+1 778-655-9554" --bias-  
position -123.11336 49.26038
```

How to use Suggest

This section contains a variety of how to guides and examples for how to use Suggest APIs.

Topics

- [How to predict suggestions based on input](#)
- [How to get results for a partially typed or misspelled query](#)

How to predict suggestions based on input

The Suggest API enables applications to complete user queries for places or categories of results. These suggestions can be used directly or refined further with the SearchText API to retrieve additional details and results.

Potential use cases

- **Restaurant or park lookup:** Locate restaurants, parks, or other places near a specific position.

Examples

Look up restaurants near a position

Sample request

```
{
  "QueryText": "restaura",
  "BiasPosition": [-124.81035775620968, 49.234628873773985],
  "AdditionalFeatures": ["Core"]
}
```

Sample response

```
{
  "ResultItems": [
    {
      "Title": "Restaurant",
      "SuggestResultItemType": "Query",
      "Query": {
        "QueryId": "AQAAAHgAw1x7...",
        "QueryType": "Category"
      },
      "Highlights": {
        "Title": [
          {
            "StartIndex": 0,
            "EndIndex": 8,
            "Value": "Restaura"
          }
        ]
      }
    }
  ],
  "QueryRefinements": []
}
```

cURL

```
curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/suggest?key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
```

```
"QueryText": "restaura",
"BiasPosition": [-124.81035775620968, 49.234628873773985],
"AdditionalFeatures": ["Core"]
}'
```

AWS CLI

```
aws geo-places suggest --key ${YourKey} \
--query-text "restaura" \
--bias-position -124.81035775620968 49.234628873773985 \
--additional-features "Core"
```

Look up locations of a chain near a position

Sample request

```
{
  "QueryText": "Domin",
  "BiasPosition": [-124.81035775620968, 49.234628873773985],
  "AdditionalFeatures": ["Core"]
}
```

Sample response

```
{
  "ResultItems": [
    {
      "Title": "Domino's",
      "SuggestResultItemType": "Query",
      "Query": {
        "QueryId": "AQAAAHUA4UNp...",
        "QueryType": "BusinessChain"
      },
      "Highlights": {
        "Title": [
          {
            "StartIndex": 0,
            "EndIndex": 5,
            "Value": "Domin"
          }
        ]
      }
    }
  ]
}
```

```

    }
  ],
  "QueryRefinements": []
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/suggest?key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
  "QueryText": "Domin",
  "BiasPosition": [-124.81035775620968, 49.234628873773985],
  "AdditionalFeatures": ["Core"]
}'

```

AWS CLI

```

aws geo-places suggest --key ${YourKey} \
  --query-text "Domin" \
  --bias-position -124.81035775620968 49.234628873773985 \
  --additional-features "Core"

```

Complete a query using a QueryId

Sample request (SearchText)

```

{
  "QueryId": "<QueryId from Suggest API>",
  "AdditionalFeatures": ["Core"]
}

```

Sample response

```

{
  "ResultItems": [
    {
      "PlaceId": "AQAAAFUAAb3YiiM...",
      "PlaceType": "PointOfInterest",
      "Title": "Domino's",
      "Address": {

```

```
"Label": "Domino's, 233 Shelly Rd, Parksville, BC V9P, Canada",
"Country": {
  "Code2": "CA",
  "Code3": "CAN",
  "Name": "Canada"
},
"Region": {
  "Code": "BC",
  "Name": "British Columbia"
},
"SubRegion": {
  "Name": "Nanaimo"
},
"Locality": "Parksville",
"PostalCode": "V9P",
"Street": "Shelly Rd",
"StreetComponents": [
  {
    "BaseName": "Shelly",
    "Type": "Rd",
    "TypePlacement": "AfterBaseName",
    "TypeSeparator": " ",
    "Language": "en"
  }
],
"AddressNumber": "233"
},
"Position": [
  -124.29334,
  49.31669
],
"Distance": 38602,
"Categories": [
  {
    "Id": "fast_food",
    "Name": "Fast Food",
    "LocalizedName": "Fast Food",
    "Primary": true
  },
  {
    "Id": "take_out_and_delivery_only",
    "Name": "Take Out and Delivery Only",
    "LocalizedName": "Take Out & Delivery Only",
    "Primary": false
  }
]
```

```

    }
  ],
  "FoodTypes": [
    {
      "LocalizedName": "Pizza",
      "Id": "pizza",
      "Primary": true
    }
  ],
  "BusinessChains": [
    {
      "Name": "Domino's",
      "Id": "Domino's"
    }
  ],
  "Contacts": {
    "Phones": [
      {
        "Value": "+12502489296"
      }
    ],
    "Websites": [
      {
        "Value": "http://pizza.dominos.ca/Parksville-British-
Columbia-10039"
      }
    ]
  },
  "OpeningHours": [
    {
      "Display": [
        "Mon-Fri: 11:00 - 24:00",
        "Sat, Sun: 00:00 - 01:00, 11:00 - 24:00"
      ],
      "OpenNow": true,
      "Components": [
        {
          "OpenTime": "T110000",
          "OpenDuration": "PT13H00M",
          "Recurrence": "FREQ:DAILY;BYDAY:MO,TU,WE,TH,SU"
        },
        {
          "OpenTime": "T110000",
          "OpenDuration": "PT14H00M",

```

```

        "Recurrence": "FREQ:DAILY;BYDAY:FR,SA"
      }
    ]
  },
  "AccessPoints": [
    {
      "Position": [
        -124.29319,
        49.31669
      ]
    }
  ]
},
{
  "PlaceId": "AQAAAFUAY3bj...",
  "PlaceType": "PointOfInterest",
  "Title": "Domino's",
  "Address": {
    "Label": "Domino's, 215 Port Augusta St, Comox, BC V9M, Canada",
    "Country": {
      "Code2": "CA",
      "Code3": "CAN",
      "Name": "Canada"
    },
    "Region": {
      "Code": "BC",
      "Name": "British Columbia"
    },
    "SubRegion": {
      "Name": "Comox Valley"
    },
    "Locality": "Comox",
    "PostalCode": "V9M",
    "Street": "Port Augusta St",
    "StreetComponents": [
      {
        "BaseName": "Port Augusta",
        "Type": "St",
        "TypePlacement": "AfterBaseName",
        "TypeSeparator": " ",
        "Language": "en"
      }
    ]
  ]
},

```



```
    "AddressNumber": "215"
  },
  "Position": [
    -124.92492,
    49.67378
  ],
  "Distance": 49528,
  "Categories": [
    {
      "Id": "fast_food",
      "Name": "Fast Food",
      "LocalizedName": "Fast Food",
      "Primary": true
    },
    {
      "Id": "take_out_and_delivery_only",
      "Name": "Take Out and Delivery Only",
      "LocalizedName": "Take Out & Delivery Only",
      "Primary": false
    }
  ],
  "FoodTypes": [
    {
      "LocalizedName": "Pizza",
      "Id": "pizza",
      "Primary": true
    }
  ],
  "BusinessChains": [
    {
      "Name": "Domino's",
      "Id": "Domino's"
    }
  ],
  "Contacts": {
    "Phones": [
      {
        "Value": "+17784310222"
      }
    ],
    "Websites": [
      {
        "Value": "http://pizza.dominos.ca/comox-british-columbia-39035"
      }
    ]
  }
}
```

```

        }
      ]
    },
    "OpeningHours": [
      {
        "Display": [
          "Mon-Thu: 10:30 - 23:00",
          "Fri: 10:30 - 24:00",
          "Sat: 00:00 - 01:00, 10:30 - 24:00",
          "Sun: 00:00 - 01:00, 10:30 - 23:00"
        ],
        "OpenNow": true,
        "Components": [
          {
            "OpenTime": "T103000",
            "OpenDuration": "PT12H30M",
            "Recurrence": "FREQ:DAILY;BYDAY:MO,TU,WE,TH,SU"
          },
          {
            "OpenTime": "T103000",
            "OpenDuration": "PT14H30M",
            "Recurrence": "FREQ:DAILY;BYDAY:FR,SA"
          }
        ]
      }
    ],
    "AccessPoints": [
      {
        "Position": [
          -124.92518,
          49.67387
        ]
      }
    ]
  },
  ...
],
"QueryRefinements": []
}

```

cURL

```
curl --request POST \
```

```
--url 'https://places.geo.eu-central-1.amazonaws.com/v2/search-text?key=Your_Key' \
--header 'Content-Type: application/json' \
--data '{
  "QueryId": "<QueryId from Suggest API>",
  "AdditionalFeatures": ["Core"]
}'
```

AWS CLI

```
aws geo-places suggest --key ${YourKey} \
--query-id "<QueryId from Suggest API>" \
--additional-features "Core"
```

Developer tips

Use filters such as `Filter.IncludeCountries` or `Filter.BoundingBox` with `BiasPosition`. These filters apply to subsequent queries using the `QueryId`.

```
{
  "QueryText": "Domin",
  "BiasPosition": [-124.81035775620968, 49.234628873773985],
  "Filter": {
    "IncludeCountries": ["CAN"]
  }
}
```

How to get results for a partially typed or misspelled query

The Suggest API enables applications to complete user queries for places or categories of results. These results are sorted from most likely match to less likely matches, allowing the API to resolve incomplete or misspelled words.

Potential use cases

- **Complete a partially typed point of interest query:** Assists users by providing suggestions based on partially typed or incorrect entries.

Examples

Look up a misspelled point of interest

Sample request

```
{
  "QueryText": "Effel tow",
  "Filter": {
    "Circle": {
      "Radius": 10000,
      "Center": [2.3431932014695382, 48.858844492141145]
    },
    "IncludeCountries": ["FRA"]
  },
  "AdditionalFeatures": ["Core"]
}
```

Sample response

```
{
  "ResultItems": [
    {
      "Title": "Tour Eiffel (Eiffel Tower)",
      "SuggestResultItemType": "Place",
      "Place": {
        "PlaceId":
        "AQAAAFUAGLJG0dE0XpI2ASemjn8c0XxgnJ8rPln0xKgWuuKhjwxirWjT7MRwJchYoMb7qE6hFVFPsGw-
        JAS3Ts1t_x23T6GqTY2MKEIx3PPlu0Y5ZMmltJFsGiNXoBHHKiSemjnnB-
        uawTqKz2oqKxLLBfqPcJcd_fFT",
        "PlaceType": "PointOfInterest",
        "Address": {
          "Label": "Tour Eiffel, 5 Avenue Anatole France, 75007 Paris,
          France",
          "Country": {"Code2": "FR", "Code3": "FRA", "Name": "France"},
          "Region": {"Code": "IDF", "Name": "Île-de-France"},
          "SubRegion": {"Name": "Paris"},
          "Locality": "Paris",
          "District": "7e Arrondissement",
          "PostalCode": "75007",
          "Street": "Avenue Anatole France",
          "StreetComponents": [{"BaseName": "Anatole France", "Type":
          "Avenue", "Language": "fr"}],

```

```

        "AddressNumber": "5"
      },
      "Position": [2.2945, 48.85824],
      "Distance": 3563,
      "Categories": [
        {"Id": "historical_monument", "Name": "Historical Monument",
"LocalizedName": "Monument historique", "Primary": true},
        {"Id": "landmark-attraction", "Name": "Landmark-Attraction",
"LocalizedName": "Lieu d'intérêt/Attraction", "Primary": false},
        {"Id": "tourist_attraction", "Name": "Tourist Attraction",
"LocalizedName": "Attraction touristique", "Primary": false},
        {"Id": "sports_complex-stadium", "Name": "Sports Complex-
Stadium", "LocalizedName": "Stade ou complexe sportif", "Primary": false}
      ]
    },
    "Highlights": {"Title": [{"StartIndex": 13, "EndIndex": 23, "Value":
"Eiffel Tow"}]}
  }
},
"QueryRefinements": []
}

```

cURL

```

curl --request POST \
  --url 'https://places.geo.eu-central-1.amazonaws.com/v2/suggest?key=Your_Key' \
  --header 'Content-Type: application/json' \
  --data '{
"QueryText": "Effel tow",
"Filter": {
  "Circle": {"Radius": 10000, "Center": [2.3431932014695382, 48.858844492141145]},
  "IncludeCountries": ["FRA"]}
},
"AdditionalFeatures": ["Core"]
}'

```

AWS CLI

```

aws geo-places suggest --key ${YourKey} \
--query-text "Effel tow" \
--filter '{"Circle": {"Radius": 1000, "Center": [2.3431932014695382,
48.858844492141145]}, "IncludeCountries": ["FRA"]}' \
--additional-features "Core"

```

Developer tips

Use filters such as `Filter.IncludeCountries` or `Filter.BoundingBox` with `BiasPosition`. These filters can help narrow down possible results and improve accuracy.

```
{
  "QueryText": "Effel tow",
  "BiasPosition": [2.2982750966095398, 48.856078089325294],
  "Filter": {"IncludeCountries": ["FRA"]}
}
```

Manage costs and usage

As you continue learning about Amazon Location places, it's important to understand how to manage service capacity, ensure you follow usage limits, and get the best results through quota and API optimizations. By applying best practices for performance and accuracy, you can tailor your application to handle place-related queries efficiently and maximize your API requests.

Topics

- [Best practices](#)
- [Places pricing](#)
- [Places quota and usage](#)

Best practices

This section outlines best practices for optimizing the performance and accuracy of API interactions in your application. By implementing techniques such as debouncing and browser caching, and leveraging filters and geographical context, you can improve the user experience and ensure that your application delivers the most relevant results.

Typeahead implementation

When working with events or API calls in web development, managing performance is essential to providing a smooth user experience. Two common techniques that can help achieve this are:

Debouncing: Debouncing limits the frequency of function execution, which is especially useful for high-frequency events like window resizing or user input. This ensures that functions are called only after a certain delay, reducing unnecessary processing and enhancing performance.

Browser Caching: By caching recent search queries and results locally, browser caching helps avoid redundant API calls for the same data. This minimizes network traffic and speeds up the application by serving cached data when available.

Both techniques work together to improve performance and efficiency in handling user interactions and API requests.

Getting the right results

Using geographical context, such as bias position or filters like circles and bounding boxes, can enhance the results by focusing on proximity and limiting the output to relevant places. Additionally, filters like countries, place types, categories, chains, and food types can help refine your search further by including or excluding specific criteria.

Places pricing

The price for the Place APIs is based on the number of API requests. The unit price for each API request depends on the response fields you request in your API request and the intended use for the result. For pricing information for the API and pricing buckets, please refer to [Amazon Location Service pricing page](#).

There are four pricing buckets for Place APIs: **Label**, **Core**, **Advanced**, and **Stored**.

Label

The Label pricing bucket provides a cost-effective option to get address text and PlaceID only. When you call the Autocomplete and Suggest APIs with `additionalFeatures = []` or `nonexistent`, the API returns PlaceID (can be used to in a GetPlace request to retrieve additional information), PlaceType, and Title fields for both APIs, and QueryRefinements, QueryType, and QueryId for Suggest API. In this case, you will be charged at the Label price. Results cannot be stored permanently for this pricing bucket. See Stored pricing bucket for long-term use cases.

Core

The Core pricing bucket supports the most common use cases for Place APIs. When you call SearchText, Geocode, ReverseGeocode, SearchNearby, GetPlace API with `additionalFeatures = []` or `nonexistent`, or Autocomplete and Suggest APIs with `additionalFeatures = Core`, the API returns full address components, categories, and other place details (when applicable). In this case, you will be charged at the Core price. Refer to [API reference](#) for a full list of response fields.

Results cannot be stored permanently for this pricing bucket, see the Stored pricing bucket for long-term use cases.

Advanced

The Advanced pricing bucket provides additional place or points-of-interest details, such as business hours, contact information, and access points. When you call `SearchText`, `SearchNearby`, and `GetPlace` API with and include one of the following values in the `additionalFeatures` request field: `Contact`, `Access`, `TimeZone`, or `Phonemes`, the API returns corresponding values for the additional information you have requested (for example, `opening_hours` and `contact_details`, `access_restriction` and `access_points`, `Phonemes`, or `Timezone`). In this case, you will be charged at the Advanced price. Results can be cached but not stored for long-term use for this pricing bucket. Results cannot be stored permanently for this pricing bucket. See Stored pricing bucket for long-term use cases.

Stored

You can store the Places results indefinitely for long-term use cases, such as reusing the results to reduce on-demand API calls or for analytical purpose. To do so, set `intendedUse = Stored` in your API request. In this case, you will be charged at the Stored price. The Stored pricing bucket supports all the features listed above, therefore, the maximum price you will be charged for a single Places API call is capped at the Stored price.

Places quota and usage

Amazon Location Service places quotas on API usage to manage service capacity and prevent overutilization. These quotas can be adjusted through the AWS service quotas console or by contacting support. This section covers the service quotas for Place APIs and API usage limits, including how to request increases and other related information.

Service quota

Amazon Location Service sets default quotas for APIs to help manage service capacity, which can be viewed in the AWS Service Quotas management console. You can request an increase in Amazon Location Service quotas through the self-service console, for up to 2x the default limit for each API. For quota limits exceeding 2x the default limit, request in the [service quota console](#) and it will submit a support ticket. Alternatively, you can connect with your premium support team. There are no direct charges for quota increase requests, but higher usage levels may lead to increased service costs based on the additional resources consumed.

API Name	Default	Adjustable Max limit	More than Adjustable Max limit
Geocode	100	200	Request on service quota console or contact support team
ReverseGeocode	100	200	Request on service quota console or contact support team
Autocomplete	100	200	Request on service quota console or contact support team
GetPlace	100	200	Request on service quota console or contact support team
SearchText	100	200	Request on service quota console or contact support team
SearchNearby	100	200	Request on service quota console or contact support team
Suggest	100	200	Request on service quota console or contact support team

Other usage limits

Feature availability

Autocomplete is not available in Japan.

API Name	Limit	Value
Geocode	Response payload size after compression	6MB
ReverseGeocode	Response payload size after compression	6MB
Autocomplete	Response payload size after compression	6MB
GetPlace	Response payload size after compression	6MB
SearchText	Response payload size after compression	6MB
SearchNearby	Response payload size after compression	6MB
Suggest	Response payload size after compression	6MB

Next steps

For additional information, see:

- [Attribution](#): Ensure proper data attribution when using Amazon Location Service data.
- [SLA](#): Review Amazon Location Service's Service Level Agreement to understand availability guarantees.
- [Service Terms](#): Familiarize yourself with the legal terms governing the use of Amazon Location Service.

Amazon Location Service Routes



With Amazon Location Service Routes, you can calculate travel time and distance between multiple starting points and ending points, visualize vehicle GPS traces aligned with roads, and better understand your serviceable areas. This helps reduce operating costs and improve customer experience.

Features

Plan a route path

Calculate routes between two or more locations, considering various factors such as distance, time, and road conditions. You can also see alternative routes for the same set of locations.

Route Optimization

Optimize routes for time or distance, choosing the fastest or shortest route. You can also sequence waypoints to optimize for the traveling salesman problem.

Route Analysis

Analyze performance metrics like travel time, distance, or the number of stops to ensure the route meets your desired goals.

Service Area

Defines the geographic area that can be serviced from a particular location based on distance or time limits.

Toll Costs

Calculate the costs associated with toll infrastructure on your route.

Avoidances

Constrain your route calculation by avoiding highways, tunnels, ferries, and toll roads.

Speed Limits

Find speed limits for each segment of a route, ensuring drivers comply with local regulations.

The image shows a navigation application interface. At the top, there are four mode icons: a car (selected), a scooter, a pedestrian, and a truck. A close button (X) is in the top right. Below the modes, the start point is 'My Location' and the destination is 'Starbucks'. A vertical double-headed arrow indicates the direction of travel. Below this, there are two dropdown menus: 'Leave now' and 'Avoid'. The selected mode is 'Drive', which is highlighted with a car icon. The distance to the destination is '650 m' and the estimated time is '2 min'. The route is shown as a vertical line with three circular markers. The top marker is teal and labeled 'Laurel St. 300 m'. The middle marker is teal and labeled 'Rose Ave 310 m'. The bottom marker is orange and labeled '2288 Granville St 30 m'.

Use cases

Provide efficient routes and turn-by-turn directions

Plan routes from any starting location (address, POI, or GPS coordinates) and calculate the best path to multiple destinations, factoring in vehicle dimensions and restrictions.

For more information, see [the section called "Calculate routes"](#).

Optimize delivery routes

Use waypoint sequencing to solve the Traveling Salesman Problem and calculate the optimal route to multiple destinations, minimizing time and distance.

For more information, see [the section called "Optimize waypoints"](#).

Make sure deliveries are made from the nearest warehouse

Use Matrix Routing and Waypoint Sequencing to identify the fastest route from your warehouses to each customer, maximizing efficiency and minimizing costs.

For more information, see [the section called "Calculate routes"](#) and [the section called "Calculate route matrix"](#).

Improve taxi and ride-share dispatch

Use Matrix Routing to identify the nearest available vehicle and calculate optimal routes based on real-time traffic. This ensures the closest vehicle reaches the customer efficiently, driving both customer satisfaction and business productivity. Make more trips, save on fuel costs, and deliver a superior service experience.

For more information, see [the section called "Calculate routes"](#), [the section called "Calculate route matrix"](#), and [the section called "Calculate isolines"](#).

Find service areas

Use Isoline routes to determine the geographic reach of your business, based on time or distance constraints. This allows the business to identify potential customers and make dispatch plans. In-home healthcare providers can ensure they have enough staff and resources to reach all patients within 15 minutes. Isoline helps optimize service areas, ensure timely deliveries, and locate new facilities.

For more information, see [the section called "Calculate isolines"](#).

Snap GPS trace to roads

Align GPS traces to roads and visualize vehicle movements, ensuring route adherence and regulatory compliance. Fleet managers can see if vehicles are staying on planned routes and identify deviations. Verify that drivers are following guidelines, spot inefficiencies, and ensure compliance with regulations. Corrects GPS inaccuracies and variations, and presents a realistic view of vehicle activity. Enable better decision-making around route optimization, driver behavior, and fleet management.

For more information, see [the section called “Snap to Roads”](#).

APIs

This table provides an overview of key Amazon Location Service APIs for route planning and location-based data processing. Each API offers unique functionality, such as calculating routes, optimizing waypoints, and snapping GPS traces to roads for accurate tracking.

APIs

API name	Description	Read more
Calculate Routes	Calculate a travel distance, travel time, and turn-by-turn directions between a starting point and multiple destinations factoring vehicle restrictions, and real-time traffic.	the section called “Calculate routes”
Calculate Route Matrix	Calculate route distance and time between a set of departure points and a set of destinations, factoring real time traffic.	the section called “Calculate route matrix”
Calculate Isolines	Identify the geographic area that can be reached within a specified time or distance based on your travel modes.	the section called “Calculate isolines”

API name	Description	Read more
Optimize Waypoints	Find the efficient order to travel to multiple destinations, reducing travel time and distance while accounting for factors like traffic and vehicle constraints.	the section called “Optimize waypoints for a route”
Snap to Roads	Match GPS traces to the nearest road segment, to improve accuracy of vehicle tracking and route visualization.	the section called “Snap to Roads”

Routes concepts

The Routes concepts in Amazon Location Service provide a robust framework for planning and optimizing journeys, whether for simple navigation or complex logistics. Routes encompass various components, such as waypoints, legs, steps, and spans, each contributing to the granularity and flexibility of route calculations. Leveraging the Routes API, you can calculate travel distances, estimate travel times, and optimize multi-stop journeys. Additionally, features like route geometry, traffic awareness, speed limits, toll costs, and flexible polyline encoding enhance the capabilities for visualization, analysis, and operational efficiency. This section covers terminology, best practices, and detailed usage patterns, guiding you through implementing effective route solutions for navigation, delivery, field services, and more.

Topics

- [Routes Terminology](#)
- [Where \(origin, destination, waypoint, and traces\)](#)
- [When \(departure and arrival\)](#)
- [How \(travel mode, avoidance, and exclusion\)](#)
- [Traffic awareness](#)
- [Optimize route and waypoint sequence](#)
- [Driver schedule and notices](#)

Routes Terminology

Route

A route provides details for traveling from a departure position through waypoint positions to a destination. It includes travel distance, travel time, route geometry, speed limits, spans, and other attributes.

Route Matrix

A matrix representing travel distance and travel time from a set of origins to a set of destinations. It is useful as an input for route planning or optimization applications.

Waypoint

Waypoints are intermediate stops along a route from the departure point to the destination. The route follows the stopover order as specified in the request.

Leg

A leg represents the journey between two consecutive positions on a route. If the positions are off-road, they are moved to the nearest road. A route with no waypoints consists of a single leg. Routes with one or more waypoints have multiple legs, with each leg representing travel from one waypoint to the next. Certain legs, like those involving ferries, contain specific information pertinent to that leg type.

Step

A step is a segment within a leg, providing summary details for that part of the journey. Types of steps include:

- **Default Steps** – Basic steps with human-readable instructions, suited for web applications that display a route overview.
- **Turn-by-Turn Steps** – Detailed steps suitable for turn-by-turn applications, providing granular instructions.
- **Before Travel Steps** – Steps to complete before starting the journey. Example: boarding a ferry.
- **After Travel Steps** – Steps to complete after reaching the end of a journey. Example: de-boarding a ferry.

Span

A span represents a continuous stretch of a road that shares a consistent set of attributes. New spans are created along a route whenever one of the requested attributes changes.

Segment

A segment is a navigable portion of a road, typically represented as a linear stretch.

Route Geometry

Route geometry depicts the path of a route for visualization, analysis, or other uses. Each route leg's geometry can be represented as a compressed, encoded polyline or as a simple line string.

Flexible Polyline

A compact, encoded polyline format for representing geometry. Recommended for limiting response size and optimized for client-side decoding.

Simple Line String

A GeoJSON LineString format representing geometry. This format produces a larger response payload and is an ordered array of coordinates that can be used to plot routes on a map.

Where (origin, destination, waypoint, and traces)

Specifies the location for route calculation, including where the route starts, ends, and intermediate stops (or locations to be passed through).

Specifies the where for route calculation

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap to Road
Origin(s)	Starting position of the route.	Yes, with options	Yes, with options	Yes, with options	Yes, with options	No
Waypoint	Intermediate positions to be included	Yes, with options	Yes, with options	Yes, with options	Yes, with options	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap to Road
	along the route.					
Destination(s)	Ending position of the route.	Yes, with options	Yes, with options	Yes, with options	Yes, with options	No
Trace points	GPS trace that includes historical position information emitted by a travel mode. These positions include typical GPS inaccuracies and gaps when the device couldn't emit or record this information.	No	No	No	No	Yes, with options

Configurable options

Provides options to customize routing behavior for waypoints, origins, and destinations.

Waypoints options

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
AvoidActionsForDistance	Avoids actions for the provided distance. Typically used to ensure drivers have enough time to make decisions near waypoints.	Yes	No	No	No	No
AvoidUTurns	Specifies whether U-turns are allowed at the waypoint.	Yes	No	No	No	No
Heading	GPS heading at the waypoint position.	Yes	No	No	Yes	No
Matching	Options to configure matching	Yes, with options	No	No	No	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
	the provided position to the road network.					
SideOfStreet	Specifies the side of the street to match the waypoint position.	Yes, with options	No	No	Yes, with options	No
StopDuration	Duration to stop over at the waypoint position.	Yes	No	No	No	No
Position	Longitude and Latitude for the waypoint.	Yes	No	No	Yes	No
PassThrough	Determines if the waypoint should be treated as a stop or passed through.	Yes	No	No	No	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
Id	Identifier string for the waypoint.	No	No	No	Yes	No
AccessHours	Specifies access hours for visiting the destination.	No	No	No	Yes	No
AppointmentTime	Scheduled appointment time at the waypoint.	No	No	No	Yes	No
ServiceDuration	Service time at the waypoint, such as duration of an appointment.	No	No	No	Yes	No
Before	Defines which waypoints must be visited after this one.	No	No	No	Yes	No

Origin options

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
AvoidActionsForDistance	Avoids actions for the provided distance.	Yes	Yes	Yes	No	No
AvoidUTurns	Specifies if U-turns are permitted at the origin.	Yes	No	No	No	No
Heading	GPS heading at the origin position.	Yes	Yes	Yes	No	No
Matching	Options to match the origin position to the road network.	Yes, with options	Yes, with options	Yes, with options	No	No
SideOfStreet	Specifies the side of the street to match the origin position.	Yes, with options	Yes, with options	Yes, with options	No	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
Id	Identifier string for the origin.	No	No	Yes	No	No

Destination options

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
AvoidActionsForDistance	Avoids actions for the provided distance at the destination.	Yes	Yes	Yes	No	No
AvoidUTurns	Specifies if U-turns are allowed at the destination.	Yes	No	No	No	No
Heading	GPS heading at the destination position.	Yes	Yes	Yes	Yes	No
Matching	Options to match the destination	Yes, with options	Yes, with options	Yes, with options	No	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
	on position to the road network.					
SideOfStreet	Specifies the side of the street to match the destination position.	Yes, with options	Yes, with options	Yes, with options	Yes, with options	No
StopDuration	Duration to stop over at the destination.	Yes	No	No	No	No
Id	Identifier string for the destination.	No	No	Yes	No	No
AccessHours	Specifies access hours for visiting the destination.	No	No	Yes	No	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
AppointmentTime	Scheduled appointment time at the destination.	No	No	Yes	No	No
ServiceDuration	Service time at the destination, such as duration of an appointment.	No	No	Yes	No	No

When (departure and arrival)

Specifies the time for route calculation. The time not only determines the timestamps for departure and arrival but also influences the traffic data to be used.

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
Departure Time	Time of departure from the Origin. If neither arrival nor departure time is provided,	Yes	Yes	Yes	Yes	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
	dynamic traffic information is not used, and only free-flow speeds based on historical traffic are applied.					
Depart Now	Uses the current time as the time of departure from the Origin.	Yes	Yes	Yes	No	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
Arrival Time	Time of arrival at the destination. If neither arrival nor departure time is provided, dynamic traffic information is not used, and only free-flow speeds based on historical traffic are applied.	Yes	No	Yes	No	No

How (travel mode, avoidance, and exclusion)

Use the following options to specify the travel mode and related features to use for route calculation.

Travel mode options

Specifies the mode of transport when calculating a route. This setting influences the estimated speed of travel, road compatibility, and the potential use of ferries where needed.

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
Car	Car travel mode.	Yes, with options	Yes, with options	Yes, with options	Yes	Yes
Scooter	Scooter travel mode.	Yes, with options	Yes	Yes, with options	Yes	Yes
Pedestrian	Walking travel mode.	Yes, with options	Yes	Yes	Yes, with options	Yes
Truck	Truck travel mode.	Yes, with options	Yes, with options	Yes, with options	Yes, with options	Yes, with options

Avoidance, exclusion, and allow options

Determines whether a specific set of features should be included, excluded, or avoided during route calculation.

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
Avoidance	Features that are avoided on a best-case basis. If the router cannot find a valid route, the avoidance will be	Yes, with options	Yes, with options	Yes, with options	Yes, with options	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
	ignored, and a notice will indicate that the avoidance could not be honored.					
Exclusion	Features that are strictly excluded. If the router cannot find a valid route with the exclusion options, no routes are returned.	Yes, with options	Yes, with options	No	No	No
Allow	Features that need to be explicitly enabled.	Yes, with options	Yes, with options	Yes, with options	Yes, with options	No

List of avoidance

Lists features that are avoided on a best-case basis. If the router cannot find a valid route, the avoidance will be ignored, and a notice will indicate that the avoidance could not be honored.

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
Areas	Areas to be avoided with exceptions.	Yes, with options	Yes, with options	Yes, with options	Yes, with options	No
Controlled Access Highways	High-speed roads with limited entry points.	Yes	Yes	Yes	Yes	No
Car Shuttle Trains	Trains transporting vehicles through tunnels.	Yes	Yes	Yes	No	No
Dirt Roads	Unpaved roads with natural surfaces.	Yes	Yes	Yes	Yes	No
Ferries	Boats carrying vehicles across water bodies	Yes	Yes	Yes	No	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
Boat Ferries	Ferries transporting vehicles on rail tracks.	No	No	Yes	Yes	No
Rail Ferries	Ferries transporting vehicles on rail tracks.	No	No	Yes	Yes	No
Seasonal Closure	Roads closed during certain seasons.	Yes	No	Yes	No	No
Tunnels	Underground passages for vehicular traffic.	Yes	Yes	Yes	Yes	No
Toll Road	Roads requiring payment for access.	Yes	Yes	Yes	Yes	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
Toll Transponders	Avoids roads where toll transponders are the only means of payment.	Yes	No	Yes	No	No
U-Turns	Points allowing vehicles to turn in reverse direction.	Yes	Yes	Yes	Yes	No
Zone Categories	Zone categories to be avoided.	Yes	Yes	Yes	No	No

List of exclusions

Lists features that are strictly excluded. If the router cannot find a valid route with the exclusion options, no routes are returned.

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
Countries	Country Code 2 or Country Code 3 for countries	Yes	Yes	No	Yes	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
	that should be strictly excluded from route calculation.					

List of allow

Lists features that need to be explicitly enabled for route calculation.

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
HOV	Enable use of high occupancy vehicle lanes for route calculation.	Yes	Yes	Yes	No	No
HOT	Enable use of high occupancy toll lanes for route calculation.	Yes	Yes	Yes	No	No

Traffic awareness

Determines the type of traffic-related information used during route calculation. Flow traffic represents congestion, excluding long-term incident-related congestion. The accuracy of flow traffic data decreases over time, making historical traffic data more reliable for past events.

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
Usage	Enable or disable traffic data during route calculation. When enabled, if <code>DepartureTime</code> , <code>ArrivalTime</code> , or <code>DepartNow</code> are not provided, only long-term closures will be considered. Otherwise, if a time is provided, all traffic data is	Yes	Yes	Yes	Yes	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To Road
	taken into account.					
FlowEvent Threshold Override	Duration in seconds for which a flow traffic event is considered valid. While valid, flow traffic data will be used over historical traffic data.	Yes	Yes	Yes	No	No

Optimize route and waypoint sequence

Optimize routing

Optimization criteria for when calculating a route. This can either be the fastest route measured by time or the shortest route measured by distance.

Option	Description	Measurement
Fastest Route	Calculate the fastest route, focusing on minimizing travel time. This takes into account traffic conditions, road speed limits, and other factors.	Time

Option	Description	Measurement
Shortest Route	Calculate the shortest route, minimizing the distance traveled. This is often used when distance is the key factor, such as reducing fuel costs or emissions.	Distance

Optimize waypoint

Optimization criteria for sequencing waypoints in a route.

Driver schedule and notices

Driver schedule

Driver settings to define work and rest schedules. This is mandatory for many jurisdictions.

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To road
Custom rest cycles	Rest cycles defined by drive duration to be followed up with a rest duration. Any number of such cycles can be provided.	Yes, with options	No	No	No	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To road
Long rest cycle and short rest cycle	Rest cycles defined by short cycle and long cycle. A short drive duration is followed by a short rest drive duration. Short drives can be repeated until a long drive duration is reached, at which point the long rest duration is enforced.	No	No	No	Yes, with options	No

Notices, warnings, and constraints

Supplemental information that provides insight into decisions made during the route calculation.

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To road
Notices	Notices regarding route calculation. Additionally may include an impact for the notice indicating if the results of the route calculation can be used as is or will need to be manually inspected before usage.	Yes, with details	No	No	No	Yes
FailedConstraints	Constraints that were provided in the request that could not be satisfied	No	No	No	Yes, with details	No

Parameter	Description	Routes	Routes Matrix	Isoline	Optimize Waypoint	Snap To road
	, leading to failure of the optimization problem.					

Route APIs

Routes provide capabilities for calculating optimized paths between locations. These features support applications requiring logistics planning, distance calculations, and route optimization. Users can also snap location points to roads for improved accuracy. For more information, See [Routes](#).

- **CalculateIsolines:** Generates isolines based on travel time or distance, useful for defining service areas or reachability zones. For more information, See [the section called “Calculate isolines”](#).
- **CalculateRouteMatrix:** Provides a matrix of distances and travel times between multiple origins and destinations, supporting logistics and trip planning. For more information, See [the section called “Calculate route matrix”](#).
- **CalculateRoutes:** Calculates optimized routes for point-to-point or multi-stop navigation, including customizable routing preferences. For more information, See [the section called “Calculate routes”](#).
- **OptimizeWaypoints:** Optimizes the order of waypoints for the most efficient travel route, minimizing distance or time. For more information, See [the section called “Optimize waypoints”](#).
- **SnapToRoads:** Aligns coordinates to the nearest road paths, enhancing GPS accuracy by snapping points to known roads. For more information, See [the section called “Snap to Roads”](#).

The following table presents a number of business use cases that are best solved with Routes APIs.

Business need	Useful API	Examples
<p>Calculate the travel distance and time for a origins and a destination</p> <p>Supports the following:</p> <ul style="list-style-type: none"> • travel modes, such as truck, pedestrian, car, and scooter • up to 25 way points (stop overs), including origin and destination • driving resting time 	CalculateRoutes	
<p>Find route polyline for a origins and a destination</p> <p>Supports the following:</p> <ul style="list-style-type: none"> • travel modes, such as truck, pedestrian, car, and scooter • up to 25 way points (stop overs), including origin and destination • driving resting time 	CalculateRoutes	
<p>Find routes turn by turn direction for an origin and a destination</p> <p>Supports the following:</p> <ul style="list-style-type: none"> • travel modes, such as truck, pedestrian, car, and scooter • up to 25 way points (stop overs), including origin and destination • driving resting time 	CalculateRoutes	

Business need	Useful API	Examples
<p>Find a route by departing now</p> <p>Supports the following:</p> <ul style="list-style-type: none">• travel modes, such as truck, pedestrian, car, and scooter• up to 25 way points (stop overs), including origin and destination• driving resting time	CalculateRoutes	
<p>Find a route if you depart at specific time</p> <p>Supports the following:</p> <ul style="list-style-type: none">• travel modes, such as truck, pedestrian, car, and scooter• up to 25 way points (stop overs), including origin and destination• driving resting time	CalculateRoutes	
<p>Find a route if you need to arrival at specific time</p> <p>Supports the following:</p> <ul style="list-style-type: none">• travel modes, such as truck, pedestrian, car, and scooter• up to 25 way points (stop overs), including origin and destination• driving resting time	CalculateRoutes	

Business need	Useful API	Examples
Find shortest routes Supports the following: <ul style="list-style-type: none">• travel modes, such as truck, pedestrian, car, and scooter• up to 25 way points (stop overs), including origin and destination• driving resting time	CalculateRoutes	
Find fastest routes Supports the following: <ul style="list-style-type: none">• travel modes, such as truck, pedestrian, car, and scooter• up to 25 way points (stop overs), including origin and destination• driving resting time	CalculateRoutes	
Find alternative routes Supports the following: <ul style="list-style-type: none">• travel modes, such as truck, pedestrian, car, and scooter• up to 25 way points (stop overs), including origin and destination• driving resting time	CalculateRoutes	

Business need	Useful API	Examples
<p>Find traffic aware routes Supports the following:</p> <ul style="list-style-type: none">• travel modes, such as truck, pedestrian, car, and scooter• up to 25 way points (stop overs), including origin and destination• driving resting time	CalculateRoutes	
<p>Find routes with avoidance such as toll, u-turn, ferry, highway, tunnel, and more Supports the following:</p> <ul style="list-style-type: none">• travel modes, such as truck, pedestrian, car, and scooter• up to 25 way points (stop overs), including origin and destination• driving resting time	CalculateRoutes	
<p>Find routes with custom avoidance by passing polyline or polygon Supports the following:</p> <ul style="list-style-type: none">• travel modes, such as truck, pedestrian, car, and scooter• up to 25 way points (stop overs), including origin and destination• driving resting time	CalculateRoutes	

Business need	Useful API	Examples
<p>Calculate toll cost Supports the following:</p> <ul style="list-style-type: none"> • travel modes, such as truck, pedestrian, car, and scooter • up to 25 way points (stop overs), including origin and destination • driving resting time 	CalculateRoutes	the section called “Calculate toll cost”
<p>Find speed limit of a road span on a route Supports the following:</p> <ul style="list-style-type: none"> • travel modes, such as truck, pedestrian, car, and scooter • up to 25 way points (stop overs), including origin and destination • driving resting time 	CalculateRoutes	
<p>Draw a route on a map Supports way point marking.</p>	GetTile and GetStyled escriptor with rendering engine (MapLibre) with Calculate route	
<p>Calculate route matrix of distance and time for multiple origins and destinations Supports the following:</p> <ul style="list-style-type: none"> • more than 100K cells • travel modes, such as truck, pedestrian, car, and scooter 	CalculateRouteMatrix	the section called “Calculate the route matrix of distance and time for multiple origins and destinations”

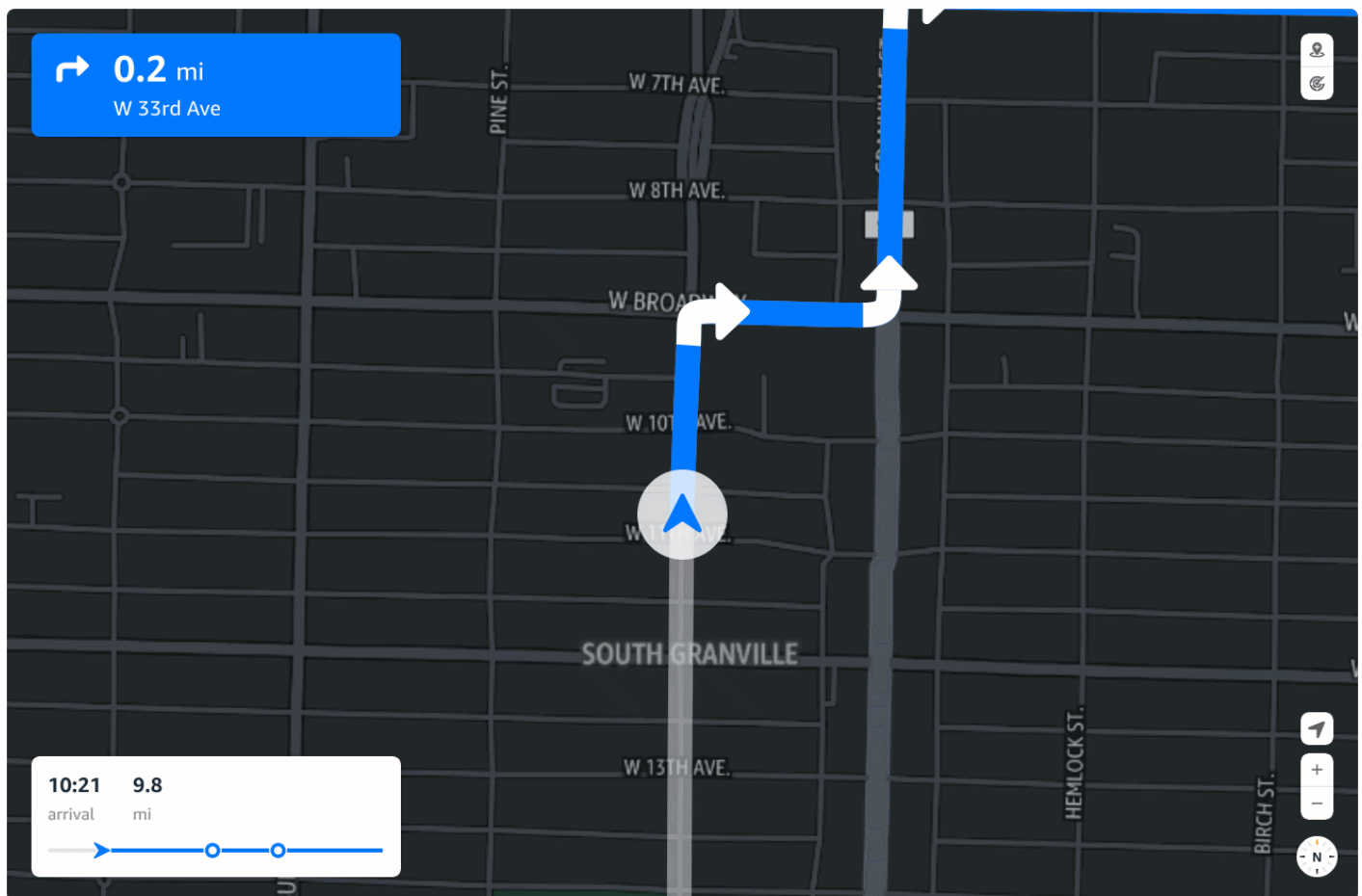
Business need	Useful API	Examples
Calculate route matrix with avoidance Supports the following: <ul style="list-style-type: none">• more than 100K cells• travel modes, such as truck, pedestrian, car, and scooter	CalculateRouteMatrix	the section called “Calculate route matrix with avoidance”
Calculate a service area based on time (isochrones) Supports travel modes, such as truck, pedestrian, car, and scooter.	CalculateIsoline	the section called “Calculate a service area based on time”
Calculate a service area based on distance (isodistance) Supports travel modes, such as truck, pedestrian, car, and scooter.	CalculateIsoline	
Calculate a service area with avoidance Supports travel modes, such as truck, pedestrian, car, and scooter.	CalculateIsoline	
Calculate a service area from reverse direction Supports travel modes, such as truck, pedestrian, car, and scooter.	CalculateIsoline	

Business need	Useful API	Examples
<p>Calculate service area for multi-range of time or distance</p> <p>Supports travel modes, such as truck, pedestrian, car, and scooter.</p>	CalculateIsoline	
<p>Optimize way point for a routes (traveling salesman problem)</p> <p>Supports travel modes, such as truck, pedestrian, car, and scooter.</p>	OptimizeWaypoint	the section called “Optimize waypoints for a route”
<p>Optimize way point for a route with traffic awareness</p> <p>Supports travel modes, such as truck, pedestrian, car, and scooter.</p>	OptimizeWaypoint	
<p>Optimize way point for a route with access hour awareness</p> <p>Supports travel modes, such as truck, pedestrian, car, and scooter.</p>	OptimizeWaypoint	
<p>Match GPS traces to road network</p> <p>Supports travel modes, such as truck, pedestrian, car, and scooter.</p>	Snap to road	the section called “Match GPS traces to road network”

Business need	Useful API	Examples
Visualize matched GPS traces on a map Supports travel modes, such as truck, pedestrian, car, and scooter.	GetStyleDescriptor with rendering engine (MapLibre) with Snap to road	

Calculate routes

The Routes API calculates routes between two or more locations with or without avoidances for different travel modes such as car, truck, scooter, and pedestrian. With this API, you can customize routing options and request additional route-related information to meet specific needs. This API supports turn-by-turn navigation and customizes route calculations by applying parameters like avoiding toll roads, motorways, or ferries. The API also returns speed limits and toll costs.



Use cases

- **Display geographic details on a route map:** Use advanced mapping features to visualize detailed routes with rich geographic information, including landmarks, terrain, and urban infrastructure. Enhance decision-making by allowing users to view clear routes from their starting point to their destination. This feature can support navigation, planning, and various logistics scenarios, and display routes for travel modes such as cars, trucks, scooters, and pedestrians. Customize routes by adding elements like avoidances or toll calculations.
- **Show turn-by-turn navigation:** Provide seamless navigation support on web and mobile devices. Users can access turn-by-turn directions, ensuring efficient travel. Both platforms can leverage navigation instructions to offer routes for personal or business travel, including speed limits.
- **Calculate toll costs along routes:** Incorporate toll cost calculations into route planning to provide accurate pricing estimates for routes that include toll roads, bridges, or tunnels. Display toll costs upfront to help drivers and planners make cost-effective decisions and avoid tolls when necessary.
- **Ensure compliance with speed limits:** Integrate speed limit data to help drivers stay within legal limits, reducing the risk of fines and promoting safer, fuel-efficient driving. Logistics and fleet management can also benefit by monitoring speed compliance in real time.
- **Assist with freight and vehicle routing solutions:** Simplify freight and vehicle routing operations by integrating routes, navigation, and tracking capabilities into logistics portals. Efficiently plan routes for multiple deliveries, track shipments in real-time, and manage fuel costs through better routing.

Understand the request

The request requires `Origin` and `Destination` parameters, while optional parameters like `Allow`, `Avoid`, and `Traffic` customize the route to meet specific needs and constraints. .

Origin

The start position of the route in longitude and latitude.

Destination

The end position of the route.

Waypoints

Intermediate positions to include along a route between the start and end positions.

OptimizeRoutingFor

Optimization criteria for the route, such as fastest or shortest.

LegGeometryFormat

Format of the geometry returned for each route leg.

Avoid

Features to be avoided during route calculation, ignored if no alternative route is found.

Traffic

Traffic-related options affecting route calculation.

Tolls

Toll-related options impacting route calculations and toll costs.

LegAdditionalFeatures

Features that can be enabled within the response for each leg of the journey.

SpanAdditionalFeatures

Span features that can be enabled within the response for each leg of the journey.

Understand the response

The response provides route details such as the legs of the journey, notices about route calculations, and summary information including distance and duration. .

Routes

Array of routes containing legs and associated properties.

Notices

Warnings or informational messages about the route.

LegGeometryFormat

Specifies the format of the route's geometry.

Leg details

Each leg of a journey can be of type `Ferry`, `Pedestrian`, or `Vehicle` depending on the transport mode. While each leg contains properties agnostic to transport mode, specific properties can be found under:

FerryLegDetails

Ferry-specific properties for the leg.

VehicleLegDetails

Vehicle-specific properties for the leg.

PedestrianLegDetails

Pedestrian-specific properties for the leg.

Steps

Each leg of a journey is divided into steps that describe actions for portions of the route. A step can be either `Default`, suitable for basic applications, or `TurnByTurn`, suitable for turn-by-turn navigation. Each step contains properties agnostic to step type, such as duration and distance, and other specific properties like `ExitStepDetails`, which applies only to exit steps.

BeforeTravelSteps

Steps to perform before beginning the journey.

TravelSteps

Steps to perform during the journey.

AfterTravelSteps

Steps to perform after completing the journey.

Spans

Each leg of a journey can be split into spans. A span is a portion of the leg with the same values for the set of requested `SpanAdditionalFeatures`. Spans are divided by road properties such as `SpeedLimit`, road names, or regions. Returned spans can be used to visualize road attributes and access-related information.

Calculate toll cost

This topic provides an overview of the fields and definitions associated with calculating toll costs. Using these fields, you can specify parameters such as payment methods, currency, and vehicle characteristics to customize toll cost calculations.

Field name	Routes
Transponders	Yes, with options
Vignettes	Yes, with options
Currency	Yes, with options
EmissionType	Yes, with options
VehicleCategory	Yes, with options

Definitions

This section provides brief definitions for each field used in toll cost calculation.

Transponders

Transponders are a method of payment for tolls, potentially resulting in a different price compared to other payment methods.

Vignettes

A vignette is a form of road pricing. When a user has the required vignette, no additional toll payments are necessary.

Currency

The currency in which toll costs are reported. In addition to the local currency, a converted currency is included, which also impacts the currency used in the toll summary within the response.

EmissionType

The emission type of the vehicle, used for calculating toll costs based on vehicle emissions.

VehicleCategory

The vehicle sub-category used for toll cost calculation.

Understanding route steps

This section defines various actions and steps that need to be taken to complete a leg of a journey. Route steps vary by travel mode and provide guidance for both overview applications and detailed turn-by-turn navigation.

Route steps overview

The following types of route steps define the actions needed to complete a route leg, varying by travel mode and the stage of the journey.

Step type	Description
Default steps	Basic steps providing human-readable instructions, often used in web-based applications to offer an overview of the route.
Turn by turn steps	Detailed steps for creating a turn-by-turn navigation application, offering more granular directions.
Before travel steps	Steps that need to be completed before starting the travel section, such as boarding a ferry.
After travel steps	Steps to be performed after the travel section is complete, like de-boarding a ferry.

Step breakdown by travel mode

Section	Step	Before Travel	Travel	After Travel
Vehicle	Arrive	No	Yes	No
Vehicle	Continue	No	Yes	No
Vehicle	ContinueH ighway	No	Yes	No

Section	Step	Before Travel	Travel	After Travel
Vehicle	Depart	No	Yes	No
Vehicle	Exit	No	Yes	No
Pedestrian	Arrive	No	Yes	No
Pedestrian	Charge	No	Yes	No
Ferry	Wait	No	No	Yes
Ferry	Board	Yes	No	No
Ferry	Deboard	No	No	Yes

Calculate route matrix

The Matrix Routing service calculates routing matrices, providing travel times or distances between multiple origins and destinations. This service offers flexible customization options, allowing you to specify travel modes, traffic conditions, and other routing parameters. The matrix calculations can vary in size and shape, supporting both square and non-square matrices, and accommodate dynamic or free-flow traffic data.

Use cases

- *Optimize delivery routes for logistics and e-commerce:* Efficiently calculate travel time and distance between multiple pickup and delivery locations to optimize routes. Logistics companies can use this feature to minimize costs and delivery time by planning efficient paths across cities. It is ideal for setting optimized delivery windows for same-day or next-day services and planning multi-stop delivery routes.
- *Match drivers and passengers in ride-sharing applications:* Use route calculations to match drivers with the closest passengers by finding the fastest route between locations. Ride-sharing apps can enhance user experience by providing real-time driver arrival estimates, ensuring prompt pickups and drop-offs. Supports various transportation modes like cars, bikes, and scooters.
- *Plan and optimize routes for fleet management:* Manage large fleets by optimizing routes to reduce fuel consumption and travel time. Fleet managers can assign the most efficient routes to vehicles for multiple stops, thereby increasing overall operational efficiency. Use cases include

service fleets, transportation companies, and utilities where optimal route planning is essential for site visits.

Understand the request

The request includes **Origins** and **Destinations** for route calculations, with optional parameters to tailor the matrix based on preferences and constraints. For more details, refer to the API Reference for Calculate Route Matrix API.

- **Origins**: List of origin coordinates in longitude and latitude.
- **Destinations**: List of destination coordinates.
- **OptimizeRoutingFor**: Optimization criteria such as "Fastest" or "Shortest" route.
- **RoutingBoundary**: Defines boundaries for calculation, either as "Unbounded" or restricted to a specific geometry.
- **Avoid**: Features to avoid during route calculation. Ignored if no viable route can be found.
- **Traffic**: Traffic-related options impacting route calculations.

Understand the response

The response includes a matrix of calculated routes between origins and destinations, with details such as distance and duration. Errors and boundaries for the routes are also provided, if applicable. Refer to the API Reference for additional details on the Calculate Route Matrix API.

- **RouteMatrix**: Matrix containing travel distances and durations between origins and destinations.
- **ErrorCount**: Number of errors encountered during route calculations.
- **RoutingBoundary**: Boundary within which the matrix is calculated.

Calculate isolines

The Calculate Isolines API allows you to determine areas reachable within specified time or distance limits. By considering factors such as road restrictions, traffic conditions, and travel mode, it generates isolines that outline accessible areas, supporting urban planning, logistics, and service accessibility applications. This API can be used for urban planning, real estate analysis, and accessibility studies by visualizing service reach, transportation options, or resources within a set

timeframe or distance limit. By displaying these isolines on a map, users can assess travel reach within specific constraints, enhancing decision-making for site selection, service coverage, and resource allocation.

Use cases

- **Assess healthcare accessibility through travel time isolines:** Generate isolines to evaluate access to healthcare facilities from various neighborhoods based on travel times. Healthcare organizations can use this feature to identify underserved areas and make informed decisions about clinic locations or mobile health services, thereby improving community healthcare access.
- **Analyze market reach for retail expansion using travel time isolines:** Create isolines to represent customer access to retail locations based on travel times. Retail businesses can assess new store locations and understand customer demographics, using these visualizations to strategically expand and optimize sales potential.
- **Optimize logistics and delivery zones with isolines:** Generate isolines to define delivery zones based on time-sensitive logistics requirements. Logistics companies can visualize areas reachable within specific timeframes from distribution centers, improving route planning, operational efficiency, and timely deliveries.
- **Plan tourism and recreational access with isolines:** Visualize travel times from tourist attractions to nearby accommodations. Tourism boards can help travelers find convenient lodging options, promoting local businesses and enhancing the travel experience by displaying these isolines on a map.
- **Improve emergency response planning through isolines:** Generate isolines to assess response times from emergency service locations to various areas in a community. Emergency management teams can identify regions within critical response times, optimizing resource allocation to improve response during incidents.
- **Analyze commuting patterns for workforce planning with isolines:** Generate isolines to visualize commuting times and identify areas with high travel times. Businesses can use these insights for remote work policies or office relocations, enhancing employee satisfaction and productivity.

Understand the request

The request accepts parameters like `Origin`, `Destination`, and `Thresholds` for defining isolines. Optional parameters, such as `Allow`, `Avoid`, and `TravelModeOptions`, allow customization of isoline constraints. For more information, see the .

Origin

The starting point for isoline calculation in longitude and latitude.

Thresholds

Time or distance limits used to define the isoline boundary.

TravelMode

The mode of transport, such as car, pedestrian, or truck.

OptimizeIsolineFor

Optimization criteria, such as Accurate, Balanced, or Fast calculation.

DepartureTime

Time of departure, if specified, for calculating time-dependent isolines.

ArrivalTime

Time of arrival, if specified, for calculating time-dependent isolines.

IsolineGranularity

The maximum number of points and resolution of the isoline boundary.

Understand the response

The response includes isolines with properties like `IsolineGeometryFormat`, outlining the reachable area based on request parameters. .

Isolines

Calculated isolines with associated properties, including geometries and connections.

Geometries

List of geometries outlining the calculated isoline boundaries.

Connections

Connections between isoline geometries, including the geometry for each connection.

Optimize waypoints

The Optimize Waypoints API calculates the most efficient sequence for visiting multiple waypoints along a route. Using advanced algorithms, this API minimizes travel time and distance while considering factors such as traffic conditions, avoidances, and vehicle specifications. Integrating the Optimize Waypoints API helps businesses streamline operations, reduce fuel consumption, enhance delivery efficiency, and improve customer satisfaction. The API provides optimized routes, enabling better decision-making and resource allocation in multi-stop travel scenarios.

Use cases

- **Enhance multi-stop delivery efficiency:** Efficiently optimize the sequence of multiple delivery stops to reduce travel time and costs. Delivery services can streamline operations by calculating the most efficient route for drivers, minimizing fuel expenses and ensuring timely deliveries, which improves customer satisfaction and operational efficiency.
- **Streamline field service operations:** Optimize the sequence of visits to multiple job sites in a single day, reducing travel time for field service technicians. This allows companies to complete more jobs daily, enhancing productivity and service delivery.
- **Plan efficient tour routes for travel agencies:** Optimize itineraries that include multiple attractions to maximize sightseeing while minimizing travel time. Travel agencies can use this feature to create optimal plans for guided tours, enhancing the overall tourist experience by making better use of available time.
- **Improve ride-sharing driver efficiency:** Optimize pick-up and drop-off sequences for multiple passengers, reducing wait times and enhancing the rider experience. Ride-sharing services can maximize driver earnings and ensure timely service for passengers by optimizing waypoints.
- **Optimize routes for waste collection services:** Plan garbage collection routes to minimize travel distance and time, which helps waste management companies streamline operations and ensure timely collection, achieving cost savings and reducing environmental impact.
- **Coordinate logistics for events and conferences:** Manage transportation logistics for delivering equipment and supplies to multiple venues, optimizing loading and unloading routes. This allows event planners to streamline transportation, reduce delays, and ensure timely material arrival for events.
- **Enhance emergency response routes:** Plan the fastest routes to multiple emergencies, optimizing response times in critical situations. Emergency services can improve response efficiency, potentially saving lives by using optimized waypoints.

- **Facilitate sales route planning for field representatives:** Optimize routes for sales representatives visiting multiple clients in a day, minimizing travel time and maximizing client visits. This helps companies increase productivity and capitalize on more sales opportunities.

Understand the request

The request requires parameters like `Origin` and `Waypoints` to calculate an optimized sequence. Optional parameters like `Avoid`, `Traffic`, and `Driver` allow additional customization.

Waypoints

A list of waypoints to be optimized in sequence.

Origin

The starting position of the route for optimization.

Destination

An optional end position of the route for optimization.

OptimizeSequencingFor

Criteria for sequencing optimization, such as fastest or shortest route.

Traffic

Traffic-related options affecting route calculation.

Driver

Driver work and rest cycles to ensure compliance with regional driving regulations.

Each waypoint can also specify constraints that must be satisfied, such as `AppointmentTime`, `AccessHours`, and ordering constraints like `Before` another waypoint.

Understand the response

The response provides details of the optimized waypoint sequence, including `OptimizedWaypoints` and the overall `Distance` and `Duration` for the journey.

OptimizedWaypoints

A list of waypoints in their optimized order.

ImpedingWaypoints

Waypoints that prevent an optimized sequence, including failed constraints that were not met.

Connections

Details about travel between waypoints, including distance and duration.

TimeBreakdown

Breakdown of total `Travel`, `Rest`, `Service`, and `Wait` durations for the route.

Snap to Roads

The Snap to Road API enhances the accuracy of geographic positioning by aligning GPS coordinates to the nearest road segments on a digital map. This API takes raw longitude and latitude data, often collected from mobile devices or vehicles, and "snaps" these points to the corresponding road network, correcting inaccuracies caused by GPS drift or signal loss. By integrating the Snap to Road API, developers can ensure their applications provide reliable and accurate data, supporting better decision-making and operational efficiency across various scenarios.

Use cases

- **Enhance navigation accuracy by snapping to the road network:** Efficiently align GPS coordinates to the nearest road segments to improve navigation accuracy. This feature is valuable for mapping and navigation services, providing users with precise directions and real-time location updates, thus enhancing the navigation experience.
- **Improve data accuracy for fleet management applications:** Correct the reported positions of vehicles by snapping their GPS coordinates to the closest roadways. In fleet management systems, this feature ensures accurate vehicle tracking, enabling optimized logistics and better operational efficiency.

Understand the request

The request requires `TracePoints` to match to roads, with optional parameters like `SnappedGeometryFormat` and `SnapRadius` for controlling geometry format and snapping radius. .

TracePoints

A list of coordinates to be snapped to the road network.

SnappedGeometryFormat

The format of the returned geometry, such as "FlexiblePolyline" or "Simple".

SnapRadius

The radius around trace points within which road snapping is considered.

Understand the response

The response contains corrected geometry and snapped trace points, with properties like `SnappedGeometry` and `SnappedTracePoints` to indicate accuracy and snapping confidence. .

SnappedGeometry

The corrected geometry of the snapped route.

SnappedTracePoints

The adjusted coordinates of the trace points snapped to roads.

Notices

Warnings or informational messages about the snapping process.

How to

This section provides guides for leveraging the Routes APIs to solve routing and navigation challenges. These tutorials cover the essential tasks for integrating routing functionality into your applications. Each topic focuses on a specific use case, enabling you to efficiently implement advanced routing features tailored to your business needs.

Topics

- [How to use CalculateRoutes](#)
- [How to calculate isolines](#)
- [How to calculate a route matrix](#)
- [How to optimize waypoints](#)
- [How to use SnapToRoads](#)

How to use CalculateRoutes

This section provides step-by-step instructions for using CalculateRoutes. This topic details guidance on finding routes with specific configurations, such as incorporating turn-by-turn navigation, setting travel modes, and adding waypoints.

Topics

- [How to find a route for an origin and destination](#)
- [How to find routes with turn-by-turn directions for an origin and destination](#)

How to find a route for an origin and destination

The CalculateRoutes API allows you to find routes between an origin and a destination. It supports various travel modes, including truck, pedestrian, car, and scooter. The API can accommodate up to 25 waypoints (stopovers) between the origin and destination, and offers options to apply constraints and driving rest times.

Potential use cases

- **Find point-to-point routes:** Determine the best route between two locations based on various travel modes and additional options.

Examples

Calculate routes from origin to destination using car mode

Sample request

```
{
  "Origin": [
    -123.118105,
    49.282423
  ],
  "Destination": [
    -123.020098,
    49.232872
  ]
}
```

Sample response

```
{
  "LegGeometryFormat": "FlexiblePolyline",
  "Notices": [],
  "Routes": [
    {
      "Legs": [
        {
          "Geometry": {
            "Polyline": "BGm4-_9Cvgx6qH9jBw8..."
          },
          "TravelMode": "Car",
          "Type": "Vehicle",
          "VehicleLegDetails": {
            "AfterTravelSteps": [],
            "Arrival": {
              "Place": {
                "ChargingStation": false,
                "OriginalPosition": [
                  -123.020098,
                  49.232872
                ],
                "Position": [
                  -123.0203051,
                  49.2328499
                ]
              }
            },
            "Departure": {
              "Place": {
                "ChargingStation": false,
                "OriginalPosition": [
                  -123.1181051,
                  49.282423
                ],
                "Position": [
                  -123.1180883,
                  49.2824349
                ]
              }
            }
          },
          "TravelSteps": [
            {
```

```
        "Distance": 1288,  
        "Duration": 102,  
        "Type": "Depart"  
    },  
    {  
        "Distance": 262,  
        "Duration": 24,  
        "Type": "Ramp"  
    },  
    {  
        "Distance": 1356,  
        "Duration": 134,  
        "Type": "Turn"  
    },  
    {  
        "Distance": 7092,  
        "Duration": 568,  
        "Type": "Keep"  
    },  
    {  
        "Distance": 65,  
        "Duration": 26,  
        "Type": "Turn"  
    },  
    {  
        "Distance": 50,  
        "Duration": 18,  
        "Type": "Turn"  
    },  
    {  
        "Distance": 0,  
        "Duration": 0,  
        "Type": "Arrive"  
    }  
  ]  
}  
}
```


cURL

```
curl --request POST \  
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/routes?key=Your_key' \  
  --header 'Content-Type: application/json' \  
  --data '{  
    "Origin": [  
      -123.118105,  
      49.282423  
    ],  
    "Destination": [  
      -123.020098,  
      49.232872  
    ]  
  }'
```

AWS CLI

```
aws geo-routes calculate-routes --key ${YourKey} \  
  --origin -123.118105 49.282423 \  
  --destination -123.020098 49.232872
```

How to find routes with turn-by-turn directions for an origin and destination

The CalculateRoutes API enables you to find routes between an origin and destination with turn-by-turn navigation. It supports various travel modes, including truck, pedestrian, car, and scooter. The API also supports up to 25 waypoints (stopovers) and allows for applying constraints, including driving rest times.

Potential use cases

- **Create a navigation mobile app:** Use the API to provide turn-by-turn navigation instructions for app users.
- **Display directions on a web platform:** Show detailed route guidance for web applications to assist users with navigation.

Examples

Calculate routes using car mode

Sample request

```
{
  "Origin": [
    -123.118105,
    49.282423
  ],
  "Destination": [
    -123.020098,
    49.232872
  ],
  "TravelStepType": "TurnByTurn",
  "TravelMode": "Car"
}
```

Sample response

```
{
  "LegGeometryFormat": "FlexiblePolyline",
  "Notices": [],
  "Routes": [
    {
      "Legs": [
        {
          "Geometry": {
            "Polyline": "BGm4-_9Cvgx6qH9jBw8BjS4czUwgBrT..."
          },
          "TravelMode": "Car",
          "Type": "Vehicle",
          "VehicleLegDetails": {
            "Arrival": {
              "Place": {
                "Position": [-123.0203051, 49.2328499]
              }
            },
            "Departure": {
              "Place": {
                "Position": [-123.1180883, 49.2824349]
              }
            }
          }
        }
      ]
    }
  ]
}
```

```
    },
    "TravelSteps": [
      {
        "Distance": 1288,
        "Duration": 102,
        "Type": "Depart",
        "NextRoad": {
          "RoadName": "W Georgia St",
          "RouteNumber": "HWY-1A"
        }
      },
      {
        "Distance": 262,
        "Duration": 24,
        "Type": "Keep",
        "NextRoad": {
          "RoadName": "Main St",
          "RouteNumber": "HWY-1A"
        }
      },
      {
        "Distance": 1356,
        "Duration": 134,
        "Type": "Turn",
        "NextRoad": {
          "RoadName": "Main St",
          "RouteNumber": "HWY-1A"
        }
      },
      {
        "Distance": 7092,
        "Duration": 568,
        "Type": "Keep",
        "NextRoad": {
          "RoadName": "Kingsway",
          "RouteNumber": "HWY-1A"
        }
      },
      {
        "Distance": 65,
        "Duration": 26,
        "Type": "Turn"
      }
    ]
  }
```

```

    "Distance": 0,
    "Duration": 0,
    "Type": "Arrive"
  }
]
}

```

cURL

```

curl --request POST \
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/routes?key=Your_key' \
  --header 'Content-Type: application/json' \
  --data '{
    "Origin": [
      -123.118105,
      49.282423
    ],
    "Destination": [
      -123.020098,
      49.232872
    ],
    "TravelStepType": "TurnByTurn",
    "TravelMode": "Car"
  }'

```

AWS CLI

```

aws geo-routes calculate-routes --key ${YourKey} \
  --origin -123.118105 49.282423 \
  --destination -123.020098 49.232872 \
  --travel-step-type "TurnByTurn" \
  --travel-mode "Car"

```

How to calculate isolines

Learn how to calculate isolines using time, distance or setting areas to avoid.

Topics

- [How to calculate a service area based on time](#)
- [How to calculate a service area based on distance](#)

How to calculate a service area based on time

The `CalculateIsolines` API allows you to determine reachable service areas within a specified time or distance, factoring in road networks and traffic conditions. This capability supports applications such as defining service areas for restaurants, grocery stores, or other service providers.

Potential use cases

- **Plan service areas:** Use this API to plan accessible areas for services like restaurants or grocery delivery based on travel time or distance.

Examples

Calculate a service area based on time with car `TravelMode`

Sample request

```
{
  "Origin": [
    -123.11679620827039,
    49.28147612192166
  ],
  "DepartureTime": "2024-10-28T21:27:56Z",
  "Thresholds": {
    "Time": [
      300
    ]
  },
  "TravelMode": "Car"
}
```

Sample response

```
{
  "DepartureTime": "2024-10-28T14:27:56-07:00",
  "IsolineGeometryFormat": "FlexiblePolyline",
}
```

```

    "Isolines": [
      {
        "Connections": [],
        "Geometries": [
          {
            "PolylinePolygon": [
              "BGg0n_9Cpx37qHsFuNmgBmgB2KmgBmgBmgB4KmgBmgBmgB4KmgBkgBmgB4KmgB-
qB8qB2KA4K1KkQrF8qBAkQsF2K2K4KA4K1KiQrFsgCAkQsFuVuV4KmgBmgBmgB2KmgB4KAsFhQA9qBsFjQuVtVmgB3K4
qBAiQsF4KmgBiQsFkQrF4K3KiQrF-qBAiQsF4KmgB2KAmgBlgBkQrF-qBAiQsF4K4KmgB2KuVwVsFiQA-
qBsFiQiQsFkQrF2K1KkQrFuVAkQsFuVuVsFkQA8qBsFkQiQsFuVAkQqFsFkQAsgCrFiQ3K4K1KmgB3K4KA2K4K4KqFiQ
qBrFiQ3K4KpFkQqFiQ21B01BqFkQA4gEpFiQ3K4KrFiQAsgCrFkQ1K4KrFiQA-
qBrFiQ3K4KpFiQAwVrFiQ3K4KrFiQAwVrFiQhQsFrgCAjQsFtVuV3KmgBz1B01B3KmgBlgBmgB1KmgB3K4KrFkQsFiQm
qBsFiQmgB4K4KmgB2K4KsFiQrFkQ1K2KrFkQAorDrFkQtVuVlgB4KrFiQAsgCrFkQlgBmgBrFiQAsgCrFkQtVuVhQsFv
            ]
          }
        ],
        "TimeThreshold": 300
      }
    ],
    "SnappedOrigin": [
      -123.11687,
      49.2813999
    ]
  }
}

```

cURL

```

curl --request POST \
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/isolines?key=Your_key' \
  --header 'Content-Type: application/json' \
  --data '{
    "Origin": [
      -123.11679620827039,
      49.28147612192166
    ],
    "DepartureTime": "2024-10-28T21:27:56Z",
    "Thresholds": {
      "Time": [
        300
      ]
    },
    "TravelMode": "Car"
  }

```

```
}'
```

AWS CLI

```
aws geo-routes calculate-isolines --key ${YourKey} \  
--origin -123.11679620827039 49.28147612192166 \  
--departure-time "2024-10-28T21:27:56Z" \  
--thresholds '{"Time": [300]}' \  
--travel-mode "Car"
```

How to calculate a service area based on distance

The CalculateIsolines API enables you to determine reachable service areas within a specified distance or time, factoring in road networks and traffic conditions. This feature can assist in planning fuel efficiency and defining accessible areas for service coverage.

Potential use cases

- **Plan fuel-efficient service areas:** Use distance-based calculations to optimize service coverage areas, helping reduce fuel costs and increase operational efficiency.

Examples

Calculate a service area based on distance with car TravelMode

Sample request

```
{  
  "Origin": [  
    -123.11679620827039,  
    49.28147612192166  
  ],  
  "DepartureTime": "2024-10-28T21:27:56Z",  
  "Thresholds": {  
    "Distance": [  
      4000  
    ]  
  },  
  "TravelMode": "Car"  
}
```

Sample response

```

{
  "DepartureTime": "2024-10-28T14:27:56-07:00",
  "IsolineGeometryFormat": "FlexiblePolyline",
  "Isolines": [
    {
      "Connections": [],
      "DistanceThreshold": 4000,
      "Geometries": [
        {
          "PolylinePolygon": [
            "BG-0s_9Cxtr9qHuViQuVsgCwVwV2KmgB1KmgBvVuV1KmgBAoiM2KmgBsgCuV-qB-
            qB4KmgBA8qB2KmgBsgCwV4KmgBA8qB3KmgBrgCuVAwVwVuV2KmgBAusJ4KmgBuVuV4KmgBA61C3KmgBtVwV3KmgBA2gE
            qB7qBsgCvVuVtVmgB3KmgB4KuVuVmgB4K81CAmgB3KuVrgCsgCrgCuVrgCuVtVwVrgC61C51CmgB3KmgB4KuVuVmgB4K
            qB7qB4KlgBA3gE2KlgB81C71CmgB1KmgB2K2KmgBA81C1KmgBvVuV1KmgBAq3K3KmgBlgB4KlgB3KrgCrgClgB3KlgB4
            qB3KmgBrgCuVtVuVrgCwVtVuVrgCuV9qB-qBtVsgCrgCsgC3KmgBA-
            qB1KkgBvVwVtVsgCtVuV3KmgBAy2G4KmgBsgCwVuVuVmgB4K8qBAmgB2K4KmgBA2rF3KmgBtVuVAuVsgCsgC4KmgB3Kmg
            qBlgB4KlgB3KpgCrgCvVA1KmgBA2rF3KmgBlgB2K9qBALgB1KtVrgCtVvV3KlgBA51C4KlgBuVtVAvVrgCrgC3KlgBA1
            qB3KmgBrgCsgC3KkgBA81C1KmgBvVuV1KmgBA-
            qB2KmgBwVuV2KmgBA2rF4KmgBuVA4KlgBA71C4KlgB8qB7qBmgB3K-
            qBAmgB4K2KmgBA8qB1KmgBtVwVvVsgCtVuV3KmgBA61C1KmgBvVuV1KmgBA-
            qB3KmgBtVuVAwVuVuV4KmgBA-qB3KmgBtVuV3KmgBA-
            qB3KmgB7qB8qBlgB4KlgB3KvVtVtVA3KmgBAy2G1KmgBvVuV1KmgBA-
            qB2KmgBwVuV2KmgBA4gE1KmgBvVwV1KmgBA8qB3KmgBrgCwVtVsgCrgCuVvVsgCtVuVtVsgCrgCuV3KmgBA-
            qB3KmgBtVuVtVsgC9qB-
            qBlgB4KlgB3K1KlgBAvhI3KlgBtVAvVuVlgB4K7qBALgB3KvVrgCtVtVtVrgCrgCtVvVrgCtVAAtVuVlgB4KlgB3KvVtV
            qBtVsgCvVuVtVsgCtVwV3KkgBA81C1KmgB9qB8qBlgB4KlgB3K3KlgBA7qB1KlgBvVvV1KlgBA7qB3KlgBrgCtV3KlgB
            qBAmgB1K2KlgB1KlgBrgCvV9qB7qBtVrgCtVvVvVrgCrgCrgC1KlgBA51C3KlgBrgCtV3KlgBA51C1KlgBlgB3K9qBAL
            qB7qBsgCvVuVrgCsgCtV4KlgBA9qB4KlgB61C51CsgCtV4KlgBA71C2KlgBsgCtV4KlgBA51C4KlgBqgCrgCwVrgCuVt
            qB9qBsgCtV4KlgBA9qB2KlgBqrDprDuVrgCsgCtV4KlgBA9qB4KlgBsgCtV2KlgBA9qB4KjgBmgB3K0rFAmgB3K4KlgB
          ]
        }
      ]
    }
  ],
  "SnappedOrigin": [
    -123.11687,
    49.2813999
  ]
}

```


cURL

```
curl --request POST \  
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/isolines?key=Your_key' \  
  --header 'Content-Type: application/json' \  
  --data '{  
    "Origin": [  
      -123.11679620827039,  
      49.28147612192166  
    ],  
    "DepartureTime": "2024-10-28T21:27:56Z",  
    "Thresholds": {  
      "Distance": [  
        4000  
      ]  
    },  
    "TravelMode": "Car"  
  }'
```

AWS CLI

```
aws geo-routes calculate-isolines --key ${YourKey} \  
  --origin -123.11679620827039 49.28147612192166 \  
  --departure-time "2024-10-28T21:27:56Z" \  
  --thresholds '{"Distance": [4000]}' \  
  --travel-mode "Car"
```

How to calculate a route matrix

Learn how to calculate a route matrix.

Topics

- [How to calculate the route matrix of distance and time for multiple origins and destinations](#)
- [How to calculate route matrix with avoidance](#)

How to calculate the route matrix of distance and time for multiple origins and destinations

The CalculateRouteMatrix API calculates routes and provides travel time and travel distance for each combination of origins and destinations. This capability is useful for applications requiring route planning and optimization across multiple locations.

Potential use cases

- **Optimize route planning:** Use the route matrix as input for route optimization software to enhance service efficiency and reduce travel time.

Examples

CalculateRouteMatrix with an unbounded routing boundary

Sample request

```
{
  "Origins": [
    {
      "Position": [-123.11679620827039, 49.28147612192166]
    },
    {
      "Position": [-123.11179620827039, 49.3014761219]
    }
  ],
  "Destinations": [
    {
      "Position": [-123.112317039, 49.28897192166]
    }
  ],
  "DepartureTime": "2024-05-28T21:27:56Z",
  "RoutingBoundary": {
    "Unbounded": true
  }
}
```

Sample response

```
{
```

```
"ErrorCount": 0,
"RouteMatrix": [
  [
    {
      "Distance": 1907,
      "Duration": 343
    }
  ],
  [
    {
      "Distance": 5629,
      "Duration": 954
    }
  ]
],
"RoutingBoundary": {
  "Unbounded": true
}
}
```

cURL

```
curl --request POST \
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/route-matrix?key=Your_key' \
  --header 'Content-Type: application/json' \
  --data '{
    "Origins": [
      {
        "Position": [-123.11679620827039, 49.28147612192166]
      },
      {
        "Position": [-123.11179620827039, 49.3014761219]
      }
    ],
    "Destinations": [
      {
        "Position": [-123.112317039, 49.28897192166]
      }
    ],
    "DepartureTime": "2024-05-28T21:27:56Z",
    "RoutingBoundary": {
      "Unbounded": true
    }
  }
```

```
}  
}'
```

AWS CLI

```
aws geo-routes calculate-route-matrix --key ${YourKey} \  
--origins '[{"Position": [-123.11679620827039, 49.28147612192166]}, {"Position":  
[-123.11179620827039, 49.3014761219]}]' \  
--destinations '[{"Position": [-123.11179620827039, 49.28897192166]}]' \  
--departure-time "2024-05-28T21:27:56Z" \  
--routing-boundary '{"Unbounded": true}'
```

CalculateRouteMatrix with a geometry-based routing boundary

Sample request

```
{  
  "Origins": [  
    {  
      "Position": [-123.11679620827039, 49.28147612192166]  
    },  
    {  
      "Position": [-123.11179620827039, 49.3014761219]  
    }  
  ],  
  "Destinations": [  
    {  
      "Position": [-123.112317039, 49.28897192166]  
    }  
  ],  
  "DepartureTime": "2024-05-28T21:27:56Z",  
  "RoutingBoundary": {  
    "Geometry": {  
      "AutoCircle": {  
        "Margin": 10000,  
        "MaxRadius": 30000  
      }  
    }  
  }  
}
```

Sample response

```
{
  "ErrorCount": 0,
  "RouteMatrix": [
    [
      {
        "Distance": 1907,
        "Duration": 344
      }
    ],
    [
      {
        "Distance": 5629,
        "Duration": 950
      }
    ]
  ],
  "RoutingBoundary": {
    "Geometry": {
      "Circle": {
        "Center": [
          -123.1142962082704,
          49.29147612191083
        ],
        "Radius": 11127
      }
    },
    "Unbounded": false
  }
}
```

cURL

```
curl --request POST \
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/route-matrix?key=Your_key' \
  --header 'Content-Type: application/json' \
  --data '{
    "Origins": [
      {
        "Position": [-123.11679620827039, 49.28147612192166]
      },
    ],
  }'
```

```

    {
      "Position": [-123.11179620827039, 49.3014761219]
    }
  ],
  "Destinations": [
    {
      "Position": [-123.112317039, 49.28897192166]
    }
  ],
  "DepartureTime": "2024-05-28T21:27:56Z",
  "RoutingBoundary": {
    "Geometry": {
      "AutoCircle": {
        "Margin": 10000,
        "MaxRadius": 30000
      }
    }
  }
}'

```

AWS CLI

```

aws geo-routes calculate-route-matrix --key ${YourKey} \
--origins '[{"Position": [-123.11679620827039, 49.28147612192166]}, {"Position": \
[-123.11179620827039, 49.3014761219]}]' \
--destinations '[{"Position": [-123.11179620827039, 49.28897192166]}]' \
--departure-time "2024-05-28T21:27:56Z" \
--routing-boundary '{"Geometry": {"AutoCircle": {"Margin": 10000, "MaxRadius": \
30000}}}'

```

How to calculate route matrix with avoidance

The CalculateRouteMatrix API computes routes and returns travel time and distance from each origin to each destination in the specified lists. The API can be used to set avoidance options for specific areas or road features, ensuring routes avoid specified zones or conditions. If an alternative route is not feasible, the avoidance preference may be bypassed.

Potential use cases

- **Route planning and optimization:** Use route matrix as input for software that requires optimized travel routes while avoiding certain areas or road features.

Examples

CalculateRouteMatrix with an avoidance area

Sample request

```
{
  "Origins": [
    {
      "Position": [-123.11679620827039, 49.28147612192166]
    }
  ],
  "Destinations": [
    {
      "Position": [-123.112317039, 49.28897192166]
    }
  ],
  "Avoid": {
    "Areas": [
      {
        "Geometry": {
          "BoundingBox": [
            -123.116561,
            49.281517,
            -123.110165,
            49.285689
          ]
        }
      }
    ]
  },
  "RoutingBoundary": {
    "Unbounded": true
  }
}
```

Sample response

```
{
  "ErrorCount": 0,
  "RouteMatrix": [
    {
```

```

        "Distance": 1855,
        "Duration": 295
      }
    ]
  ],
  "RoutingBoundary": {
    "Unbounded": true
  }
}

```

cURL

```

curl --request POST \
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/route-matrix?key=Your_key' \
  --header 'Content-Type: application/json' \
  --data '{
    "Origins": [
      {
        "Position": [-123.11679620827039, 49.28147612192166]
      }
    ],
    "Destinations": [
      {
        "Position": [-123.112317039, 49.28897192166]
      }
    ],
    "Avoid": {
      "Areas": [
        {
          "Geometry": {
            "BoundingBox": [
              -123.116561,
              49.281517,
              -123.110165,
              49.285689
            ]
          }
        }
      ]
    }
  },
  "RoutingBoundary": {
    "Unbounded": true
  }
}

```



```

    [
      {
        "Distance": 1855,
        "Duration": 295
      }
    ],
    "RoutingBoundary": {
      "Unbounded": true
    }
  }
}

```

cURL

```

curl --request POST \
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/route-matrix?key=Your_key' \
  --header 'Content-Type: application/json' \
  --data '{
    "Origins": [
      {
        "Position": [-123.11679620827039, 49.28147612192166]
      }
    ],
    "Destinations": [
      {
        "Position": [-123.112317039, 49.28897192166]
      }
    ],
    "Avoid": {
      "TollRoads": true,
      "ControlledAccessHighways": true,
      "Ferries": true
    },
    "RoutingBoundary": {
      "Unbounded": true
    }
  }'

```

AWS CLI

```

aws geo-routes calculate-route-matrix --key ${YourKey} \
  --origins '[{"Position": [-123.11679620827039, 49.28147612192166]}]' \

```

```
--destinations '[{"Position": [-123.112317039, 49.28897192166]]' \  
--avoid '{"TollRoads": true, "ControlledAccessHighways": true, "Ferries": true}' \  
--routing-boundary '{"Unbounded": true}'
```

How to optimize waypoints

Learn how to optimize waypoints.

Topics

- [How to optimize waypoints for a route](#)
- [How to optimize waypoints for a route with traffic awareness](#)

How to optimize waypoints for a route

The OptimizeWaypoints API calculates the most efficient route between a series of waypoints, minimizing either travel time or total distance. This API solves the Traveling Salesman Problem by considering road networks and traffic conditions to determine the optimal path.

Potential use cases

- **Analyze service area patterns:** Use waypoint optimization to make informed decisions about business service areas and improve logistics efficiency.

Examples

Optimize waypoints using Car TravelMode

Sample Request

```
{  
  "Origin": [  
    -123.095740,  
    49.274426  
  ],  
  "Waypoints": [  
    {  
      "Position": [  
        -123.115193,  
        49.280596  
      ]  
    }  
  ]  
}
```

```
    ]
  },
  {
    "Position": [
      -123.089557,
      49.271774
    ]
  }
],
"DepartureTime": "2024-10-25T18:13:42Z",
"Destination": [
  -123.095185,
  49.263728
],
"TravelMode": "Car"
}
```

Sample Response

```
{
  "Connections": [
    {
      "Distance": 1989,
      "From": "Origin",
      "RestDuration": 0,
      "To": "Waypoint0",
      "TravelDuration": 258,
      "WaitDuration": 0
    },
    {
      "Distance": 3010,
      "From": "Waypoint0",
      "RestDuration": 0,
      "To": "Waypoint1",
      "TravelDuration": 298,
      "WaitDuration": 0
    },
    {
      "Distance": 2371,
      "From": "Waypoint1",
      "RestDuration": 0,
      "To": "Destination",
      "TravelDuration": 311,
    }
  ]
}
```

```
        "WaitDuration": 0
      }
    ],
    "Distance": 7370,
    "Duration": 867,
    "ImpedingWaypoints": [],
    "OptimizedWaypoints": [
      {
        "DepartureTime": "2024-10-25T18:13:42Z",
        "Id": "Origin",
        "Position": [
          -123.09574,
          49.274426
        ]
      },
      {
        "DepartureTime": "2024-10-25T18:18:00Z",
        "Id": "Waypoint0",
        "Position": [
          -123.115193,
          49.280596
        ]
      },
      {
        "DepartureTime": "2024-10-25T18:22:58Z",
        "Id": "Waypoint1",
        "Position": [
          -123.089557,
          49.271774
        ]
      },
      {
        "ArrivalTime": "2024-10-25T18:28:09Z",
        "Id": "Destination",
        "Position": [
          -123.095185,
          49.263728
        ]
      }
    ],
    "TimeBreakdown": {
      "RestDuration": 0,
      "ServiceDuration": 0,
      "TravelDuration": 867,
    }
  }
}
```

```
    "WaitDuration": 0
  }
}
```

cURL

```
curl --request POST \
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/optimize-waypoints?
key=Your_key' \
  --header 'Content-Type: application/json' \
  --data '{
  "Origin": [
    -123.095740,
    49.274426
  ],
  "Waypoints": [
    {
      "Position": [
        -123.115193,
        49.280596
      ]
    },
    {
      "Position": [
        -123.089557,
        49.271774
      ]
    }
  ],
  "DepartureTime": "2024-10-25T18:13:42Z",
  "Destination": [
    -123.095185,
    49.263728
  ],
  "TravelMode": "Car"
}'
```

AWS CLI

```
aws geo-routes optimize-waypoints --key ${YourKey} \
--origin -123.095740 49.274426 \
--waypoints '[{"Position": [-123.115193 , 49.280596]}, {"Position": [-123.089557 ,
49.271774]}]' \
```

```
--destination -123.095185 49.263728 \  
--departure-time "2024-10-25T18:13:42Z" \  
--travel-mode "Car"
```

How to optimize waypoints for a route with traffic awareness

The OptimizeWaypoints API calculates the optimal route between multiple waypoints to minimize travel time or total distance. It utilizes advanced algorithms to solve the Traveling Salesman Problem, determining the most efficient path while accounting for factors such as road networks and real-time traffic conditions.

Potential use cases

- **Optimize multi-stop routes for delivery efficiency:** Improve delivery operations by calculating the shortest or fastest route among several stops. This is useful for reducing operational costs, fuel consumption, and travel time in logistics and delivery services.

Examples

Optimize waypoints with traffic awareness using car TravelMode

Sample request

```
{  
  "Origin": [  
    -123.095740,  
    49.274426  
  ],  
  "Waypoints": [  
    {  
      "Position": [  
        -123.115193,  
        49.280596  
      ]  
    },  
    {  
      "Position": [  
        -123.089557,  
        49.271774  
      ]  
    }  
  ]  
}
```

```
],
  "DepartureTime": "2024-10-25T18:13:42Z",
  "Destination": [
    -123.095185,
    49.263728
  ],
  "TravelMode": "Car",
  "Traffic": {
    "Usage": "UseTrafficData"
  }
}
```

Sample response

```
{
  "Connections": [
    {
      "Distance": 1989,
      "From": "Origin",
      "RestDuration": 0,
      "To": "Waypoint0",
      "TravelDuration": 324,
      "WaitDuration": 0
    },
    {
      "Distance": 2692,
      "From": "Waypoint0",
      "RestDuration": 0,
      "To": "Waypoint1",
      "TravelDuration": 338,
      "WaitDuration": 0
    },
    {
      "Distance": 2371,
      "From": "Waypoint1",
      "RestDuration": 0,
      "To": "Destination",
      "TravelDuration": 395,
      "WaitDuration": 0
    }
  ],
  "Distance": 7052,
  "Duration": 1057,
```



```
"ImpedingWaypoints": [],
"OptimizedWaypoints": [
  {
    "DepartureTime": "2024-10-25T18:13:42Z",
    "Id": "Origin",
    "Position": [
      -123.09574,
      49.274426
    ]
  },
  {
    "ArrivalTime": "2024-10-25T18:19:06Z",
    "DepartureTime": "2024-10-25T18:19:06Z",
    "Id": "Waypoint0",
    "Position": [
      -123.115193,
      49.280596
    ]
  },
  {
    "ArrivalTime": "2024-10-25T18:24:44Z",
    "DepartureTime": "2024-10-25T18:24:44Z",
    "Id": "Waypoint1",
    "Position": [
      -123.089557,
      49.271774
    ]
  },
  {
    "ArrivalTime": "2024-10-25T18:31:19Z",
    "Id": "Destination",
    "Position": [
      -123.095185,
      49.263728
    ]
  }
],
"TimeBreakdown": {
  "RestDuration": 0,
  "ServiceDuration": 0,
  "TravelDuration": 1057,
  "WaitDuration": 0
}
```

```
}
```

cURL

```
curl --request POST \  
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/optimize-waypoints?key=Your_key' \  
  --header 'Content-Type: application/json' \  
  --data '{  
    "Origin": [  
      -123.095740,  
      49.274426  
    ],  
    "Waypoints": [  
      {  
        "Position": [  
          -123.115193,  
          49.280596  
        ]  
      },  
      {  
        "Position": [  
          -123.089557,  
          49.271774  
        ]  
      }  
    ],  
    "DepartureTime": "2024-10-25T18:13:42Z",  
    "Destination": [  
      -123.095185,  
      49.263728  
    ],  
    "TravelMode": "Car",  
    "Traffic": {  
      "Usage": "UseTrafficData"  
    }  
  }'  
'
```

AWS CLI

```
aws geo-routes optimize-waypoints --key ${YourKey} \  
  --origin -123.095740 49.274426 \  
  --destination -123.095185 49.263728
```

```
--waypoints '[{"Position": [-123.115193 , 49.280596]}, {"Position": [-123.089557 , 49.271774]]]' \  
--destination -123.095185 49.263728 \  
--departure-time "2024-10-25T18:13:42Z" \  
--travel-mode "Car" \  
--traffic '{"Usage": "UseTrafficData"}'
```

How to use SnapToRoads

This topic explains how to use the SnapToRoads API to align GPS traces with road networks, enhancing positional accuracy for navigation and fleet management applications. This API corrects GPS drift and signal loss by snapping coordinates to the nearest road segments, while respecting travel mode restrictions. Examples illustrate practical uses, such as overlaying GPS traces, filling data gaps, and reducing noise for clearer route visualization.

Topics

- [How to match GPS traces to road network](#)

How to match GPS traces to road network

The SnapToRoads API allows you to match GPS traces onto the road network. A GPS trace includes positions and metadata like timestamp, speed, and heading that are recorded using a GPS device. These traces often have a margin of error, making them challenging to use for analysis and visualization directly.

SnapToRoads considers legal and time restrictions for the specified travel mode while matching traces. If the trace strongly suggests a restriction violation, the actual route taken is maintained.

Potential use cases

- **Overlay GPS traces onto the most likely driven roads:** This feature helps align GPS data to the most accurate path on the road network, supporting clearer data visualization.
- **Interpolate gaps in GPS traces:** SnapToRoads can fill in gaps by snapping coordinates to road segments, creating a more continuous and useful dataset for applications.
- **Filter noise and outliers:** By snapping to the nearest road, this API can help remove outliers and reduce GPS noise, improving data reliability for analysis.

Examples

Match GPS trace using car mode

Sample request

```
{
  "TracePoints": [
    {
      "Position": [8.53404,50.16364],
      "Timestamp": "2024-05-22T18:13:42Z"
    },
    {
      "Position": [8.53379056,50.16352417],
      "Speed": 20,
      "Timestamp": "2024-05-22T18:13:59Z"
    }
  ],
  "TravelMode": "Car"
}
```

Sample response

```
{
  "Notices": [],
  "SnappedGeometry": {
    "Polyline": "BGs931_Cog8oQTnBnGzP"
  },
  "SnappedGeometryFormat": "FlexiblePolyline",
  "SnappedTracePoints": [
    {
      "Confidence": 1,
      "OriginalPosition": [8.53404, 50.16364],
      "SnappedPosition": [8.53402, 50.16367]
    },
    {
      "Confidence": 0.86,
      "OriginalPosition": [8.53379056, 50.16352417],
      "SnappedPosition": [8.53375, 50.16356]
    }
  ]
}
```

cURL

```
curl --request POST \  
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/snap-to-roads?key=Your_key' \  
  --header 'Content-Type: application/json' \  
  --data '{  
    "TracePoints": [  
      {  
        "Position": [8.53404,50.16364],  
        "Timestamp": "2024-05-22T18:13:42Z"  
      },  
      {  
        "Position": [8.53379056,50.16352417],  
        "Speed": 20,  
        "Timestamp": "2024-05-22T18:13:59Z"  
      }  
    ],  
    "TravelMode": "Car"  
  }'
```

AWS CLI

```
aws geo-routes snap-to-roads --key ${YourKey} \  
--trace-points '[{"Position": [8.53404, 50.16364], "Timestamp": "2024-05-22T18:13:42Z"}, {"Position": [8.53379056, 50.16352417], "Speed": 20, "Timestamp": "2024-05-22T18:13:59Z"}]' \  
--travel-mode "Car"
```

Match GPS trace using truck mode with options

Sample request

```
{  
  "TracePoints": [  
    {  
      "Position": [8.53404,50.16364],  
      "Timestamp": "2024-05-22T18:13:42Z"  
    },  
    {  
      "Position": [8.53379056,50.16352417],  
      "Speed": 20,  
      "Timestamp": "2024-05-22T18:13:59Z"  
    }  
  ]  
}
```

```

    "Timestamp": "2024-05-22T18:13:59Z"
  }
],
"TravelMode": "Truck",
"TravelModeOptions": {
  "Truck": {
    "GrossWeight": 10000
  }
}
}

```

Sample response

```

{
  "Notices": [],
  "SnappedGeometry": {
    "Polyline": "BGs931_Cog8oQTnBnGzP"
  },
  "SnappedGeometryFormat": "FlexiblePolyline",
  "SnappedTracePoints": [
    {
      "Confidence": 1,
      "OriginalPosition": [8.53404, 50.16364],
      "SnappedPosition": [8.53402, 50.16367]
    },
    {
      "Confidence": 0.86,
      "OriginalPosition": [8.53379056, 50.16352417],
      "SnappedPosition": [8.53375, 50.16356]
    }
  ]
}

```

cURL

```

curl --request POST \
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/snap-to-roads?
key=Your_key' \
  --header 'Content-Type: application/json' \
  --data '{
"TracePoints": [
  {
    "Position": [8.53404,50.16364],

```

```
    "Timestamp": "2024-05-22T18:13:42Z"
  },
  {
    "Position": [8.53379056,50.16352417],
    "Speed": 20,
    "Timestamp": "2024-05-22T18:13:59Z"
  }
],
"TravelMode": "Truck",
"TravelModeOptions": {
  "Truck": {
    "GrossWeight": 10000
  }
}
}'
```

AWS CLI

```
aws geo-routes snap-to-roads --key ${YourKey} \
--trace-points '[{"Position": [8.53404, 50.16364], "Timestamp":
"2024-05-22T18:13:42Z"}, {"Position": [8.53379056, 50.16352417], "Speed": 20,
"Timestamp": "2024-05-22T18:13:59Z"}]' \
--travel-mode "Truck" \
--travel-mode-options '{"Truck": {"GrossWeight": 10000}}'
```

Manage costs and usage

As you continue learning about Amazon Location routes, it's important to understand how to manage service capacity, ensure you follow usage limits, and get the best results through quota and API optimizations. By applying best practices for performance and accuracy, you can tailor your application to handle place-related queries efficiently and maximize your API requests.

Topics

- [Best Practices](#)
- [Routes pricing](#)
- [Routes Quota and Usage](#)

Best Practices

This section covers best practices for using compression and choosing between Simple (GeoJSON) and FlexiblePolyline formats when interacting with the API, providing guidance on optimizing performance, bandwidth, and data handling.

Compression

To enhance the performance and efficiency of your applications when interacting with our API, it is recommended to enable compression for responses, especially when dealing with large text-based payloads. You can activate compression by including the `Accept-Encoding` header in your API requests, specifying your preferred compression method. We support `gzip` and `deflate` for their compression capabilities, with `gzip` typically offering better compression ratios.

When to Enable Compression

Large Responses

Enable compression for large text-based responses to reduce bandwidth usage and improve load times.

Network Constraints

If your application operates over limited bandwidth or high-latency networks, compression can enhance data transfer efficiency.

How to Use Compression Effectively

Set the Accept-Encoding Header

Include `Accept-Encoding: gzip, deflate` in your HTTP requests to inform our API that you support these compression methods. The method to enable and handle compression varies by [AWS SDK](#) and programming language. For example, the [AWS SDK for Java v1](#) uses the `withGzip` method in the `ClientConfiguration` class to enable `gzip`, while the AWS SDK for Go requires adding specific middleware for compression handling. For other SDKs, please refer to the [AWS SDK Reference Guide](#) for detailed instructions.

Handle Decompression Properly

Ensure your client application can correctly decompress the responses based on the `Content-Encoding` header returned by our API.

Test and Monitor

Regularly evaluate the impact of compression on your application's performance, balancing the benefits of reduced payload sizes against any additional CPU overhead from decompression processes.

Polyline

Best practices for choosing between Simple (GeoJSON) and FlexiblePolyline formats when interacting with our API, to optimize both performance and usability of your geospatial data.

Use Simple (GeoJSON) Format

Readability and Standardization

Use when you require a widely recognized and human-readable format for ease of debugging and interoperability with various geospatial tools.

Precision

Choose Simple format when your application needs high precision for coordinates, as GeoJSON maintains full decimal precision without loss.

Smaller Datasets

Simple format is ideal when working with smaller sets of coordinate data where the size reduction benefits of compression are minimal.

Use FlexiblePolyline Format

Data Size Reduction

FlexiblePolyline is ideal when you need to minimize the amount of data transmitted, especially for large lists of coordinates, by leveraging lossy compression techniques.

URL Safety

FlexiblePolyline provides a compact, URL-safe string that can be used directly in query parameters without additional encoding.

Performance Optimization

FlexiblePolyline helps reduce the payload size, leading to faster data transfer and lower bandwidth usage, making it crucial for high-performance applications or those operating over constrained networks.

Routes pricing

Please see below for pricing buckets for each API:

Calculate Routes

This price is based on the number of routes calculated. CalculateRoutes has three pricing buckets: Core, Advanced, and Premium.

Core

This price bucket supports the travel modes Car, Truck, and Pedestrian, without toll cost calculation.

Advanced

This price bucket supports alternative travel modes such as Scooter, without toll cost calculation.

Premium

This price bucket supports toll cost calculation. You will be charged at Premium price when you request toll cost calculation by setting the request parameters `LegAdditionalFeatures["Tolls"]` or `SpanAdditionalFeatures["TollSystems"]`, regardless of travel mode.

Calculate Route Matrix

This price is based on the number of routes calculated. The number of routes calculated in each request is equal to the number of origins multiplied by the number of destinations, $\text{Number of Routes} = \text{Number of origins} \times \text{Number of Destinations}$. For example, when using a matrix size of 300 origins by 100 destinations, the total number of routes calculated is 30,000 ($300 \times 100 = 30,000$).

Note

Route calculations are billed for each origin and destination pair. If you use a large matrix of origins and destinations, your costs will increase accordingly.

`CalculateRouteMatrix` has 2 pricing buckets: Core and Advanced.

Core

This price bucket supports travel modes Car, Truck, and Pedestrian.

Advanced

This price bucket supports alternative travel modes, such as Scooter.

Optimize Waypoint

This price is based on the number of API requests. `OptimizeWaypoint` has 2 pricing buckets: Advanced and Premium.

Advanced

This pricing bucket supports travel modes Car, Truck and Pedestrian, with the bounding box of the input points within 200km, and no optional parameters (such as `Avoid`, `Driver`, `Exclude.Countries`, `TravelModeOptions.Truck.HazardousCargos`, `TravelModeOptions.Truck.TunnelRestrictionCode`, and no additional waypoints or destination constraints such as `AccessHours`, `AppointmentTime`, `Before`, `Heading`, `ServiceDuration`, `SideOfStreet`).

Premium

This pricing bucket has no restrictions on travel modes and supports optional parameters, including: `Avoid`, `Driver`, `Exclude.Countries`, `TravelModeOptions.Truck.HazardousCargos`, `TravelModeOptions.Truck.TunnelRestrictionCode`, and optional waypoints or destination constraints such as: `AccessHours`, `AppointmentTime`, `Before`, `Heading`, `ServiceDuration`, `SideOfStreet`.

Both Advanced and Premium price buckets support up to 20 waypoints in a single request.

Snap-to-road

This price is based on the number of API requests. SnaptoRoad has 2 pricing buckets: Advanced and Premium.

Advanced

This pricing bucket supports travel modes Car, Truck and Pedestrian, with a TracePoints count up to 200 and with a maximum airline distance between TracePoints of 100 kilometers.

Premium

This pricing bucket has no restrictions on travel modes, up to 5,000 TracePoints points.

Calculate Isoline

This price is based on the number of Isolines calculated in the response. CalculateIsoline has 2 pricing buckets: Advanced and Premium.

Advanced

This pricing bucket supports travel modes Car, Truck and Pedestrian, with Thresholds .Time values up to 60 minutes or Thresholds .Distance values up to 100 kilometers.

Premium

This pricing bucket has no restrictions on travel modes, with Thresholds .Time values up to 180 minutes or Thresholds .Distance values up to 300KM.

Routes Quota and Usage

Service Quota

Amazon Location Service APIs have default quotas. You can increase quotas using the [service quota console](#). For limits exceeding 2x the default, request via the self-service console or contact support.

Service Quota Limits

API Name	Default	Max Adjustable Limit	More than Adjustable Max Limit
the section called "Calculate routes"	20	40	Request on service quota console or contact support team
Calculatelsoline	20	40	Request on service quota console or contact support team
the section called "Snap to Roads"	20	40	Request on service quota console or contact support team
the section called "Calculate route matrix"	5	10	Request on service quota console or contact support team
the section called "Optimize waypoints"	5	10	Request on service quota console or contact support team

Other Usage Limits

In addition to service quotas, the following API usage limits apply:

Other Usage Limits

API Name	Limit	Value
the section called "Snap to Roads"	Sum of Geodesic distance between all TracePoints	500KM
the section called "Optimize waypoints"	Sum of Geodesic distance between the Origin,	100KM

API Name	Limit	Value
	Waypoints in the provided ordering, and Destination	
the section called "Optimize waypoints"	Perimeter of the bounding box surrounding the Origin, Waypoints, and Destination	500KM
the section called "Calculate route matrix"	Max Distance between Origins and Destinations for Unbounded routing (If Avoid or TravelModeOptions.Truck is used)	60KM
the section called "Calculate route matrix"	Max Distance between Origins and Destinations for Unbounded routing	10000KM
the section called "Calculate routes"	Response payload size after compression	6MB
the section called "Calculate route matrix"	Response payload size after compression	6MB
the section called "Calculate isolines"	Response payload size after compression	6MB
the section called "Optimize waypoints"	Response payload size after compression	6MB
the section called "Snap to Roads"	Response payload size after compression	6MB

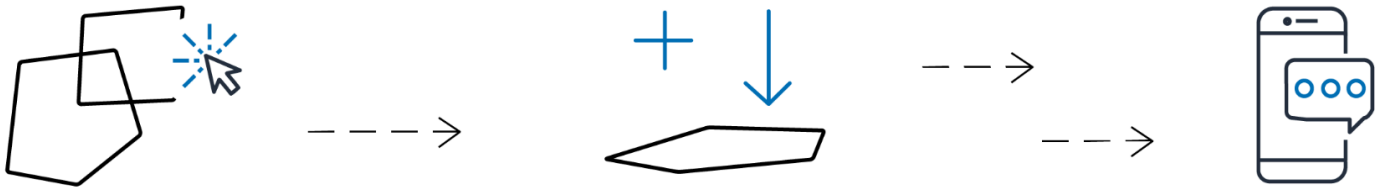
Next Steps

Please check the following for further details:

- [Attribution](#): Information on data attribution requirements for Amazon Location Service.

- [SLA](#): The service level agreement for Amazon Location Service, including uptime commitments and response times.
- [Service Terms](#): Terms governing the use of Amazon Location Service, including restrictions and limitations.

Amazon Location Service Geofences



Geofence collection resources allow you to store and manage geofences - virtual boundaries on a map. You can evaluate locations against a geofence collection resource and receive notifications when the location update crosses the boundary of any of the geofences in the collection.

Geofences and geofence collection

A geofence is a polygon or circle geometry that defines a virtual boundary on a map. A geofence collection contains zero or more geofences. It's capable of geofence monitoring by emitting ENTER and EXIT events, when requested, to evaluate a device position against its geofences.

Geofence events

Locations for positions you're monitoring are referenced by an ID called a `DeviceId`. The positions are referred to as device positions. You can send a list of device positions to evaluate directly to the geofence collection resource, or you can use a tracker. For more information about using trackers, see [Trackers](#).

You receive events (via Amazon EventBridge) only when a device enters or exits a geofence, not for every position change. This means that you will typically receive events and have to respond to them much less frequently than every device position update.

Note

For the first location evaluation for a specific `DeviceID`, it is assumed that the device was previously not in any geofences. So the first update will generate an ENTER event, if inside a geofence in the collection, and no event if not.

In order to calculate whether a device has entered or exited a geofence, Amazon Location Service must keep previous position state for the device. This position state is stored for 30 days. After 30 days without an update for a device, a new location update will be treated as the first position update.

Use cases for Amazon Location Service Geofences

The following are a few common uses for Amazon Location Service Geofences.

Improve field service operations

Keep a pulse on your mobile workforce with real-time tracking. Set geofences around customer sites and service areas to receive alerts when staff arrive and depart. Use location data to optimize scheduling, dispatch the nearest available technician, and reduce response times. Empower your field teams (such as a your plumbing or HVAC repair business) to work more efficiently, while enhancing the customer experience.

Monitor and control critical assets

Utilize Amazon Location Service to track the real-time location and status of your valuable equipment, inventory, and other mobile assets. Set up geofences to receive alerts on unauthorized movements or removals, enhancing security and compliance. Use this location visibility to improve asset utilization, optimize maintenance schedules, and ensure your critical resources are accounted for at all times. Always monitor your heavy machinery, IT hardware, or retail inventory with precision, reduce losses, and make more informed operational decisions.

Enhance supply chain visibility

Leverage Amazon Location Service to track shipments and deliveries across your entire supply chain. Define geofences around distribution centers, stores, and other key facilities to monitor the movement of inventory and assets. Use real-time location data to improve inventory management, optimize logistics planning, and deliver a superior customer experience. Gain end-to-end visibility into your supply chain operations, identify bottlenecks, and make data-driven decisions that drive efficiency and responsiveness.

Strengthen safety and security

Geofencing enables you to set up virtual boundaries around secure areas, restricted zones, and other critical locations. Receive instant alerts when unauthorized personnel or assets enter or exit these predefined geofences. Leverage this real-time location monitoring to enhance workplace safety, deter trespassing, and ensure regulatory compliance. Whether you manage a manufacturing facility, construction site, or corporate campus, geofencing empowers you to maintain tighter control over access, improve incident response, and protect your people, property, and assets.

Location-based marketing

Unlock the power of location data to supercharge your geomarketing efforts. Use Amazon Location Service to set virtual boundaries around competitor locations, events, and high-traffic areas. Trigger personalized ads, offers, and notifications when customers enter these geofenced zones. Analyze foot traffic patterns to optimize ad placements and uncover prime sites for new business locations. Monitor customer movements within your own geofenced spaces to gain deeper insights on browsing behaviors and path-to-purchase. Combine real-time location tracking with precision geofencing to deliver hyper-targeted, contextual engagement that drives sales and loyalty in the physical world.

Geofence concepts

This section provides some common geofence concepts, including common terminology and how to manage geofences.

Amazon Location Service geofence terminology

Geofence collection

Contains zero or more geofences. It is capable of geofence monitoring by emitting Entry and Exit events, when requested, to evaluate a device position against its geofences.

Geofence

A polygon or circle geometry that defines a virtual boundary on a map.

Polygon geometry

An Amazon Location geofence is a virtual boundary for a geographical area and is represented as a polygon geometry or as a circle.

A circle is a point with a distance around it. Use a circle when you want to be notified if a device is within a certain distance of a location.

A polygon is an array composed of 1 or more linear rings. Use a polygon when you want to define a specific boundary for device notifications. A linear ring is an array of four or more vertices, where the first and last vertex are the same to form a closed boundary. Each vertex is a 2-dimensional point of the form `[longitude, latitude]`, where the units of longitude and latitude are degrees. The vertices must be listed in counter-clockwise order around the polygon.

The following is an example of a single linear external ring:

```
[
```

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

Note

Amazon Location Service doesn't support polygons with more than one ring. This includes holes, islands, or multipolygons. Amazon Location also doesn't support polygons that are wound clockwise, or that cross the antimeridian.


Get started with Amazon Location Service Geofences

Geofences are powerful tools for defining geographic boundaries and triggering actions based on location updates. This guide walks you through the process of creating and using geofence collection resources in Amazon Location. By setting up geofences and evaluating locations against them, you can monitor movement and generate automated events, such as notifications when a device enters or exits a defined area. These features are ideal for applications like fleet tracking, location-based notifications, and more.

1. Create a geofence collection resource in your AWS account.
2. Add geofences to the collection. You can use the geofence upload tool on the Amazon Location console or the Amazon Location Geofences API. For more information about available options, see [Authentication](#). Geofences can either be defined by a polygon or by a circle. Use a polygon to find when a device enters a specific area. Use a circle to find when a device comes within a certain distance (radius) of a point.
3. You can start evaluating locations against all your geofences. When a location update crosses the boundaries of one or more geofences, your geofence collection resource emits one of the following geofence event types on Amazon EventBridge:
 - **ENTER** – One event is generated for each geofence where the location update crosses its boundary by entering it.
 - **EXIT** – One event is generated for each geofence where the location update crosses its boundary by exiting it.

For more information, see [the section called “React to events with EventBridge”](#). You can also integrate monitoring using services such as Amazon CloudWatch and AWS CloudTrail. For more information see, [the section called “Monitor with Amazon CloudWatch”](#) and [the section called “Monitor and log with AWS CloudTrail”](#).

For example, you are tracking a fleet of trucks and want to be notified when a truck comes within a certain area of any of your warehouses. Create a geofence for the area around each warehouse. When the trucks send you updated locations, use Amazon Location Service to evaluate those positions and see if a truck has entered (or exited) one of the geofence areas.

 **Note**

You're billed by the number of geofence collections you evaluate against. Your bill is not affected by the number of geofences in each collection. Since each geofence collection may contain up to 50,000 geofences, you may want to combine your geofences into fewer collections, where possible, to reduce your cost of geofence evaluations. The events generated will include the ID of the individual geofence in the collection, as well as the ID of the collection.

How to work with Amazon Location Service Geofences

section provides step-by-step guidance for working with geofence-related tasks in Amazon Location. Learn how to evaluate device positions against geofences, respond to geofence events using Amazon EventBridge, and effectively manage your geofence resources. These tutorials are designed to help you implement key functionality for tracking and managing location-based events with ease.

Topics

- [Evaluate device positions against geofences](#)
- [React to Amazon Location Service events with Amazon EventBridge](#)
- [Manage your geofence collection resources](#)

Evaluate device positions against geofences

There are two ways to evaluate positions against geofences to generate geofence events:

- You can link Trackers and Geofence Collections. For more information, see the section: [Link a tracker to a geofence collection](#).
- You can make a direct request to the geofence collection resource to evaluate one or more positions.

If you also want to track your device location history or display locations on a map, link the tracker with a geofence collection. Alternatively, you may not want to evaluate all location updates, or you don't intend to store location data in a tracker resource. If either of these is the case, you can make a direct request to the geofence collection and evaluate one or more device positions against its geofences.

Evaluating device positions against geofences generates events. You can react to these events and route them to other AWS services. For more information about actions that you can take when receiving geofence events, see [Reacting to Amazon Location Service events with Amazon EventBridge](#).

An Amazon Location event includes the attributes of the device position update that generates it, including the time, position, accuracy, and key-value metadata, and some attributes of the geofence that is entered or exited. For more information about the data included in a geofence event, see [the section called "Event examples"](#).

The following examples use the AWS CLI, or the Amazon Location APIs.

API

To evaluate device positions against the position of geofences using the Amazon Location APIs

Use the [BatchEvaluateGeofences](#) operation from the Amazon Location Geofences APIs.

The following example uses an API request to evaluate the position of device *ExampleDevice* to an associated geofence collection *ExampleGeofenceCollection*. Replace these values with your own geofence and device IDs.

```
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

To evaluate device positions against the position of geofences using AWS CLI commands

Use the [batch-evaluate-geofences](#) command.

The following example uses an AWS CLI to evaluate the position of *ExampleDevice* against an associated geofence collection *ExampleGeofenceCollection*. Replace these values with your own geofence and device IDs.

```
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"}}]'
```

React to Amazon Location Service events with Amazon EventBridge

Amazon EventBridge is a serverless event bus that efficiently connects applications together using data from AWS services like Amazon Location. EventBridge receives events from Amazon Location and routes that data to targets like AWS Lambda. You can set up routing rules to determine where to send your data to build application architectures that react in real time.

Only geofence events (ENTER and EXIT events, as devices enter or leave the geofenced areas) are sent to EventBridge by default. You can also enable all filtered position update events for a tracker resource. For more information, see [the section called “Enable update events for a tracker”](#).

For more information, see [the Events and Event Patterns](#) in *the Amazon EventBridge User Guide*.

Topics

- [Enable update events for a tracker](#)
- [Create event rules for Amazon Location](#)
- [Amazon EventBridge event examples for Amazon Location Service](#)

Enable update events for a tracker

By default, Amazon Location sends only ENTER and EXIT geofence events to EventBridge. You can enable all filtered position UPDATE events for a tracker to be sent to EventBridge. You can do this when you [create](#) or [update](#) a tracker.

For example, to update an existing tracker using the AWS CLI, you can use the following command (use the name of your tracker resource in place of *MyTracker*).

```
aws location update-tracker --tracker-name MyTracker --event-bridge-enabled
```

To turn off position events for a tracker, you must use the API or the Amazon Location Service console.

Create event rules for Amazon Location

You can create [up to 300 rules per event bus](#) in EventBridge to configure actions taken in response to an Amazon Location event.

For example, you can create a rule for geofence events where a push notification will be sent when a phone is detected within a geofenced boundary.

To create a rule for Amazon Location events

Using the following values, [create an EventBridge rule](#) based on Amazon Location events:

- For **Rule type**, choose **Rule with an event pattern**.
- In the **Event pattern** box, add the following pattern:

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"]
}
```

To create a rule for tracker position updates, you can instead use the following pattern:

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Device Position Event"]
}
```

You can optionally specify only ENTER or EXIT events by adding a `detail` tag (if your rule is for tracker position updates, there is only a single `EventType`, so there is no need to filter on it):

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"],
  "detail": {
    "EventType": ["ENTER"]
  }
}
```

You can also optionally filter on properties of the position or geofence:

```
{
  "source": ["aws.geo"],
  "detail-type": ["Location Geofence Event"],
  "detail": {
    "EventType": ["ENTER"],
    "GeofenceProperties": {
      "Type": "LoadingDock"
    },
    "PositionProperties": {
      "VehicleType": "Truck"
    }
  }
}
```


- For **Select targets**, choose the target action to take when an event is received from Amazon Location Service.

For example, use an Amazon Simple Notification Service (SNS) topic to send an email or text message when an event occurs. You first need to create an Amazon SNS topic using the Amazon SNS console. For more information, see [Using Amazon SNS for user notifications](#).

Warning

It's best practice to confirm that the event rule was successfully applied or your automated action may not initiate as expected. To verify your event rule, initiate conditions for the event rule. For example, simulate a device entering a geofenced area.

You can also capture all events from Amazon Location, by just excluding the `detail-type` section. For example:

```
{
  "source": [
    "aws.geo"
  ]
}
```

Note

The same event may be delivered more than one time. You can use the event id to deduplicate the events that you receive.

Amazon EventBridge event examples for Amazon Location Service

The following is an example of an event for entering a geofence initiated by calling `BatchUpdateDevicePosition`.

```
{
  "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"
  }
}
}

```

The following is an example of an event for exiting a geofence initiated by calling `BatchUpdateDevicePosition`.

```

{
  "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"
    }
  }
}

```

The following is an example of an event for a position update, initiated by calling `BatchUpdateDevicePosition`.

```

{
  "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"
}
}
```

Manage your geofence collection resources

Manage your geofence collections using the Amazon Location console, the AWS CLI, or the Amazon Location APIs.

List your geofence collection resources

You can view your geofence collection list using the Amazon Location console, the AWS CLI, or the Amazon Location APIs:

Console

To view a list of geofence collections using the Amazon Location console

1. Open the Amazon Location console at <https://console.aws.amazon.com/location/>.
2. Choose **Geofence collections** from the left navigation pane.
3. View a list of your geofence collections under **My geofence collections**.

API

Use the [ListGeofenceCollections](#) operation from the Amazon Location Geofences APIs.

The following example is an API request to get a list of geofence collections in the AWS account.

```
POST /geofencing/v0/list-collections
```

The following is an example response for `ListGeofenceCollections`:

```
{
  "Entries": [
    {
      "CollectionName": "ExampleCollection",
      "CreateTime": 2020-09-30T22:59:34.142Z,
      "Description": "string",
      "UpdateTime": 2020-09-30T23:59:34.142Z
    },
    "NextToken": "1234-5678-9012"
  ]
}
```

CLI

Use the [list-geofence-collections](#) command.

The following example is an AWS CLI to get a list of geofence collections in the AWS account.

```
aws location list-geofence-collections
```

Get geofence collection details

You can get details about any geofence collection resource in your AWS account using the Amazon Location console, the AWS CLI, or the Amazon Location APIs:

Console

To view the details of a geofence collection using the Amazon Location console

1. Open the Amazon Location console at <https://console.aws.amazon.com/location/>.
2. Choose **Geofence collections** from the left navigation pane.
3. Under **My geofence collections**, select the name link of the target geofence collection.

API

Use the [DescribeGeofenceCollection](#) operation from the Amazon Location Geofences APIs.

The following example is an API request to get the geofence collection details for *ExampleCollection*.

```
GET /geofencing/v0/collections/ExampleCollection
```

The following is an example response for DescribeGeofenceCollection:

```
{
  "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

Use the [describe-geofence-collection](#) command.

The following example is an AWS CLI to get the geofence collection details for *ExampleCollection*.

```
aws location describe-geofence-collection \
  --collection-name "ExampleCollection"
```

Delete a geofence collection

You can delete a geofence collection from your AWS account using the Amazon Location console, the AWS CLI, or the Amazon Location APIs.

Console

To delete a geofence collection using the Amazon Location console

⚠ Warning

This operation deletes the resource permanently.

1. Open the Amazon Location console at <https://console.aws.amazon.com/location/>.
2. Choose **Geofence collections** from the left navigation pane.
3. Under **My geofence collection**, select the target geofence collection.
4. Choose **Delete geofence collection**.

API

Use the [DeleteGeofenceCollection](#) operation from the Amazon Location APIs.

The following example is an API request to delete the geofence collection

ExampleCollection.

```
DELETE /geofencing/v0/collections/ExampleCollection
```

The following is an example response for DeleteGeofenceCollection:

```
HTTP/1.1 200
```

CLI

Use the [delete-geofence-collection](#) command.

The following example is an AWS CLI command to delete the geofence collection

ExampleCollection.

```
aws location delete-geofence-collection \  
  --collection-name "ExampleCollection"
```

List stored geofences

You can list geofences stored in a specified geofence collection using the Amazon Location console, the AWS CLI, or the Amazon Location APIs.

Console

To view a list of geofences using the Amazon Location console

1. Open the Amazon Location console at <https://console.aws.amazon.com/location/>.
2. Choose **Geofence collections** from the left navigation pane.
3. Under **My geofence collection**, select the name link of the target geofence collection.
4. View geofences in the geofence collection under **Geofences**

API

Use the [ListGeofences](#) operation from the Amazon Location Geofences APIs.

The following example is an API request to get a list of geofences stored in the geofence collection *ExampleCollection*.

```
POST /geofencing/v0/collections/ExampleCollection/list-geofences
```

The following is an example response for ListGeofences:

```
{
  "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

Use the [list-geofences](#) command.

The following example is an AWS CLI to get a list of geofences stored in the geofence collection *ExampleCollection*.

```
aws location list-geofences \  
  --collection-name "ExampleCollection"
```

Get geofence details

You can get the details of a specific geofence, such as the create time, update time, geometry, and status, from a geofence collection using the Amazon Location console, AWS CLI, or the Amazon Location APIs.

Console

To view the status of a geofence using the Amazon Location console

1. Open the Amazon Location console at <https://console.aws.amazon.com/location/>.
2. Choose **Geofence collections** from the left navigation pane.
3. Under **My geofence collection**, select the name link of the target geofence collection.
4. Under **Geofences**, you'll be able to view the status of your geofences.

API

Use the [GetGeofence](#) operation from the Amazon Location Geofences APIs.

The following example is an API request to get the geofence details from a geofence collection *ExampleCollection*.

```
GET /geofencing/v0/collections/ExampleCollection/geofences/ExampleGeofence1
```

The following is an example response for GetGeofence:

```
{  
  "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

Use the [get-geofence](#) command.

The following example is an AWS CLI to get the geofence collection details for *ExampleCollection*.

```
aws location get-geofence \
  --collection-name "ExampleCollection" \
  --geofence-id "ExampleGeofence1"
```

Delete geofences

You can delete geofences from a geofence collection using the Amazon Location console, the AWS CLI, or the Amazon Location APIs.

Console

To delete a geofence using the Amazon Location console

Warning

This operation deletes the resource permanently.

1. Open the Amazon Location console at <https://console.aws.amazon.com/location/>.
2. Choose **Geofence collections** from the left navigation pane.
3. Under **My geofence collection**, select the name link of the target geofence collection.

4. Under **Geofences**, select the target geofence.
5. Choose **Delete geofence**.

API

Use the [BatchDeleteGeofence](#) operation from the Amazon Location Geofences APIs.

The following example is an API request to delete geofences from the geofence collection *ExampleCollection*.

```
POST /geofencing/v0/collections/ExampleCollection/delete-geofences
Content-type: application/json

{
  "GeofenceIds": [ "ExampleGeofence11" ]
}
```

The following is an example success response for [BatchDeleteGeofence](#).

```
HTTP/1.1 200
```

CLI

Use the [batch-delete-geofence](#) command.

The following example is an AWS CLI command to delete geofences from the geofence collection *ExampleCollection*.

```
aws location batch-delete-geofence \
  --collection-name "ExampleCollection" \
  --geofence-ids "ExampleGeofence11"
```

Manage costs and usage

As you continue learning about Amazon Location Geofences, it's important to understand how to manage service capacity, ensure you follow usage limits, and get the best results through quota and API optimizations. By applying best practices for performance and accuracy, you can tailor your application to handle place-related queries efficiently and maximize your API requests.

Topics

- [Geofences pricing](#)
- [Geofences quotas and usage](#)

Geofences pricing

For pricing information for tracking and geofencing APIs, see the [Amazon Location Service pricing page](#).

Position Evaluation

You can use `BatchEvaluateGeofences` to evaluate device positions against the geofence geometries from a given geofence collection. One request will evaluate up to 10 device positions against all geofences in a single geofence collection. Price is based on the number of API requests. Unit price per API call is based on the total monthly request volume. See the [Amazon Location Service pricing page](#) for details on unit price and volume tiers.

You can optimize your Position Evaluation cost by configuring the device position update frequency (also known as ping rate) from your tracking devices, and leveraging the filtering feature on Trackers to only evaluate relevant position updates.

Geofence Management and Storage

You can use `GetGeofence`, `PutGeofence`, `BatchPutGeofence`, `ListGeofences`, and `BatchDeleteGeofence` to manage your geofences in a geofence collection. The price is based on number of API requests.

The storage for geofences will be charged monthly (only for geofences you store for more than 1 month).

Geofence Event Forecast

You can use `ForecastGeofenceEvents` to forecast future geofence events that are likely to occur within a specified time horizon if a device continues moving at its current speed. The price is based on number of API requests.

Geofences quotas and usage

This topic provides a summary of rate limits and quotas for Amazon Location Service Geofences.

Note

If you require a higher quota, you can use the Service Quotas console to [request quota increases](#) for adjustable quotas. When requesting a quota increase, select the Region you require the quota increase in, since most quotas are specific to the AWS Region. You can request up to twice the default limit for each API.

For requests that exceed twice the default limit, your request will submit a support ticket. You can also connect to your premium support team. There are no direct charges for quota increase requests, but higher usage levels may lead to increased service costs based on the additional resources consumed. See [the section called “Manage quotas”](#) for more information.

Service Quotas are maximum number of resources you can have per AWS account and AWS Region. Amazon Location Service denies additional requests that exceed the service quota.

Resources

API name	Default	Max adjustable limit
Collection resources per account	1500	3000 If you need more than this, request quota increases or contact the support team.
Geofences per collection	50000	Contact the support team.

CRUD API**Note**

If you need a higher limit for any of these APIs, [request quota increases](#) or contact the support team.

API name	Default	Max adjustable limit
CreateGeofenceCollection	10	20
DeleteGeofenceCollection	10	20
DescribeGeofenceCollection	10	20
ListGeofenceCollections	10	20
UpdateGeofenceCollection	10	20

Data API

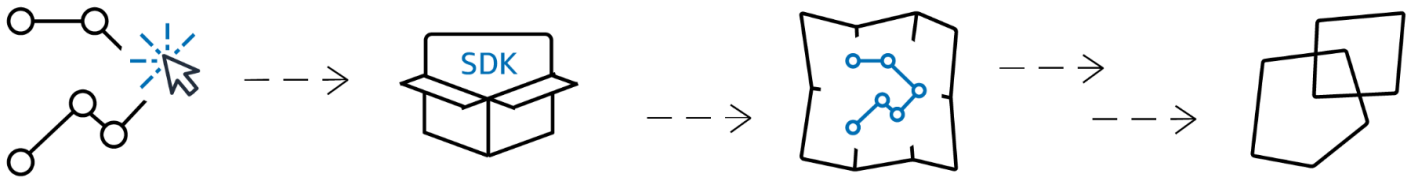
Note

If you need a higher limit for any of these APIs, [request quota increases](#) or contact the support team.

API name	Default	Max adjustable limit
BatchEvaluateGeofences	50	100
PutGeofence	50	100
BatchPutGeofence	50	100
ListGeofences	50	100
GetGeofence	50	100
BatchDeleteGeofence	50	100

Other usage limits

Amazon Location Service trackers



Note

Tracker storage is encrypted with AWS owned keys automatically. You can add another layer of encryption using KMS keys that you manage, to ensure that only you can access your data. For more information, see [the section called “Data at rest encryption”](#).

A tracker stores position updates for a collection of devices. The tracker can be used to query the devices' current location or location history. It stores the updates, but reduces storage space and visual noise by filtering the locations before storing them.

Each position update stored in your tracker resources can include a measure of position accuracy and up to 3 fields of metadata about the position or device that you want to store. The metadata is stored as key-value pairs, and can store information such as speed, direction, tire pressure, or engine temperature.

Tracker position filtering and query are useful on their own, but trackers are especially useful when paired with geofences. You can link trackers to one or more of your geofence collection resources, and position updates are evaluated automatically against the geofences in those collections. Proper use of filtering can greatly reduce the costs of your geofence evaluations, as well.

1. First, you create a tracker resource in your AWS account.
2. Next, decide how you send location updates to your tracker resources. Use [AWS SDKs](#) to integrate tracking capabilities into your mobile applications. Alternately, you can use MQTT by following step-by-step directions in [tracking using MQTT](#).
3. You can now use your tracker resource to record location history and visualize it on a map.
4. You can also link your tracker resource to one or more geofence collections so that every position update sent to your tracker resource is automatically evaluated against all the geofence

in all the linked geofence collections. You can link resource on the tracker resource details page of the Amazon Location console or by using the Amazon Location Trackers API.

5. You can then integrate monitoring using services such as Amazon CloudWatch and AWS CloudTrail. For more information see, [the section called “Monitor with Amazon CloudWatch”](#) and [the section called “Monitor and log with AWS CloudTrail”](#).

Features

- **Position filtering** – Trackers can automatically filter the positions that are sent to them. There are several reasons why you might want to filter out some of your device location updates. If you have a system that only sends reports every minute or so, you might want to filter devices by time, storing and evaluating positions only every 30 seconds. Even if you are monitoring more frequently, you might want to filter position updates to clean up the inherent noisiness associated with GPS hardware and position reporting. Their accuracy is not 100% perfect, so even a device that is stationary appears to be moving around slightly. At low speeds, this *jitter* causes visual clutter and can cause false entry and exit events if the device is near the edge of a geofence.

The position filtering works as position updates are received by a tracker, reducing visual noise in your device paths (jitter), reducing the number of false geofence entry and exit events, and helping manage costs by reducing the number of position updates stored and geofence evaluations triggered.


Trackers offer three position filtering options to help manage costs and reduce jitter in your location updates.

- **Accuracy-based** – *Use with any device that provides an accuracy measurement. Most GPS and mobile devices provide this information.*

The accuracy of each position measurement is affected by many environmental factors, including GPS satellite reception, landscape, and the proximity of WiFi and Bluetooth devices. Most devices, including most mobile devices, can provide an estimate of the accuracy of the measurement along with the measurement. With AccuracyBased filtering, Amazon Location ignores location updates if the device moved less than the measured accuracy.

For example, if two consecutive updates from a device have an accuracy range of 5 m and 10 m, Amazon Location ignores the second update if the device has moved less than 15 m. Amazon Location neither evaluates ignored updates against geofences, nor stores them.

When accuracy is not provided, it is treated as zero, and the measurement is considered perfectly accurate, and no filtering will be applied to the updates.

 **Note**

You can use accuracy-based filtering to remove all filtering. If you select accuracy-based filtering, but override all accuracy data to zero, or omit the accuracy entirely, then Amazon Location will not filter out any updates.

- **Distance-based** – *Use when your devices do not provide an accuracy measurement, but you still want to take advantage of filtering to reduce jitter and manage costs.*

DistanceBased filtering ignores location updates in which devices have moved less than 30 m (98.4 ft). When you use DistanceBased position filtering, Amazon Location neither evaluates these ignored updates against geofences nor stores the updates.

The accuracy of most mobile devices, including the average accuracy of iOS and Android devices, is within 15 m. In most applications, DistanceBased filtering can reduce the effect of location inaccuracies when displaying device trajectory on a map, and the bouncing effect of multiple consecutive entry and exit events when devices are near the border of a geofence. It can also help reduce the cost of your application, by making fewer calls to evaluate against linked geofences or retrieve device positions.

Distance-based filtering is useful if you want to filter, but your device doesn't provide accuracy measurements, or you want to filter out a larger number of updates than with accuracy-based.

- **Time-based** – (default) *Use when your devices send position updates very frequently (more than once every 30 seconds), and you want to achieve near real-time geofence evaluations without storing every update.*

In TimeBased filtering, every location update is evaluated against linked geofence collections, but not every location update is stored. If your update frequency is more often than 30 seconds, only one update per 30 seconds is stored for each unique device ID.

Time-based filtering is particularly useful when you want to store fewer positions, but want every position update to be evaluated against the associated geofence collections.

Note

Be mindful of the costs of your tracking application when deciding your filtering method and the frequency of position updates. You are billed for every location update and once for evaluating the position update against each linked geofence collection. For example, when using time-based filtering, if your tracker is linked to two geofence collections, every position update will count as one location update request and two geofence collection evaluations. If you are reporting position updates every 5 seconds for your devices and using time-based filtering, you will be billed for 720 location updates and 1,440 geofence evaluations per hour for each device.

Use cases for Amazon Location Service trackers

The following are a few common uses for Amazon Location Service trackers.

Use trackers with geofences

Trackers provide additional functionality when paired with geofences. You associate a tracker with a geofence collection, either through the Amazon Location console or the API, to automatically evaluate tracker locations. Each time the tracker receives an updated location, that location will be evaluated against each geofence in the collection, and the appropriate ENTER and EXIT events are generated in Amazon EventBridge. You can also apply filtering to the tracker, and, depending on the filtering, you can reduce the costs for geofence evaluations by only evaluating meaningful location updates.

If you associate the tracker with a geofence collection after it has already received some position updates, the first position update after association is treated as an initial update for the geofence evaluations. If it is within a geofence, you will receive an ENTER event. If it is not within any geofences you will not receive an EXIT event, regardless of the previous state.

Improve field service operations

Keep a pulse on your mobile workforce with real-time tracking. Set geofences around customer sites and service areas to receive alerts when staff arrive and depart. Use location data to optimize scheduling, dispatch the nearest available technician, and reduce response times. Empower your field teams (such as a your plumbing or HVAC repair business) to work more efficiently, while enhancing the customer experience.

Monitor and control critical assets

Utilize Amazon Location Service to track the real-time location and status of your valuable equipment, inventory, and other mobile assets. Set up geofences to receive alerts on unauthorized movements or removals, enhancing security and compliance. Use this location visibility to improve asset utilization, optimize maintenance schedules, and ensure your critical resources are accounted for at all times. Always monitor your heavy machinery, IT hardware, or retail inventory with precision, reduce losses, and make more informed operational decisions.

Enhance supply chain visibility

Leverage Amazon Location Service to track shipments and deliveries across your entire supply chain. Define geofences around distribution centers, stores, and other key facilities to monitor the movement of inventory and assets. Use real-time location data to improve inventory management, optimize logistics planning, and deliver a superior customer experience. Gain end-to-end visibility into your supply chain operations, identify bottlenecks, and make data-driven decisions that drive efficiency and responsiveness.

Location-based marketing

Unlock the power of location data to supercharge your geomarketing efforts. Use Amazon Location Service to set virtual boundaries around competitor locations, events, and high-traffic areas. Trigger personalized ads, offers, and notifications when customers enter these geofenced zones. Analyze foot traffic patterns to optimize ad placements and uncover prime sites for new business locations. Monitor customer movements within your own geofenced spaces to gain deeper insights on browsing behaviors and path-to-purchase. Combine real-time location tracking with precision geofencing to deliver hyper-targeted, contextual engagement that drives sales and loyalty in the physical world.

Tracker concepts

This section details common trackers concepts.

Common Amazon Location Service trackers terminology

Tracker resource

An AWS resource that receives location updates from devices. The tracker resource provides support for location queries, such as current and historic device location. Linking a tracker

resource to a geofence collection evaluates location updates against all geofences in the linked geofence collection automatically.

Position data tracked

A tracker resource stores information about your devices over time. The information includes a series of position updates, where each update includes location, time, and optional metadata. The metadata can include a position's accuracy, and up to three key-value pairs to help you track key information about each position, such as speed, direction, tire pressure, remaining fuel, or engine temperature of the vehicle you are tracking. Trackers maintain device location history for 30 days.

Position filtering

Position filtering can help you control costs and improve the quality of your tracking application by filtering out position updates that don't provide valuable information before the updates are stored or evaluated against geofences.

You can choose `AccuracyBased`, `DistanceBased`, or `TimeBased` filtering. By default, position filtering is set to `TimeBased`.

You can configure position filtering when you create or update tracker resources.

RFC 3339 timestamp format

Amazon Location Service trackers use the [RFC 3339](#) format, which follows the [International Organization for Standardization \(ISO\) 8601](#) format for dates and time.

The format is "YYYY-MM-DDThh:mm:ss.sssZ+00:00":

- YYYY-MM-DD — Represents the date format.
- T — Indicates that the time values will follow.
- hh:mm:ss.sss — Represents the time in 24-hour format.
- Z — Indicates that the time zone used is UTC, which can be followed with deviations from the UTC time zone.
- +00:00 — Optionally indicate deviations from the UTC time zone. For example, +01:00 indicates UTC + 1 hour.

Example

For July 2, 2020, at 12:15:20 in the afternoon, with an adjustment of an additional 1 hour to the UTC time zone.

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

Get started with Amazon Location Service trackers

This section provides a comprehensive guide to creating and using trackers with Amazon Location. Trackers allow you to store, process, and evaluate device positions while filtering location updates to reduce noise and manage costs. With advanced position filtering options, support for linked geofence collections, and integration with AWS services like EventBridge and IoT Core, trackers enable accurate real-time tracking and geofencing applications tailored to your specific needs.

Topics

- [Create a tracker](#)
- [Authenticating your requests](#)
- [Update your tracker with a device position](#)
- [Get a device's location history from a tracker](#)
- [List your device positions](#)

Create a tracker

Create a tracker resource to store and process position updates from your devices. You can use the Amazon Location Service console, the AWS CLI, or the Amazon Location APIs.

Each position update stored in your tracker resources can include a measure of position accuracy, and up to three fields of metadata about the position or device that you want to store. The metadata is stored as key-value pairs, and can store information such as speed, direction, tire pressure, or engine temperature.


Trackers filter position updates as they are received. This reduces visual noise in your device paths (called *jitter*), and reduces the number of false geofence entry and exit events. This also helps manage costs by reducing the number of geofence evaluations initiated.

Trackers offer three position filtering options to help manage costs and reduce jitter in your location updates.

- **Accuracy-based** – *Use with any device that provides an accuracy measurement. Most mobile devices provide this information.* The accuracy of each position measurement is affected by many

environmental factors, including GPS satellite reception, landscape, and the proximity of Wi-Fi and Bluetooth devices. Most devices, including most mobile devices, can provide an estimate of the accuracy of the measurement along with the measurement. With `AccuracyBased` filtering, Amazon Location ignores location updates if the device moved less than the measured accuracy. For example, if two consecutive updates from a device have an accuracy range of 5 m and 10 m, Amazon Location ignores the second update if the device has moved less than 15 m. Amazon Location neither evaluates ignored updates against geofences, nor stores them.

When accuracy is not provided, it is treated as zero, and the measurement is considered perfectly accurate.

 **Note**

You can also use accuracy-based filtering to remove all filtering. If you select accuracy-based filtering, but override all accuracy data to zero, or omit the accuracy entirely, then Amazon Location will not filter out any updates.

- **Distance-based** – *Use when your devices do not provide an accuracy measurement, but you still want to take advantage of filtering to reduce jitter and manage costs.* `DistanceBased` filtering ignores location updates in which devices have moved less than 30 m (98.4 ft). When you use `DistanceBased` position filtering, Amazon Location neither evaluates these ignored updates against geofences nor stores the updates.

The accuracy of most mobile devices, including the average accuracy of iOS and Android devices, is within 15 m. In most applications, `DistanceBased` filtering can reduce the effect of location inaccuracies when displaying device trajectory on a map, and the bouncing effect of multiple consecutive entry and exit events when devices are near the border of a geofence. It can also help reduce the cost of your application, by making fewer calls to evaluate against linked geofences or retrieve device positions.

- **Time-based** – (default) *Use when your devices send position updates very frequently (more than once every 30 seconds), and you want to achieve near real-time geofence evaluations without storing every update.* In `TimeBased` filtering, every location update is evaluated against linked geofence collections, but not every location update is stored. If your update frequency is more often than 30 seconds, only one update per 30 seconds is stored for each unique device ID.

Note

Be mindful of the costs of your tracking application when deciding your filtering method and the frequency of position updates. You are billed for every location update and once for evaluating the position update against each linked geofence collection. For example, when using time-based filtering, if your tracker is linked to two geofence collections, every position update will count as one location update request and two geofence collection evaluations. If you are reporting position updates every 5 seconds for your devices and using time-based filtering, you will be billed for 720 location updates and 1,440 geofence evaluations per hour for each device.

Your bill is not affected by the number of geofences in each collection. Since each geofence collection may contain up to 50,000 geofences, you may want to combine your geofences into fewer collections, where possible, to reduce your cost of geofence evaluations.

By default, you will get EventBridge events each time a tracked device enters or exits a linked geofence. For more information, see [Link a tracker to a geofence collection](#).

You can enable events for all filtered position updates for a tracker resource. For more information, see [the section called “Enable update events for a tracker”](#).

Note

If you wish to encrypt your data using your own AWS KMS customer managed key, then the Bounding Polygon Queries feature will be disabled by default. This is because by using this Bounding Polygon Queries feature, a representation of your device positions will not be encrypted using your AWS KMS managed key. However, the exact device position is still encrypted using your managed key.


You can choose to opt-in to the Bounding Polygon Queries feature by setting the `KmsKeyEnableGeospatialQueries` parameter to true when creating or updating a Tracker.

Console

To create a tracker using the Amazon Location console

1. Open the Amazon Location Service console at <https://console.aws.amazon.com/location/>.

2. In the left navigation pane, choose **Trackers**.
3. Choose **Create tracker**.
4. Fill the following fields:
 - **Name** – Enter a unique name. For example, *ExampleTracker*. Maximum 100 characters. Valid entries include alphanumeric characters, hyphens, periods, and underscores.
 - **Description** – Enter an optional description.
5. Under **Position filtering**, choose the option that best fits how you intend to use your tracker resource. If you do not set **Position filtering**, the default setting is TimeBased. For more information, see [Trackers](#) in this guide, and [PositionFiltering](#) in the Amazon Location Service Trackers API Reference.
6. (Optional) Under **Tags**, enter a tag **Key** and **Value**. This adds a tag your new geofence collection. For more information, see [the section called “How to use tags”](#).
7. (Optional) Under **Customer managed key encryption**, you can choose to **Add a customer managed key**. This adds a symmetric customer managed key that you create, own, and manage over the default AWS owned encryption. For more information, see [Encrypting data at rest](#).
8. (Optional) Under **KmsKeyEnableGeospatialQueries**, you can choose to enable **Geospatial Queries**. This allows you use the Bounding Polygon Queries feature, while encrypting your data using a customer AWS KMS managed key.

 **Note**

When you use the Bounding Polygon Queries feature a representation of your device positions is not be encrypted using your AWS KMS managed key. However, the exact device position is still encrypted using your managed key.

9. (Optional) Under **EventBridge configuration**, you can choose to enable EventBridge events for filtered position updates. This will send an event each time a position update for a device in this tracker meets the position filtering evaluation.
10. Choose **Create tracker**.

API

To create a tracker by using the Amazon Location APIs

Use the [CreateTracker](#) operation from the Amazon Location Trackers APIs.

The following example uses an API request to create a tracker called *ExampleTracker*. The tracker resource is associated with a [customer managed AWS KMS key to encrypt customer data](#), and does not [enable position updates in EventBridge](#).

```
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"
  }
}
```

Create a tracker with KmsKeyEnableGeospatialQueries enabled

The following example has the parameter `KmsKeyEnableGeospatialQueries` set to true. This allows you use the Bounding Polygon Queries feature, while encrypting your data using a customer AWS KMS managed key.

For information on using the Bounding Polygon Queries feature, see [the section called “List your device positions”](#)

Note

When you use the Bounding Polygon Queries feature a representation of your device positions is not be encrypted using your AWS KMS managed key. However, the exact device position is still encrypted using your managed key.

```
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

To create a tracker using AWS CLI commands

Use the [create-tracker](#) command.

The following example uses the AWS CLI to create a tracker called *ExampleTracker*. The tracker resource is associated with a [customer managed AWS KMS key to encrypt customer data](#), and does not [enable position updates in EventBridge](#).

```
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"
```

Create a tracker with KmsKeyEnableGeospatialQueries enabled

The following example has the parameter `KmsKeyEnableGeospatialQueries` set to true. This allows you use the Bounding Polygon Queries feature, while encrypting your data using a customer AWS KMS managed key.

For information on using the Bounding Polygon Queries feature, see [the section called "List your device positions"](#)

Note

When you use the Bounding Polygon Queries feature a representation of your device positions is not be encrypted using your AWS KMS managed key. However, the exact device position is still encrypted using your managed key.

```
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

Billing depends on your usage. You may incur fees for the use of other AWS services. For more information, see [Amazon Location Service pricing](#).

You can edit the **Description**, **Position filtering**, and **EventBridge configuration** after the tracker is created by choosing **Edit tracker**.

Authenticating your requests

Once you create a tracker resource and you're ready to begin evaluating device positions against geofences, choose how you would authenticate your requests:

- To explore ways you can access the services, see [Authentication](#).
- If you want to publish device positions with unauthenticated requests, you may want to use Amazon Cognito.

Example

The following example shows using an Amazon Cognito identity pool for authorization, using [AWS JavaScript SDK v3](#), and the Amazon Location [the section called "Web"](#).

```
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);
```

Update your tracker with a device position

To track your devices, you can post device position updates to your tracker. You can later retrieve these device positions or the device position history from your tracker resource.

Each position update must include the device ID, a timestamp, and a position. You may optionally include other metadata, including accuracy and up to 3 key-value pairs for your own use.

If your tracker is linked to one or more geofence collections, updates will be evaluated against those geofences (following the filtering rules that you specified for the tracker). If a device breaches a geofenced area (by moving from inside the area to outside, or vice versa), you will receive events in EventBridge. These ENTER or EXIT events include the position update details, including the device ID, the timestamp, and any associated metadata.

Note

For more information about position filtering, see [the section called "Create a tracker"](#). For more information about geofence events, see [the section called "React to events with EventBridge"](#).

Use either of these methods to send device updates:

- [Send MQTT updates](#) to an AWS IoT Core resource and link it to your tracker resource.
- Send location updates using the Amazon Location Trackers API, by using the AWS CLI, or the Amazon Location APIs. You can use the [AWS SDKs](#) to call the APIs from your iOS or Android application.

API

To send a position update using the Amazon Location APIs

Use the [BatchUpdateDevicePosition](#) operation from the Amazon Location Trackers APIs.

The following example uses an API request to post a device position update for *ExampleDevice* to a tracker *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

To send a position update using AWS CLI commands

Use the [batch-update-device-position](#) command.

The following example uses an AWS CLI to post a device position update for *ExampleDevice-1* and *ExampleDevice-2* to a tracker *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"}}]'
```

Get a device's location history from a tracker

Your Amazon Location tracker resource maintains the location history of all your tracked devices for a period of 30 days. You can retrieve device location history, including all associated metadata, from your tracker resource. The following examples use the AWS CLI, or the Amazon Location APIs.

API

To get the device location history from a tracker using the Amazon Location APIs

Use the [GetDevicePositionHistory](#) operation from the Amazon Location Trackers APIs.

The following example uses an API URI request to get the device location history of *ExampleDevice* from a tracker called *ExampleTracker* starting from 19:05:07 (inclusive) and ends at 19:20:07 (exclusive) on 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

To get the device location history from a tracker using AWS CLI commands

Use the [get-device-position-history](#) command.

The following example uses an AWS CLI to get the device location history of *ExampleDevice* from a tracker called *ExampleTracker* starting from 19:05:07 (inclusive) and ends at 19:20:07 (exclusive) on 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"
```

List your device positions

You can view a list device positions for a tracker using the AWS CLI, or the Amazon Location APIs, with the ListDevicePositions API. When you call the ListDevicePositions API, a list of the latest positions for all devices associated with a given tracker is returned. By default this API returns 100 of the latest device positions per page of results for a given tracker. To only return devices within a specific region use the FilterGeometry parameter to create a Bounding Polygon Query. This way when you call ListDevicePositions, only devices inside the polygon will be returned.

Note

If you wish to encrypt your data using your own AWS KMS customer managed key, then the Bounding Polygon Queries feature will be disabled by default. This is because by using this feature, a representation of your device positions will not be encrypted using your

AWS KMS managed key. The exact device position, however; is still encrypted using your managed key.

You can choose to opt-in to the Bounding Polygon Queries feature. This is done by setting the `KmsKeyEnableGeospatialQueries` parameter to true when creating or updating a Tracker.

API

Use the [ListDevicePositions](#) operation from the Amazon Location Trackers APIs.

The following example is an API request to get a list of device positions in polygonal area, using the optional parameter [FilterGeometry](#). The example returns 3 device locations present in the area defined by the Polygon array.

```
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"
}
```

The following is an example response for [ListDevicePositions](#):

```
{
  "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

Use the [list-trackers](#) command.

The following example is an AWS CLI to get a list of devices in a polygonal area.

```
aws location list-device-positions TODO: add arguments add props for filter geo
```

How to work with Amazon Location Service trackers

This section provides instructions for working with Amazon Location trackers. Learn how to verify device positions, link trackers to geofence collections, and track locations using AWS IoT and MQTT. Additionally, learn how to manage your trackers effectively to support your location-based applications and ensure accurate, real-time tracking.

Topics

- [Verify device positions](#)
- [Link a tracker to a geofence collection](#)
- [Track using AWS IoT and MQTT with Amazon Location Service](#)
- [Manage your Amazon Location Service tracker](#)

Verify device positions

To check the integrity of a device position use the [VerifyDevicePosition](#) API. This API returns information about the integrity of the device's position, by evaluating properties such as the device's cell signal, Wi-Fi access point, Ipv4 address, and if a proxy is in use.

Prerequisites

Before being able to use the listed APIs for device verification, make sure you have the following prerequisite:

- You have created a tracker for the device or devices you want to check. For more information, see [Get started with Amazon Location Service trackers](#).

The following example shows a request for the Amazon Location [VerifyDevicePosition](#) API.

API

To verify device positions using the Amazon Location APIs

Use the [VerifyDevicePosition](#) operation from the Amazon Location Tracking APIs.

The following example shows an API request to evaluate the integrity of the position of a device. Replace these values with your own device 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

The **Integrity SDK** provides enhanced features related to device verification, and it is available for use by request. To get access to the SDK, contact [Sales Support](#).

Link a tracker to a geofence collection

Now that you have a geofence collection and a tracker, you can link them together so that location updates are automatically evaluated against all of your geofences. If you don't want to evaluate all location updates, or alternatively, if you aren't storing some of your locations in a tracker resource, you can [evaluate device positions against geofences](#) on demand.

When device positions are evaluated against geofences, events are generated. You can set an action to these events. For more information about actions that you can set for geofence events, see [Reacting to Amazon Location Service events with Amazon EventBridge](#).

An Amazon Location event includes the attributes of the device position update that generates it and some attributes of the geofence that is entered or exited. For more information about the data included in a geofence event, see [the section called "Event examples"](#).

The following examples link a tracker resource to a geofence collection using the console, the AWS CLI, or the Amazon Location APIs.

Console

To link a tracker resource to a geofence collection using the Amazon Location Service console

1. Open the Amazon Location Service console at <https://console.aws.amazon.com/location/>.
2. In the left navigation pane, choose **Trackers**.
3. Under **Device trackers**, select the name link of the target tracker.
4. Under **Linked Geofence Collections**, choose **Link Geofence Collection**.
5. In the **Linked Geofence Collection** window, select a geofence collection from the dropdown menu.
6. Choose **Link**.

After you link the tracker resource, it will be assigned an **Active** status.

API

To link a tracker resource to a geofence collection using the Amazon Location APIs

Use the [AssociateTrackerConsumer](#) operation from the Amazon Location Trackers APIs.

The following example uses an API request that associates *ExampleTracker* with a geofence collection using its [Amazon Resource Name](#) (ARN).

```
POST /tracking/v0/trackers/ExampleTracker/consumers
Content-type: application/json

{
  "ConsumerArn": "arn:aws:geo:us-west-2:123456789012:geofence-
collection/ExampleGeofenceCollection"
}
```

AWS CLI

To link a tracker resource to a geofence collection using AWS CLI commands

Use the [associate-tracker-consumer](#) command.

The following example uses an AWS CLI to create a geofence collection called *ExampleGeofenceCollection*.

```
aws location \
  associate-tracker-consumer \
    --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-
collection/ExampleGeofenceCollection" \
    --tracker-name "ExampleTracker"
```

Track using AWS IoT and MQTT with Amazon Location Service

[MQTT](#) is a lightweight and widely adopted messaging protocol designed for constrained devices. AWS IoT Core supports device connections that use the MQTT protocol and MQTT over WebSocket Secure (WSS) protocol.

[AWS IoT Core](#) connects devices to AWS and enables you to send and receive messages between them. The AWS IoT Core rules engine stores queries about your devices' message topics and enables you to define actions for sending messages to other AWS services, such as Amazon Location Service. Devices that are aware of their location as coordinates can have their locations forwarded to Amazon Location through the rules engine.

Note

Devices may know their own position, for example via built-in GPS. AWS IoT also has support for third party device location tracking. For more information, see [AWS IoT Core Device Location](#) in the *AWS IoT Core Developer Guide*.

The following walkthrough describes tracking using AWS IoT Core rules. You can also send the device information to your own AWS Lambda function, if you need to process it before sending to Amazon Location. For more details about using Lambda to process your device locations, see [Use AWS Lambda with MQTT](#).

Topics

- [Prerequisite](#)
- [Create an AWS IoT Core rule](#)
- [Test your AWS IoT Core rule in the console](#)

- [Use AWS Lambda with MQTT](#)

Prerequisite

Before you can begin tracking, you must complete the following prerequisites:

- [Create a tracker resource](#) that you will send the device location data to.
- [Create an IAM role](#) for granting AWS IoT Core access to your tracker.

When following those steps, use the following policy to give access to your tracker:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "WriteDevicePosition",
      "Effect": "Allow",
      "Action": "geo:BatchUpdateDevicePosition",
      "Resource": "arn:aws:geo:*:*:tracker/*"
    }
  ]
}
```

Create an AWS IoT Core rule

Next, create an AWS IoT Core rule to forward your devices' positional telemetry to Amazon Location Service. For more information about creating rules, see the following topics in the *AWS IoT Core Developer Guide*:

- [Creating an AWS IoT rule](#) for information about creating a new rule.
- [Location action](#) for information specific to creating a rule for publishing to Amazon Location

Test your AWS IoT Core rule in the console

If no devices are currently publishing telemetry that includes location, you can test your rule using the AWS IoT Core console. The console has a test client where you can publish a sample message to verify the results of the solution.

1. Sign in to the AWS IoT Core console at <https://console.aws.amazon.com/iot/>.

2. In the left navigation, expand **Test**, and choose **MQTT test client**.
3. Under **Publish to a topic**, set the **Topic name** to *iot/topic* (or the name of the topic that you set up in your AWS IoT Core rule, if different), and provide the following for the **Message payload**. Replace the timestamp *1604940328* with a valid timestamp within the last 30 days (any timestamps older than 30 days are ignored by Amazon Location Service trackers).

```
{
  "payload": {
    "deviceid": "thing123",
    "timestamp": 1604940328,
    "location": { "lat": 49.2819, "long": -123.1187 },
    "accuracy": { "Horizontal": 20.5 },
    "positionProperties": { "field1": "value1", "field2": "value2" }
  }
}
```

4. Choose **Publish** to topic to send the test message.
5. To validate that the message was received by Amazon Location Service, use the following AWS CLI command. If you modified it during setup, replace the tracker name with the one that you used.

```
aws location batch-get-device-position --tracker-name MyTracker --device-ids
thing123
```

Use AWS Lambda with MQTT

While using AWS Lambda is no longer required when sending device location data to Amazon Location for tracking, you may still want to use Lambda in some cases. For example, if you wish to process your device location data yourself, before sending it on to Amazon Location. The following topics describe how to use Lambda to process messages before sending them to your tracker. For more information about this pattern, see the [reference architecture](#).

Topics

- [Prerequisite](#)
- [Create a Lambda function](#)
- [Create an AWS IoT Core rule](#)
- [Test your AWS IoT Core rule in the console](#)

Prerequisite

Before you can begin tracking, you must [create a tracker resource](#). To create a tracker resource, you can use the Amazon Location console, the AWS CLI, or the Amazon Location APIs.

The following example uses the Amazon Location Service console to create the tracker resource:

1. Open the Amazon Location Service console at <https://console.aws.amazon.com/location/>.
2. In the left navigation pane, choose **Trackers**.
3. Choose **Create tracker**.
4. Fill out the following boxes:
 - **Name** – Enter a unique name that has a maximum of 100 characters. Valid entries include alphanumeric characters, hyphens, and underscores. For example, *MyTracker*.
 - **Description** – Enter an optional description. For example, *Tracker for storing AWS IoT Core device positions*.
 - **Position filtering** – Select the filtering that you want to use for position updates. For example, **Accuracy-based filtering**.
5. Choose **Create tracker**.

Create a Lambda function

To create a connection between AWS IoT Core and Amazon Location Service, you need an AWS Lambda function to process messages forwarded by AWS IoT Core. This function will extract any positional data, format it for Amazon Location Service, and submit it through the Amazon Location Tracker API. You can create this function through the AWS Lambda console, or you can use the AWS Command Line Interface (AWS CLI) or the AWS Lambda APIs.

To create a Lambda function that publishes position updates to Amazon Location using the console:

1. Open the AWS Lambda console at <https://console.aws.amazon.com/lambda/>.
2. From the left navigation, choose **Functions**.
3. Choose **Create Function**, and make sure that **Author from scratch** is selected.
4. Fill out the following boxes:

- **Function name** – Enter a unique name for your function. Valid entries include alphanumeric characters, hyphens, and underscores with no spaces. For example, *MyLambda*.
 - **Runtime** – Choose *Python 3.8*.
5. Choose **Create function**.
 6. Choose the **Code** tab to open the editor.
 7. Overwrite the placeholder code in `lambda_function.py` with the following, replacing the value assigned to `TRACKER_NAME` with the name of the tracker that you created as a [prerequisite](#).

```
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. Choose **Deploy** to save the updated function.
9. Choose the **Configuration** tab.
10. In the **Permissions** section, choose the hyperlinked Role name to grant Amazon Location Service permissions to your Lambda function.
11. From your role's **Summary** page, choose **Add permissions**, and then from the dropdown list, select **Create inline policy**.
12. Choose the **JSON** tab, and overwrite the policy with the following document. This allows your Lambda function to update device positions managed by all tracker resources across all Regions.

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "WriteDevicePosition",
    "Effect": "Allow",
    "Action": "geo:BatchUpdateDevicePosition",
    "Resource": "arn:aws:geo:*:*:tracker/*"
  }
]
}
```

13. Choose **Review policy**.
14. Enter a policy name. For example, *AmazonLocationTrackerWriteOnly*.
15. Choose **Create policy**.

You can modify this function code, as necessary, to adapt to your own device message schema.

Create an AWS IoT Core rule

Next, create an AWS IoT Core rule to forward your devices' positional telemetry to the AWS Lambda function for transformation and publication to Amazon Location Service. The example rule provided assumes that any necessary transformation of device payloads is handled by your Lambda function. You can create this rule through the AWS IoT Core console, the AWS Command Line Interface (AWS CLI), or the AWS IoT Core APIs.

Note

While the AWS IoT console handles the permission necessary to allow AWS IoT Core to invoke your Lambda function, if you are creating your rule from the AWS CLI or SDK, you must [configure a policy to grant permission to AWS IoT](#).

To create an AWS IoT Core using the console

1. Sign in to the AWS IoT Core console at <https://console.aws.amazon.com/iot/>.
2. In the left navigation, expand **Act**, and choose **Rules**.
3. Choose **Create a rule** to start the new rule wizard.
4. Enter a name and description for your rule.

5. For the **Rule query statement**, update the FROM attribute to refer to a topic where at least one device is publishing telemetry that includes location. If you are testing the solution, no modification is needed.

```
SELECT * FROM 'iot/topic'
```

6. Under **Set one or more actions**, choose **Add action**.
7. Select **Send a message to a lambda function**.
8. Choose **Configure action**.
9. Find and select your Lambda function from the list.
10. Choose **Add action**.
11. Choose **Create rule**.

Test your AWS IoT Core rule in the console

If no devices are currently publishing telemetry that includes location, you can test your rule and this solution using the AWS IoT Core console. The console has a test client where you can publish a sample message to verify the results of the solution.

1. Sign in to the AWS IoT Core console at <https://console.aws.amazon.com/iot/>.
2. In the left navigation, expand **Test**, and choose **MQTT test client**.
3. Under **Publish to a topic**, set the **Topic name** to *iot/topic* (or the name of the topic that you set up in your AWS IoT Core rule, if different), and provide the following for the **Message payload**. Replace the timestamp *1604940328* with a valid timestamp within the last 30 days (any timestamps older than 30 days are ignored).

```
{
  "payload": {
    "deviceid": "thing123",
    "timestamp": 1604940328,
    "location": { "lat": 49.2819, "long": -123.1187 },
    "accuracy": { "Horizontal": 20.5 },
    "positionProperties": { "field1": "value1", "field2": "value2" }
  }
}
```

4. Choose **Publish to topic** to send the test message.

5. To validate that the message was received by Amazon Location Service, use the following AWS CLI command. If you modified them during setup, replace the tracker name and device id with the ones that you used.

```
aws location batch-get-device-position --tracker-name MyTracker --device-ids
thing123
```

Manage your Amazon Location Service tracker

You can manage your trackers using the Amazon Location console, the AWS CLI, or the Amazon Location APIs.

List your trackers

You can view your trackers list using the Amazon Location console, the AWS CLI, or the Amazon Location APIs:

Console

To view a list of existing trackers using the Amazon Location console

1. Open the Amazon Location console at <https://console.aws.amazon.com/location/>.
2. Choose **Trackers** from the left navigation.
3. View a list of your tracker resources under **My trackers**.

API

Use the [ListTrackers](#) operation from the Amazon Location Trackers APIs.

The following example is an API request to get a list of trackers in your AWS account.

```
POST /tracking/v0/list-trackers
```

The following is an example response for [ListTrackers](#):

```
{
  "Entries": [
    {
```

```
    "CreateTime": 2020-10-02T19:09:07.327Z,  
    "Description": "string",  
    "TrackerName": "ExampleTracker",  
    "UpdateTime": 2020-10-02T19:10:07.327Z  
  }  
],  
"NextToken": "1234-5678-9012"  
}
```

CLI

Use the [list-trackers](#) command.

The following example is an AWS CLI to get a list of trackers in your AWS account.

```
aws location list-trackers
```

Disconnecting a tracker from a geofence collection

You can disconnect a tracker from a geofence collection using the Amazon Location console, the AWS CLI, or the Amazon Location APIs:

Console

To disassociate a tracker from an associated geofence collection using the Amazon Location console

1. Open the Amazon Location console at <https://console.aws.amazon.com/location/>.
2. Choose **Trackers** from the left navigation pane.
3. Under **My trackers**, select the name link of the target tracker.
4. Under **Linked Geofence Collections**, select a geofence collection with a **Linked** status.
5. Choose **Unlink**.

API

Use the [DisassociateTrackerConsumer](#) operation from the Amazon Location Trackers APIs.

The following example is an API request to disassociate a tracker from an associated geofence collection.


```
DELETE /tracking/v0/trackers/ExampleTracker/consumers/arn:aws:geo:us-west-2:123456789012:geofence-collection/ExampleCollection
```

The following is an example response for [DisassociateTrackerConsumer](#):

```
HTTP/1.1 200
```

CLI

Use the [disassociate-tracker-consumer](#) command.

The following example is an AWS CLI command to disassociate a tracker from an associated geofence collection.

```
aws location disassociate-tracker-consumer \  
  --consumer-arn "arn:aws:geo:us-west-2:123456789012:geofence-collection/  
ExampleCollection" \  
  --tracker-name "ExampleTracker"
```

Get tracker details

You can get details about any tracker in your AWS account by using the Amazon Location console, the AWS CLI, or the Amazon Location APIs.

Console

To view tracker details by using the Amazon Location console

1. Open the Amazon Location console at <https://console.aws.amazon.com/location/>.
2. Choose **Trackers** from the left navigation.
3. Under **My trackers**, select the name link of the target tracker.
4. View the tracker details under **Information**.

API

Use the [DescribeTracker](#) operation from the Amazon Location Tracker APIs.

The following example is an API request to get the tracker details for *ExampleTracker*.

```
GET /tracking/v0/trackers/ExampleTracker
```

The following is an example response for [DescribeTracker](#):

```
{
  "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

Use the [describe-tracker](#) command.

The following example is an AWS CLI command to get tracker details for *ExampleTracker*.

```
aws location describe-tracker \
  --tracker-name "ExampleTracker"
```

Delete a tracker

You can delete a tracker from your AWS account using the Amazon Location console, the AWS CLI, or the Amazon Location APIs:

Console

To delete an existing map resource using the Amazon Location console

⚠ Warning

This operation deletes the resource permanently. If the tracker resource is in use, you may encounter an error. Make sure that the target resource isn't a dependency for your applications.

1. Open the Amazon Location console at <https://console.aws.amazon.com/location/>.
2. Choose **Trackers** from the left navigation pane.
3. Under **My trackers**, select the target tracker.
4. Choose **Delete tracker**.

API

Use the [DeleteTracker](#) operation from the Amazon Location Tracker APIs.

The following example is an API request to delete the tracker *ExampleTracker*.

```
DELETE /tracking/v0/trackers/ExampleTracker
```

The following is an example response for [DeleteTracker](#):

```
HTTP/1.1 200
```

CLI

Use the [delete-tracker](#) command.

The following example is an AWS CLI command to delete the tracker *ExampleTracker*.

```
aws location delete-tracker \  
  --tracker-name "ExampleTracker"
```

Manage costs and usage

As you continue learning about Amazon Location trackers, it's important to understand how to manage service capacity, ensure you follow usage limits, and get the best results through quota

and API optimizations. By applying best practices for performance and accuracy, you can tailor your application to handle place-related queries efficiently and maximize your API requests.

Topics

- [Trackers pricing](#)
- [Trackers quota and usage](#)

Trackers pricing

For pricing information for tracking and geofencing APIs, see the [Amazon Location Service pricing page](#).

Position Written

You can use `BatchUpdateDevicePosition` to upload position update data for one or more devices to a tracker resource (up to 10 devices per batch). Price is based on number of API requests. Unit price per API call is based on the total monthly request volume. See the [Amazon Location Service pricing page](#) for details on unit price and volume tiers.

You can optimize your Position Written cost by configuring the device position update frequency (also known as ping rate) from your tracking devices, and optionally use a local filter to only upload relevant device position updates to Amazon Location Service.

Position Read

You can use `BatchGetDevicePosition` to lists the latest device positions for requested devices, up to 100 devices per request. You can also use `GetDevicePosition` to retrieve a device's most recent position according to its sample time.

Price is based on the number of API requests.

Position Delete

You can use `BatchDeleteDevicePositionHistory` to delete the position history of one or more devices from a tracker resource, up to 100 devices per request.

Price is based on the number of API requests.

Position Integrity Evaluation

You can use `VerifyDevicePosition` to verify the integrity of the device's position by determining if it was reported behind a proxy, and by comparing it to an inferred position estimated based on the device's state.

Price is based on the number of API requests.

Trackers quota and usage

This topic provides a summary of rate limits and quotas for Amazon Location Service trackers.

Note

If you require a higher quota, you can use the Service Quotas console to [request quota increases](#) for adjustable quotas. When requesting a quota increase, select the Region you require the quota increase in, since most quotas are specific to the AWS Region. You can request up to twice the default limit for each API.

For requests that exceed twice the default limit, your request will submit a support ticket. You can also connect to your premium support team. There are no direct charges for quota increase requests, but higher usage levels may lead to increased service costs based on the additional resources consumed. See [the section called "Manage quotas"](#) for more information.

Service Quotas are maximum number of resources you can have per AWS account and AWS Region. Amazon Location Service denies additional requests that exceed the service quota.

Resources

API name	Default	Max adjustable limit
Tracker resources per account	500	1000 If you need more than this, request quota increases or contact the support team.
Tracker consumers per tracker	5	Max adjustable limit is not applicable.

API name	Default	Max adjustable limit
		Contact the support team.

CRUD API

Note

If you need a higher limit for any of these APIs, [request quota increases](#) or contact the support team.

API name	Default	Max adjustable limit
AssociateTrackerConsumer	10	20
CreateTracker	10	20
DeleteTracker	10	20
DescribeTracker	10	20
DisassociateTrackerConsumer	10	20
ListTrackerConsumers	10	20
ListTrackers	10	20
UpdateTracker	10	20

Data API

Note

If you need a higher limit for any of these APIs, [request quota increases](#) or contact the support team.

API name	Default	Max adjustable limit
BatchGetDevicePosition	50	100
BatchUpdateDevicePosition	50	100
GetDevicePosition	50	100
GetDevicePositionHistory	50	100
BatchDeleteDevicePositionHistory	50	100
ListDevicePositions	50	100

Other usage limits

Developer tools for using Amazon Location Service

AWS provides Software Development Kits (SDKs) for multiple programming languages, allowing you to easily integrate the Amazon Location Service into your applications. This page outlines the available SDKs, their installation procedures, and code examples to help you get started with the Amazon Location Service in your preferred development environment.

There are several tools that will help you to use Amazon Location Service.

Topics

- [SDKs and frameworks for Amazon Location Service](#)
- [Amazon Location Service API and CLI](#)
- [Examples and Learning Resources](#)

SDKs and frameworks for Amazon Location Service

AWS provides Software Development Kits (SDKs) for multiple programming languages, allowing you to easily integrate the Amazon Location Service into your applications. This page outlines the available SDKs, their installation procedures, and code examples to help you get started with the Amazon Location Service in your preferred development environment.

There are several tools that will help you to use Amazon Location Service.

- **AWS SDKs** – The AWS software development kits (SDKs) are available in many popular programming languages, providing an API, code examples, and documentation that makes it easier to build applications in your preferred language. The AWS SDKs include the core Amazon Location APIs and functionality, including access to Maps, Places, Routes, Geofencing, and Trackers. To learn more about the SDKs available to use with Amazon Location Service for different applications and languages, see [the section called “SDKs by language”](#).
- **MapLibre** – Amazon Location Service recommends rendering maps using the [MapLibre](#) rendering engine. MapLibre is an engine for displaying maps in web or mobile applications. MapLibre also has a plugin model, and supports user interface for searching and routes in some languages and platforms. To learn more about using MapLibre and the functionality it provides, see [the section called “Use MapLibre tools”](#).
- **Amazon Location SDKs** – The Amazon Location SDKs are a set of open source libraries that make it easier to develop applications with Amazon Location Service. The libraries provide

functionality to support authentication for mobile and web applications, location tracking for mobile applications, conversion between Amazon Location data types and [GeoJSON](#), as well as a hosted package of the Amazon Location client for the AWS SDK v3. To learn more about the Amazon Location SDKs, see [the section called “SDKs by language”](#).

- **Amazon Location Migration SDK** – The Amazon Location Migration SDK provides a bridge that allows you to migrate existing applications from Google Maps to Amazon Location. The Migration SDK provides an option for your application built using the Google Maps SDK for JavaScript to use Amazon Location Service without needing to rewrite any of the application or business logic if Amazon Location supports the capabilities used. The Migration SDK redirects all API calls to the Amazon Location instead of Google Map. To get started, see the [Amazon Location Migration SDK](#) on GitHub.

Developer tutorials

Use this section to learn how to use various aspects of the Amazon Location Service SDK.

Topics

- [How to use authentication helpers](#)
- [Use Amazon Location MapLibre Geocoder GL plugin](#)
- [How to use Tracking SDKs](#)
- [Use MapLibre tools and related libraries with Amazon Location](#)

How to use authentication helpers

This section provides additional information about authentication helpers.

Web

The Amazon Location JavaScript authentication utilities assistw in authenticating when making Amazon Location Service API calls from JavaScript applications. These utilities specifically support authentication using API keys or Amazon Cognito.

Installation

- Install this library using NPM:

```
npm install @aws/amazon-location-utilities-auth-helper
```

- To use it directly in the browser, include the following in your HTML file:

```
<script src="https://cdn.jsdelivr.net/npm/@aws/amazon-location-utilities-auth-helper@1"></script>
```

Usage

To use the authentication helpers, import the library and call the necessary utility functions. This library supports authenticating requests from the Amazon Location Service SDKs, including the [Maps](#), [Places](#), and [Routes](#) standalone SDKs, as well as rendering maps with [MapLibre GL JS](#).

Usage with Modules

This example demonstrates the use of the standalone Places SDK to make a request authenticated with API keys:

```
npm install @aws-sdk/geo-places-client

import { GeoPlacesClient, GeocodeCommand } from "@aws-sdk/geo-places-client";
import { withAPIKey } from "@aws/amazon-location-utilities-auth-helper";

const authHelper = withAPIKey("<API Key>", "<Region>");
const client = new GeoPlacesClient(authHelper.getClientConfig());

const input = { ... };
const command = new GeocodeCommand(input);
const response = await client.send(command);
```

This example demonstrates the use of the standalone Routes SDK to make a request authenticated with API keys:

```
npm install @aws-sdk/geo-routes-client

import { GeoRoutesClient, CalculateRoutesCommand } from "@aws-sdk/geo-routes-client";
import { withAPIKey } from "@aws/amazon-location-utilities-auth-helper";

const authHelper = withAPIKey("<API Key>", "<Region>");
const client = new GeoRoutesClient(authHelper.getClientConfig());

const input = { ... };
const command = new CalculateRoutesCommand(input);
```

```
const response = await client.send(command);
```

This example uses the Location SDK with API key authentication:

```
npm install @aws-sdk/client-location

import { LocationClient, ListGeofencesCommand } from "@aws-sdk/client-location";
import { withAPIKey } from "@aws/amazon-location-utilities-auth-helper";

const authHelper = withAPIKey("<API Key>", "<Region>");
const client = new LocationClient(authHelper.getClientConfig());

const input = { ... };
const command = new ListGeofencesCommand(input);
const response = await client.send(command);
```

Usage with Browser

Utility functions are accessible under the `amazonLocationAuthHelper` global object when used directly in a browser environment.

This example demonstrates a request with the Amazon Location Client, authenticated using API keys:

```
<script src="https://cdn.jsdelivr.net/npm/@aws/amazon-location-client@1"></script>

const authHelper = amazonLocationClient.withAPIKey("<API Key>", "<Region>");
const client = new amazonLocationClient.GeoRoutesClient(authHelper.getClientConfig());
const input = { ... };
const command = new amazonLocationClient.routes.CalculateRoutesCommand(input);
const response = await client.send(command);
```

This example demonstrates rendering a map with MapLibre GL JS, authenticated with an API key:

```
<script src="https://cdn.jsdelivr.net/npm/maplibre-gl@4"></script>

const apiKey = "<API Key>";
const region = "<Region>";
const styleName = "Standard";

const map = new maplibregl.Map({
```

```
    container: "map",
    center: [-123.115898, 49.295868],
    zoom: 10,
    style: `https://maps.geo.${region}.amazonaws.com/v2/styles/${styleName}/descriptor?
key=${apiKey}`,
  });
```

This example demonstrates rendering a map with MapLibre GL JS using Amazon Cognito:

```
<script src="https://cdn.jsdelivr.net/npm/maplibre-gl@4"></script>
<script src="https://cdn.jsdelivr.net/npm/@aws/amazon-location-utilities-auth-
helper@1"></script>

const identityPoolId = "<Identity Pool ID>";
const authHelper = await amazonLocationAuthHelper.withIdentityPoolId(identityPoolId);

const map = new maplibregl.Map({
  container: "map",
  center: [-123.115898, 49.295868],
  zoom: 10,
  style: `https://maps.geo.${region}.amazonaws.com/v2/styles/${styleName}/descriptor`,
  ...authHelper.getMapAuthenticationOptions(),
});
```

Alternative Usage with Authenticated Identities

You can modify the `withIdentityPoolId` function to include custom parameters for authenticated identities:

```
const userPoolId = "<User Pool ID>";

const authHelper = await amazonLocationAuthHelper.withIdentityPoolId(identityPoolId, {
  logins: {
    [`cognito-idp.${region}.amazonaws.com/${userPoolId}`]: "cognito-id-token"
  }
});
```

iOS

The Amazon Location Service Mobile Authentication SDK for iOS helps authenticate requests to Amazon Location Service APIs from iOS applications. It specifically supports authentication via API keys or Amazon Cognito.

Installation

- Open Xcode and go to **File > Add Package Dependencies**.
- Type the package URL (<https://github.com/aws-geospatial/amazon-location-mobile-auth-sdk-ios/>) into the search bar and press Enter.
- Select the "amazon-location-mobile-auth-sdk-ios" package and click **Add Package**.
- Choose the "AmazonLocationiOSAuthSDK" package product and click **Add Package**.

Usage

After installing the library, use the `AuthHelper` class to configure client settings for either API keys or Amazon Cognito.

API Keys

Here is an example using the standalone Places SDK with API key authentication:

```
import AmazonLocationiOSAuthSDK
import AWSGeoPlaces

func geoPlacesExample() {
    let apiKey = "<API key>"
    let region = "<Region>"

    let authHelper = try await AuthHelper.withApiKey(apiKey: apiKey, region: region)
    let client: GeoPlacesClient = GeoPlacesClient(config:
authHelper.getGeoPlacesClientConfig())

    let input = AWSGeoPlaces.SearchTextInput(
        biasPosition: [-97.7457518, 30.268193],
        queryText: "tacos"
    )

    let output = try await client.searchText(input: input)
}
```

Here is an example using the standalone Routes SDK with API key authentication:

```
import AmazonLocationiOSAuthSDK
import AWSGeoRoutes
```

```
func geoRoutesExample() {
    let apiKey = "<API key>"
    let region = "<Region>"

    let authHelper = try await AuthHelper.withApiKey(apiKey: apiKey, region: region)
    let client: GeoRoutesClient = GeoRoutesClient(config:
authHelper.getGeoRoutesClientConfig())

    let input = AWSGeoRoutes.CalculateRoutesInput(
        destination: [-123.1651031, 49.2577281],
        origin: [-97.7457518, 30.268193]
    )

    let output = try await client.calculateRoutes(input: input)
}
```

Here is an example using the Location SDK with API key authentication:

```
import AmazonLocationiOSAuthSDK
import AWSLocation

func locationExample() {
    let apiKey = "<API key>"
    let region = "<Region>"

    let authHelper = try await AuthHelper.withApiKey(apiKey: apiKey, region: region)
    let client: LocationClient = LocationClient(config:
authHelper.getLocationClientConfig())

    let input = AWSLocation.ListGeofencesInput(
        collectionName: "<Collection name>"
    )

    let output = try await client.listGeofences(input: input)
}
```

Here is an example using the standalone Places SDK with Amazon Cognito:

```
import AmazonLocationiOSAuthSDK
import AWSGeoPlaces

func geoPlacesExample() {
```

```
let identityPoolId = "<Identity Pool ID>"

let authHelper = try await AuthHelper.withIdentityPoolId(identityPoolId:
identityPoolId)
let client: GeoPlacesClient = GeoPlacesClient(config:
authHelper.getGeoPlacesClientConfig())

let input = AWSGeoPlaces.SearchTextInput(
    biasPosition: [-97.7457518, 30.268193],
    queryText: "tacos"
)

let output = try await client.searchText(input: input)
}
```

Here is an example using the standalone Routes SDK with Amazon Cognito:

```
import AmazonLocationiOSAuthSDK
import AWSGeoRoutes

func geoRoutesExample() {
    let identityPoolId = "<Identity Pool ID>"

    let authHelper = try await AuthHelper.withIdentityPoolId(identityPoolId:
identityPoolId)
    let client: GeoRoutesClient = GeoRoutesClient(config:
authHelper.getGeoRoutesClientConfig())

    let input = AWSGeoRoutes.CalculateRoutesInput(
        destination: [-123.1651031, 49.2577281],
        origin: [-97.7457518, 30.268193]
    )

    let output = try await client.calculateRoutes(input: input)
}
```

Here is an example using the Location SDK with Amazon Cognito:

```
import AmazonLocationiOSAuthSDK
import AWSLocation

func locationExample() {
    let identityPoolId = "<Identity Pool ID>"
```

```
    let authHelper = try await AuthHelper.withIdentityPoolId(identityPoolId:
identityPoolId)
    let client: LocationClient = LocationClient(config:
authHelper.getLocationClientConfig())

    let input = AWSLocation.ListGeofencesInput(
        collectionName: "<Collection name>"
    )

    let output = try await client.listGeofences(input: input)
}
```

Android

The Amazon Location Service Mobile Authentication SDK for Android helps you authenticate requests to Amazon Location Service APIs from Android applications, specifically supporting authentication using Amazon Cognito.

Installation

- This authentication SDK works with the overall AWS Kotlin SDK. Both SDKs are published to Maven Central. Check the latest version of the [auth SDK](#) on Maven Central.
- Add the following lines to the dependencies section of your `build.gradle` file in Android Studio:

```
implementation("software.amazon.location:auth:1.1.0")
implementation("org.maplibre.gl:android-sdk:11.5.2")
implementation("com.squareup.okhttp3:okhttp:4.12.0")
```

- For the standalone Maps, Places, and Routes SDKs, add the following lines:

```
implementation("aws.sdk.kotlin:geomaps:1.3.65")
implementation("aws.sdk.kotlin:geoplaces:1.3.65")
implementation("aws.sdk.kotlin:georoutes:1.3.65")
```

- For the consolidated Location SDK that includes Geofencing and Tracking, add the following line:

```
implementation("aws.sdk.kotlin:location:1.3.65")
```


Usage

Import the following classes in your code:

```
// For the standalone Maps, Places, and Routes SDKs
import aws.sdk.kotlin.services.geomaps.GeoMapsClient
import aws.sdk.kotlin.services.geoplaces.GeoPlacesClient
import aws.sdk.kotlin.services.georoutes.GeoRoutesClient

// For the consolidated Location SDK
import aws.sdk.kotlin.services.location.LocationClient

import software.amazon.location.auth.AuthHelper
import software.amazon.location.auth.LocationCredentialsProvider
import software.amazon.location.auth.AwsSignerInterceptor
import org.maplibre.android.module.http.HttpRequestUtil
import okhttp3.OkHttpClient
```

You can create an `AuthHelper` and use it with the AWS Kotlin SDK:

Example: Credential Provider with Identity Pool ID

```
private suspend fun exampleCognitoLogin() {
    val authHelper = AuthHelper.withCognitoIdentityPool("MY-COGNITO-IDENTITY-POOL-ID",
        applicationContext)

    var geoMapsClient = GeoMapsClient(authHelper?.getGeoMapsClientConfig())
    var geoPlacesClient = GeoPlacesClient(authHelper?.getGeoPlacesClientConfig())
    var geoRoutesClient = GeoRoutesClient(authHelper?.getGeoRoutesClientConfig())

    var locationClient = LocationClient(authHelper?.getLocationClientConfig())
}
```

Example: Credential Provider with Custom Credential Provider

```
private suspend fun exampleCustomCredentialLogin() {
    var authHelper = AuthHelper.withCredentialsProvider(MY-CUSTOM-CREDENTIAL-PROVIDER,
        "MY-AWS-REGION", applicationContext)

    var geoMapsClient = GeoMapsClient(authHelper?.getGeoMapsClientConfig())
    var geoPlacesClient = GeoPlacesClient(authHelper?.getGeoPlacesClientConfig())
    var geoRoutesClient = GeoRoutesClient(authHelper?.getGeoRoutesClientConfig())
}
```

```

    var locationClient = LocationClient(authHelper?.getLocationClientConfig())
}

```

Example: Credential Provider with API Key

```

private suspend fun exampleApiKeyLogin() {
    var authHelper = AuthHelper.withApiKey("MY-API-KEY", "MY-AWS-REGION",
        applicationContext)

    var geoMapsClient = GeoMapsClient(authHelper?.getGeoMapsClientConfig())
    var geoPlacesClient = GeoPlacesClient(authHelper?.getGeoPlacesClientConfig())
    var geoRoutesClient = GeoRoutesClient(authHelper?.getGeoRoutesClientConfig())

    var locationClient = LocationClient(authHelper?.getLocationClientConfig())
}

```

You can use `LocationCredentialsProvider` to load the MapLibre map. Here is an example:

```

HttpRequestUtil.setOkHttpClient(
    OkHttpClient.Builder()
        .addInterceptor(
            AwsSignerInterceptor(
                "geo",
                "MY-AWS-REGION",
                locationCredentialsProvider,
                applicationContext
            )
        )
        .build()
)

```

Use the created clients to make calls to Amazon Location Service. Here is an example that searches for places near a specified latitude and longitude:

```

val suggestRequest = SuggestRequest {
    biasPosition = listOf(-97.718833, 30.405423)
    maxResults = MAX_RESULT
    language = "PREFERRED-LANGUAGE"
}
val nearbyPlaces = geoPlacesClient.suggest(suggestRequest)

```

Use Amazon Location MapLibre Geocoder GL plugin

The Amazon Location MapLibre geocoder plugin is designed to make it easier for you to incorporate Amazon Location functionality into your JavaScript applications, when working with map rendering and geocoding using the [maplibre-gl-geocoder](#) library.

Installation

Install Amazon Location MapLibre geocoder plugin from NPM for usage with modules. Type this command:

```
npm install @aws/amazon-location-for-maplibre-gl-geocoder
```

You can also import HTML and CSS files for usage directly in the browser with a script:

```
<script src="https://cdn.jsdelivr.net/npm/@aws/amazon-location-for-maplibre-gl-geocoder@2"></script>
<link
  href="https://cdn.jsdelivr.net/npm/@aws/amazon-location-for-maplibre-gl-geocoder@2/dist/amazon-location-for-mlg-styles.css"
  rel="stylesheet"
/>
```

Usage with module - standalone GeoPlaces SDK

This example uses the [AWS SDK for JavaScript V3](#) to get a GeoPlacesClient to provide to the library and [AuthHelper](#) for authenticating the GeoPlacesClient. It enables all APIs for the geocoder.

```
// Import MapLibre GL JS
import maplibregl from "maplibre-gl";
// Import from the AWS JavaScript SDK V3
import { GeoPlacesClient } from "@aws-sdk/client-geo-places";
// Import the utility functions
import { withAPIKey } from "@aws/amazon-location-utilities-auth-helper";
// Import the AmazonLocationMaplibreGeocoder
import {
  buildAmazonLocationMaplibreGeocoder,
  AmazonLocationMaplibreGeocoder,
} from "@aws/amazon-location-for-maplibre-gl-geocoder";

const apiKey = "<API Key>";
const mapName = "Standard";
```

```
const region = "<Region>"; // region containing Amazon Location API Key

// Create an authentication helper instance using an API key and region
const authHelper = await withAPIKey(apiKey, region);

const client = new GeoPlacesClient(authHelper.getClientConfig());

// Render the map
const map = new maplibregl.Map({
  container: "map",
  center: [-123.115898, 49.295868],
  zoom: 10,
  style: `https://maps.geo.${region}.amazonaws.com/maps/v2/styles/${mapStyle}/
descriptor?key=${apiKey}`,
});

// Gets an instance of the AmazonLocationMaplibreGeocoder Object.
const amazonLocationMaplibreGeocoder = buildAmazonLocationMaplibreGeocoder(client,
  { enableAll: true });

// Now we can add the Geocoder to the map.
map.addControl(amazonLocationMaplibreGeocoder.getPlacesGeocoder());
```

Usage with a browser - standalone GeoPlaces SDK

This example uses the Amazon Location client to make a request that authenticates using an API key.

Note

Some of these example use the Amazon Location GeoPlacesClient. This client is based on the [AWS SDK for JavaScript V3](#) and allows for making calls to Amazon Location through a script referenced in an HTML file.

Include the following in an HTML file:

```
<!-- Import the Amazon Location For Maplibre Geocoder -->
<script src="https://cdn.jsdelivr.net/npm/@aws/amazon-location-for-maplibre-gl-
geocoder@2"></script>
<link
```

```

href="https://cdn.jsdelivr.net/npm/@aws/amazon-location-for-maplibre-gl-geocoder@2/
dist/amazon-location-for-mlg-styles.css"
rel="stylesheet"
/>
<!-- Import the Amazon GeoPlacesClient -->
<script src="https://cdn.jsdelivr.net/npm/@aws/amazon-location-client@1"></script>

```

Include the following in a JavaScript file:

```

const apiKey = "<API Key>";
const mapStyle = "Standard";
const region = "<Region>"; // region containing Amazon Location API key

// Create an authentication helper instance using an API key and region
const authHelper = await amazonLocationClient.withAPIKey(apiKey, region);

const client = new amazonLocationClient.GeoPlacesClient(authHelper.getClientConfig());

// Render the map
const map = new maplibregl.Map({
  container: "map",
  center: [-123.115898, 49.295868],
  zoom: 10,
  style: `https://maps.geo.${region}.amazonaws.com/maps/v2/styles/${mapStyle}/
descriptor?key=${apiKey}`,
});

// Initialize the AmazonLocationMaplibreGeocoder object
const amazonLocationMaplibreGeocoderObject =
  amazonLocationMaplibreGeocoder.buildAmazonLocationMaplibreGeocoder(
    client,
    { enableAll: true },
  );

// Use the AmazonLocationWithMaplibreGeocoder object to add a geocoder to the map.
map.addControl(amazonLocationMaplibreGeocoderObject.getPlacesGeocoder());

```

Functions

Listed below are the functions used in the Amazon Location MapLibre geocoder plugin:

- `buildAmazonLocationMaplibreGeocoder`

This class creates an instance of the `AmazonLocationMaplibreGeocoder`, which is the entry point to the other all other calls.

Using standalone `GeoPlacesClient` API calls (client is instance of `GeoPlacesClient`):

```
const amazonLocationMaplibreGeocoder = buildAmazonLocationMaplibreGeocoder(client,
  { enableAll: true });
```

Using consolidated `LocationClient` API calls (client is instance of `LocationClient`):

```
const amazonLocationMaplibreGeocoder = buildAmazonLocationMaplibreGeocoder(client, {
  enableAll: true,
  placesIndex: placeIndex,
});
```

- `getPlacesGeocoder`

Returns a ready-to-use `IControl` object that can be added directly to a map.

```
const geocoder = getPlacesGeocoder();

// Initialize map see: <insert link to initializing a map instance here>
let map = await initializeMap();

// Add the geocoder to the map.
map.addControl(geocoder);
```

How to use Tracking SDKs

This topic provides information about how to use Tracking SDKs.

iOS

The Amazon Location mobile tracking SDK provides utilities which help easily authenticate, capture device positions, and send position updates to Amazon Location Trackers. The SDK supports local filtering of location updates with configurable update intervals. This reduces data costs and optimizes intermittent connectivity for your iOS applications.

The iOS tracking SDK is available on GitHub: [Amazon Location Mobile Tracking SDK for iOS](#).

This section covers the following topics for the Amazon Location mobile tracking iOS SDK:

Topics

- [Installation](#)
- [Usage](#)
- [Filters](#)
- [iOS Mobile SDK tracking functions](#)
- [Examples](#)

Installation

Use the following procedure to install the mobile tracking SDK for iOS:

1. In your Xcode project, go to **File** and select **Add Package Dependencies**.
2. Type the following URL: <https://github.com/aws-geospatial/amazon-location-mobile-tracking-sdk-ios/> into the search bar and press the enter key.
3. Select the amazon-location-mobile-tracking-sdk-ios package and click on **Add Package**.
4. Select the AmazonLocationiOSTrackingSDK package product and click on **Add Package**.

Usage

The following procedure shows you how to create an authentication helper using credentials from Amazon Cognito.

1. After installing the library, you need to add one or both of the descriptions into your `info.plist` file:

```
Privacy - Location When In Use Usage Description
Privacy - Location Always and When In Use Usage Description
```

2. Next, import the AuthHelper in your class:

```
import AmazonLocationiOSAuthSDKimport AmazonLocationiOSTrackingSDK
```

3. Then you will create an AuthHelper object and use it with the AWS SDK, by creating an authentication helper using credentials from Amazon Cognito.

```
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)
```

Filters

The Amazon Location mobile tracking iOS SDK has three inbuilt location filters.

- **TimeLocationFilter**: Filters the current location to be uploaded based on a defined time interval.
- **DistanceLocationFilter**: Filters location updates based on a specified distance threshold.
- **AccuracyLocationFilter**: Filters location updates by comparing the distance moved since the last update with the current location's accuracy.

This example adds filters in the LocationTracker at the creation time:

```
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,
)

```

This example enables and disables filter at runtime with `LocationTracker`:

```

// To enable the filter
locationTracker?.enableFilter(TimeLocationFilter())

// To disable the filter
locationTracker?.disableFilter(TimeLocationFilter())

```

iOS Mobile SDK tracking functions

The Amazon Location mobile tracking SDK for iOS includes the following functions:

- **Class:** `LocationTracker`

```

init(provider: LocationCredentialsProvider, trackerName: String, config:
LocationTrackerConfig? = nil)

```

This is an initializer function to create a `LocationTracker` object. It requires instances of `LocationCredentialsProvider`, `trackerName` and optionally an instance of `LocationTrackingConfig`. If the config is not provided it will be initialized with default values.

- **Class:** `LocationTracker`

```

setTrackerConfig(config: LocationTrackerConfig)

```

This sets Tracker's config to take effect at any point after initialization of location tracker.

- **Class:** `LocationTracker`

```

getTrackerConfig()

```

This gets the location tracking config to use or modify in your app.

Returns: `LocationTrackerConfig`

- **Class:** `LocationTracker`

`getDeviceId()`

Gets the location tracker's generated device Id.

Returns: `String?`

- **Class:** `LocationTracker`

`startTracking()`

Starts the process of accessing the user's location and sending it to the AWS tracker.

- **Class:** `LocationTracker`

`resumeTracking()`

Resumes the process of accessing the user's location and sending it to the AWS tracker.

- **Class:** `LocationTracker`

`stopTracking()`

Stops the process of tracking the user's location.

- **Class:** `LocationTracker`

`startBackgroundTracking(mode: BackgroundTrackingMode)`

Starts the process of accessing the user's location and sending it to the AWS tracker while the application is in the background. `BackgroundTrackingMode` has the following options:

- **Active:** This option doesn't automatically pauses location updates.
 - **BatterySaving:** This option automatically pauses location updates.
 - **None:** This option overall disables background location updates.
- **Class:** `LocationTracker`

`resumeBackgroundTracking(mode: BackgroundTrackingMode)`

Resumes the process of accessing the user's location and sending it to the AWS tracker while the application is in the background.

- **Class:** `LocationTracker`

```
stopBackgroundTracking()
```

Stops the process of accessing the user's location and sending it to the AWS tracker while the application is in the background.

- **Class:** `LocationTracker`

```
getTrackerDeviceLocation(nextToken: String?, startTime: Date? = nil,  
endTime: Date? = nil, completion: @escaping (Result<GetLocationResponse,  
Error>)
```

Retrieves the uploaded tracking locations for the user's device between start and end date and time.

Returns: `Void`

- **Class:** `LocationTrackerConfig`

```
init()
```

This initializes the `LocationTrackerConfig` with default values.

- **Class:** `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)
```

This initializes the `LocationTrackerConfig` with user-defined parameter values. If a parameter value is not provided it will be set to a default value.

- **Class:** `LocationFilter`

```
shouldUpload(currentLocation: LocationEntity, previousLocation:  
LocationEntity?, trackerConfig: LocationTrackerConfig)
```

The `LocationFilter` is a protocol that users can implement for their custom filter implementation. A user would need to implement `shouldUpload` function to compare previous and current location and return if the current location should be uploaded.

Examples

This sections details examples of using the Amazon Location Mobile Tracking SDK for iOS.

Note

Ensure that the necessary permissions are set in the `info.plist` file. These are the same permissions listed in the [the section called "Usage"](#) section.

The following example demonstrates functionality for tracking device location and retrieving tracked locations:

```
Privacy - Location When In Use Usage Description
Privacy - Location Always and When In Use Usage Description
```

Start tracking the location:

```
do {
    try locationTracker.startTracking()
    }
catch TrackingLocationError.permissionDenied {
    // Handle permissionDenied by showing the alert message or opening the app
    settings
    }
}
```

Resume tracking the location:

```
do {
    try locationTracker.resumeTracking()
    }
catch TrackingLocationError.permissionDenied {
    // Handle permissionDenied by showing the alert message or opening the app settings
    }
}
```

Stop tracking the location:

```
locationTracker.stopTracking()
```

Start background tracking:

```
do {  
    locationTracker.startBackgroundTracking(mode: .Active) // .Active, .BatterySaving, .None  
    }  
catch TrackingLocationError.permissionDenied {  
    // Handle permissionDenied by showing the alert message or opening the app settings  
    }
```

Resume background tracking:

```
do {  
    locationTracker.resumeBackgroundTracking(mode: .Active)  
    }  
catch TrackingLocationError.permissionDenied {  
    // Handle permissionDenied by showing the alert message or opening the app settings  
    }
```

To stop background tracking:

```
locationTracker.stopBackgroundTracking()
```

Retrieve device's tracked locations from the tracker:

```
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)
    }
})
}
```

Android Mobile Tracking SDK

The Amazon Location mobile tracking SDK provides utilities which help easily authenticate, capture device positions, and send position updates to Amazon Location Trackers. The SDK supports local filtering of location updates with configurable update intervals. This reduces data costs and optimizes intermittent connectivity for your Android applications.

The Android tracking SDK is available on GitHub: [Amazon Location Mobile Tracking SDK for Android](#). Additionally, both the mobile authentication SDK and the AWS SDK are available on the [AWS Maven repository](#). The Android tracking SDK is designed to work with the general AWS SDK.

This section covers the following topics for the Amazon Location mobile tracking Android SDK:

Topics

- [Installation](#)
- [Usage](#)
- [Filters](#)
- [Android Mobile SDK tracking functions](#)
- [Examples](#)

Installation

To install the SDK, add the following lines to the dependencies section of your build.gradle file in Android Studio:

```
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")
```

Usage

This procedure shows you how to use the SDK to authenticate and create the `LocationTracker` object:

Note

This procedure assumes you have imported the library mentioned in the [the section called "Installation"](#) section.

1. Import the following classes in your code:

```
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. Next create an `AuthHelper`, since the `LocationCredentialsProvider` parameter is required for creating a `LocationTracker` object:

```
// Create an authentication helper using credentials from Amazon Cognito
val authHelper = AuthHelper(applicationContext)
val locationCredentialsProvider : LocationCredentialsProvider =
    authHelper.authenticateWithCognitoIdentityPool("My-Cognito-Identity-Pool-Id")
```

3. Now, use the `LocationCredentialsProvider` and `LocationTrackerConfig` to create a `LocationTracker` object:

```
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,  
)
```

Filters

The Amazon Location mobile tracking Android SDK has three inbuilt location filters.

- **TimeLocationFilter**: Filters the current location to be uploaded based on a defined time interval.
- **DistanceLocationFilter**: Filters location updates based on a specified distance threshold.
- **AccuracyLocationFilter**: Filters location updates by comparing the distance moved since the last update with the current location's accuracy.

This example adds filters in the `LocationTracker` at the creation time:

```
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,  
)
```

This example enables and disables filter at runtime with `LocationTracker`:

```
// To enable the filter
```



```
locationTracker?.enableFilter(TimeLocationFilter())

// To disable the filter
locationTracker?.disableFilter(TimeLocationFilter())
```

Android Mobile SDK tracking functions

The Amazon Location mobile tracking SDK for Android includes the following functions:

- **Class:** `LocationTracker`

```
constructor(context: Context,locationCredentialsProvider:
LocationCredentialsProvider,trackerName: String),or
constructor(context: Context,locationCredentialsProvider:
LocationCredentialsProvider,clientConfig: LocationTrackerConfig)
```

This is an initializer function to create a `LocationTracker` object. It requires instances of `LocationCredentialsProvider`, `trackerName` and optionally an instance of `LocationTrackingConfig`. If the config is not provided it will be initialized with default values.

- **Class:** `LocationTracker`

```
start(locationTrackingCallback: LocationTrackingCallback)
```

Starts the process of accessing the user's location and sending it to an Amazon Location tracker.

- **Class:** `LocationTracker`

```
isTrackingInForeground()
```

Checks if location tracking is currently in progress.

- **Class:** `LocationTracker`

```
stop()
```

Stops the process of tracking the user's location.

- **Class:** `LocationTracker`

```
startTracking()
```

Starts the process of accessing the user's location and sending it to the AWS tracker.

- **Class:** `LocationTracker`

```
startBackground(mode: BackgroundTrackingMode, serviceCallback: ServiceCallback)
```

Starts the process of accessing the user's location and sending it to the AWS tracker while the application is in the background. `BackgroundTrackingMode` has the following options:

- `ACTIVE_TRACKING`: This option actively tracks a user's location updates.
- `BATTERY_SAVER_TRACKING`: This option tracks user's location updates every 15 minutes.

- **Class:** `LocationTracker`

```
stopBackgroundService()
```

Stops the process of accessing the user's location and sending it to the AWS tracker while the application is in the background.

- **Class:** `LocationTracker`

```
getTrackerDeviceLocation()
```

Retrieves the device location from Amazon Location services.

- **Class:** `LocationTracker`

```
getDeviceLocation(locationTrackingCallback: LocationTrackingCallback?)
```

Retrieves the current device location from the fused location provider client and uploads it to Amazon Location tracker.

- **Class:** `LocationTracker`

```
uploadLocationUpdates(locationTrackingCallback: LocationTrackingCallback?)
```

Uploads the device location to Amazon Location services after filtering based on the configured location filters.

- **Class:** `LocationTracker`

```
enableFilter(filter: LocationFilter)
```

Enables a particular location filter.

- **Class:** `LocationTracker`

```
checkFilterIsExistsAndUpdateValue(filter: LocationFilter)
```

Disable particular location filter.

- **Class:** 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())
```

This initializes the LocationTrackerConfig with user-defined parameter values. If a parameter value is not provided, it will be set to a default value.

- **Class:** LocationFilter

```
shouldUpload(currentLocation: LocationEntry, previousLocation:
LocationEntry?): Boolean
```

The LocationFilter is a protocol that users can implement for their custom filter implementation. You need to implement the shouldUpload function to compare previous and current location and return if the current location should be uploaded.

Examples

The following code sample shows the mobile tracking SDK functionality.

This example uses the LocationTracker to start and stop tracking in background:

```
// 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()
```

Use MapLibre tools and related libraries with Amazon Location

[MapLibre](#) is primarily a rendering engine for displaying maps in a web or mobile application. However, it also includes support for plug-ins and provides functionality for working with other aspects of Amazon Location. The following describes tools that you can use, based on the area or location that you want to work with.

Note

To use any aspect of Amazon Location, install the [AWS SDK for the language that you want to use](#).

• Maps

To display maps in your application, you need a map rendering engine that will use the data provided by Amazon Location, and draw to the screen. Map rendering engines also provide functionality to pan and zoom the map, or to add markers or pushpins and other annotations to the map.

Amazon Location Service recommends rendering maps using the [MapLibre](#) rendering engine. MapLibre GL JS is an engine for displaying maps in JavaScript, while MapLibre Native provides maps for either iOS or Android.

MapLibre also has a plug-in ecosystem to extend the core functionality. For more information, visit <https://maplibre.org/maplibre-gl-js-docs/plugins/>.

• Places search

To make creating a search user interface simpler, you can use the [MapLibre geocoder](#) for web (Android applications can use the [Android Places plug-in](#)).

Use the [Amazon Location for MapLibre geocoder library](#) to simplify the process of using Amazon Location with `amazon-location-for-maplibre-gl-geocoder` in JavaScript Applications.

For more information, see [the section called “Use MapLibre Geocoder GL plugin”](#).

- **Routes**
- **Geofences and Trackers**

MapLibre doesn't have any specific rendering or tools for geofences and tracking, but you can use the rendering functionality and [plug-ins](#) to show the geofences and tracked devices on the map.

The devices being tracked can use [MQTT](#) or manually send updates to Amazon Location Service. Geofence events can be responded to using [AWS Lambda](#).

Many open source libraries are available to provide additional functionality for Amazon Location Service, for example [Turf](#) which provide spatial analysis functionality.

Many libraries use the open standard [GeoJSON](#) formatted data. Amazon Location Service provides a library to convert responses into GeoJSON for use in JavaScript applications. For more information, see [the section called “SDKs and frameworks”](#).

SDKs by language

SDK Versions

We recommend that you use the most recent build of the AWS SDK, and any other SDKs, that you use in your projects, and to keep the SDKs up to date. The AWS SDK provides you with the latest features and functionality, and also security updates. To find the latest build of the AWS SDK for JavaScript, for example, see the [browser installation](#) topic in the *AWS SDK for JavaScript* documentation.

The following tables provide information about AWS SDKs and Map Rendering Framework versions for languages and frameworks, by application type: web, mobile, or backend application.

Web frontend

The following AWS SDKs and Map Rendering Framework versions are available for web frontend application development.

Language / Framework	AWS SDK	Map Rendering Framework
Fully supported		
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/
Partially supported		
Flutter	https://docs.amplify.aws/start/q/integration/flutter/ Flutter is not yet fully supported by AWS, but limited support is offered via Amplify.	https://github.com/maplibre/flutter-maplibre-gl The MapLibre Flutter library is considered experimental.
Node.js	https://aws.amazon.com/sdk-for-javascript/	https://github.com/maplibre/maplibre-native https://www.npmjs.com/package/@maplibre/maplibre-gl-native
PHP	https://aws.amazon.com/sdk-for-php/	There is no MapLibre support for PHP.

Mobile frontend

The following AWS SDKs and Map Rendering Framework versions are available for mobile frontend application development.

Language / Framework	AWS SDK	Map Rendering Framework
Fully supported		
Java	https://aws.amazon.com/sdk-for-java/	https://maplibre.org/projects/maplibre-native/
Kotlin	https://aws.amazon.com/sdk-for-kotlin/ Amazon Location Service Mobile Authentication SDK for Android: 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/ Requires custom bindings, as MapLibre is Java-based.
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
Swift	https://aws.amazon.com/sdk-for-swift/ Amazon Location Service Mobile Authentication SDK	https://maplibre.org/projects/maplibre-native/

Language / Framework	AWS SDK	Map Rendering Framework
	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	

Partially supported

Flutter	https://docs.amplify.aws/start/q/integration/flutter/ Flutter is not yet fully supported by AWS, but limited support is offered via Amplify.	https://github.com/maplibre/flutter-maplibre-gl The MapLibre Flutter library is considered experimental.
---------	---	---

Backend application

The following AWS SDKs are available for backend application development. Map Rendering Frameworks are not listed here, because map rendering is not typically needed for backend applications.

Language	AWS SDK
.NET	https://aws.amazon.com/sdk-for-net/
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/

Language	AWS SDK
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/ The AWS SDK for Rust is in developer preview.

Map Rendering SDK by language

We recommend rendering Amazon Location Service maps using the [MapLibre](#) rendering engine.

MapLibre is an engine for displaying maps in web or mobile applications. MapLibre also has a plugin model and supports user interfaces for searching and routes in some languages and platforms.

To learn more about using MapLibre and the functionality it provides, see [the section called “Use MapLibre tools”](#) and [the section called “Dynamic maps”](#).

The following tables provide information about Map Rendering SDKs versions for languages and frameworks, by application type: web or mobile application.

Web frontend

The following Map Rendering SDKs are available for web frontend application development.

Language / Framework	Map Rendering Framework
Fully supported	
JavaScript	https://maplibre.org/projects/maplibre-gl-js/
ReactJS	https://github.com/maplibre/maplibre-react-native
TypeScript	https://maplibre.org/projects/maplibre-gl-js/
Partially supported	
Flutter	https://github.com/maplibre/flutter-maplibre-gl The MapLibre Flutter library is considered experimental.
Node.js	There is no MapLibre support for Node.js.
PHP	There is no MapLibre support for PHP.

Mobile frontend

The following Map Rendering SDKs are available for mobile frontend application development.

Language / Framework	Map Rendering Framework
Fully supported	
Java	https://maplibre.org/projects/maplibre-native/
Kotlin	https://maplibre.org/projects/maplibre-native/ Requires custom bindings, as MapLibre is Java-based.
ObjectiveC	https://maplibre.org/projects/maplibre-native/
ReactNative	https://github.com/maplibre/maplibre-react-native

Language / Framework	Map Rendering Framework
Swift	https://maplibre.org/projects/maplibre-native/
Partially supported	
Flutter	https://github.com/maplibre/flutter-maplibre-gl The MapLibre Flutter library is considered experimental.

Amazon Location Service API and CLI

Amazon Location Service provides API and CLI operations access to the location functionality. See the lists below for more information.

Amazon Location Service API

This includes the following APIs:

- [Places](#)
- [Routes](#)
- [Authentication](#)
- [Maps](#)
- [Geofences](#)
- [Trackers](#)
- [Tags](#)

Amazon Location Service CLI

This includes the following CLIs:

AWS CLI Operations for Amazon Location Service

Amazon Location Service provides AWS CLI (Command-line interface) operations to access location functionality, including the following APIs:

Places

- [autocomplete](#)
- [geocode](#)
- [get-place](#)
- [reverse-geocode](#)
- [search-nearby](#)
- [search-text](#)
- [Learn More](#)

Routes

- [calculate-isolines](#)
- [calculate-route-matrix](#)
- [calculate-routes](#)
- [optimize-waypoints](#)
- [snap-to-roads](#)
- [Learn More](#)

Authentication

- [create-key](#)
- [delete-key](#)
- [describe-key](#)
- [list-keys](#)
- [update-key](#)
- [Learn More](#)

Maps

- [get-glyphs](#)
- [get-sprites](#)
- [get-static-map](#)
- [get-style-descriptor](#)
- [get-tile](#)
- [Learn More](#)

Geofences

- [batch-delete-geofence](#)
- [batch-evaluate-geofences](#)
- [batch-put-geofence](#)
- [forecast-geofence-events](#)
- [create-geofence-collection](#)
- [delete-geofence-collection](#)
- [describe-geofence-collection](#)
- [list-geofence-collections](#)
- [update-geofence-collection](#)
- [get-geofence](#)
- [list-geofences](#)
- [put-geofence](#)
- [Learn More](#)

Trackers

- [batch-get-device-position](#)
- [batch-update-device-position](#)
- [batch-delete-device-position-history](#)
- [get-device-position](#)
- [get-device-position-history](#)
- [associate-tracker-consumer](#)
- [disassociate-tracker-consumer](#)
- [create-tracker](#)
- [delete-tracker](#)
- [describe-tracker](#)
- [list-trackers](#)
- [update-tracker](#)
- [list-tracker-consumers](#)
- [verify-device-position](#)
- [Learn More](#)

Tags

- [list-tags-for-resource](#)
- [tag-resource](#)
- [untag-resource](#)
- [Learn More](#)

Examples and Learning Resources

This topic provides links and details about our available demos and sample projects. These resources are designed to help you quickly understand and implement key features of our tools and APIs. Additionally, you'll find links to GitHub repositories containing source code, and example solutions.

Demos

- **Web**

- Visit: <https://location.aws.com/demo>
- Explore source code: <https://github.com/aws-geospatial/amazon-location-features-demo-web>

- **Android**

- Install: <https://play.google.com/store/apps/details?id=com.aws.amazonlocation&trk=devguide>
- Explore source code: <https://github.com/aws-geospatial/amazon-location-features-demo-android>

- **iOS**

- Do the following to use our app:
 - Install [TestFlight](#)
 - [Join the Amazon Location beta](#)
- Explore source code: <https://github.com/aws-geospatial/amazon-location-features-demo-ios>

Samples

1. Visit: <https://location.aws.com/samples>.
2. Explore source code:

- JS: <https://github.com/aws-geospatial/amazon-location-samples-js/>
- React: <https://github.com/aws-geospatial/amazon-location-samples-react>
- Swift: <https://github.com/aws-geospatial/amazon-location-samples-ios>
- Android: <https://github.com/aws-geospatial/amazon-location-samples-android>

GitHub

See <https://github.com/aws-geospatial/>.

AWS integration for Amazon Location Service

Amazon Location Service is integrated with various AWS services for efficient authentication, monitoring, management and development.

Monitor

- **Amazon CloudWatch** – View metrics on service usage and health, including requests, latency, faults, and logs. For more information, see [the section called “Monitor with Amazon CloudWatch”](#).
- **AWS CloudTrail** – Log and monitor your API calls, which include actions taken by a user, role or an AWS service. For more information, see [the section called “Monitor and log with AWS CloudTrail”](#).

Manage

- **AWS CloudFormation** – Amazon Location is integrated with AWS CloudFormation, a service that helps you to model and set up your AWS resources so that you can spend less time creating and managing your resources and infrastructure. For more information, see [the section called “Create resources with AWS CloudFormation”](#).
- **Service Quotas** – Use the Service Quotas console and AWS CLI to request changes to your adjustable quotas. For more information, see [the section called “Manage quotas”](#).
- **Tags** – Use resource tagging in Amazon Location to create tags to categorize your resources by purpose, owner, environment, or criteria. Tagging your resources helps you manage, identify, organize, search, and filter your resources. For more information, see [the section called “Manage resources with Tags”](#).

Authenticate

- **Amazon Cognito** – You can use Amazon Cognito authentication as an alternative to directly using AWS Identity and Access Management (IAM) with both frontend SDKs and direct HTTPS requests. For more information, see [the section called “Use Amazon Cognito”](#).
- **IAM** – AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use Amazon Location Service resources. For more information, see [the section called “Use IAM”](#).

Value added

- **Amazon EventBridge** – Enable an event-driven application architecture so you can use AWS Lambda functions to activate other parts of your application and work flows. For more information, see [the section called “React to events with EventBridge”](#).
- **AWS IoT** – The AWS IoT Core rules engine stores queries about your devices' message topics and enables you to define actions for sending messages to other AWS services, such as Amazon Location Service. Devices that are aware of their location as coordinates can have their locations forwarded to Amazon Location through the rules engine. For more information, see [the section called “Track using AWS IoT and MQTT”](#).

Developer tool

- **SDKs** – Amazon Location Service offers a variety of tools for developers to build location-enabled applications. These include the standard AWS SDKs, mobile and web SDKs. For more information, see [the section called “SDKs and frameworks”](#).
- **AWS CLI** – The AWS Command Line Interface (AWS CLI) is an open source tool that enables you to interact with AWS services using commands in your command-line shell. With minimal configuration. For more information, see [AWS Command Line Interface](#) or learn more about [AWS CLI](#).
- **Sample code** – Sample code uses AWS SDKs, mobile and web SDKs, MapLibre to demonstrate how you can use Amazon Location. For more information, see [samples](#).
- **Amazon Location Service console** – Use the Amazon Location console to learn about APIs, resources, and to get started with a visual and interactive learning tool. For more information, see the [Amazon Location Service console](#).

Cost and billing

- **AWS Billing and Cost Management** – Service provides helps to you pay your bills and optimize your costs. Amazon Web Services bills your account for usage, which ensures that you pay only for what you use. For more information, see [the section called “Pricing model”](#) or [the section called “Manage billing and costs”](#).

Resource Management

Resource management provides tools and processes to manage quotas, organize resources with tags, control costs, and automate resource creation using AWS CloudFormation. These capabilities enable you to efficiently allocate, monitor, and manage your resources within Amazon Location.

Use these tools to maintain operational efficiency by setting service limits, tagging resources for better organization, monitoring your expenses, and using infrastructure as code with CloudFormation to create and manage resources programmatically.

Topics

- [Manage quotas with Service Quotas](#)
- [Manage resources with Tags](#)
- [Manage billing and costs with AWS Billing and Cost Management](#)
- [Create resources with AWS CloudFormation](#)

Manage quotas with Service Quotas

Note

If you require a higher quota, you can use the Service Quotas console to [request quota increases](#) for adjustable quotas. When requesting a quota increase, select the Region you require the quota increase in since most quotas are specific to the AWS Region.

Service Quotas console allow you to request quota increases or decrease quota for adjustable quotas. Service Quotas are the maximum number of API call or resources you can have per AWS account and AWS Region. Amazon Location Service denies additional requests that exceed the service quota.

Rate limits (quotas that start with Rate of...) are the maximum number of requests per second, with a burst rate of 80 percent of the limit within any part of the second, defined for each API operation. Amazon Location Service throttles requests that exceed the operation's rate limit.

Managing your Amazon Location service quotas

Amazon Location Service is integrated with Service Quotas, an AWS service that enables you to view and manage your quotas from a central location. For more information, see [What Is Service Quotas?](#) in the *Service Quotas User Guide*.

Service Quotas makes it easy to look up the value of your Amazon Location service quotas.

AWS Management Console

1. Open the [Service Quotas console](#).
2. In the navigation pane, choose **AWS services**.
3. From the **AWS services** list, search for and select **Amazon Location**.
4. In the **Service quotas** list, you can see the service quota name, applied value (if it is available), AWS default quota, and whether the quota value is adjustable.
5. To view additional information about a service quota, such as the description, choose the quota name.
6. (Optional) To request a quota increase, select the quota that you want to increase, select **Request quota increase**, enter or select the required information, and select **Request**.

To work more with service quotas using the console, see the *Service Quotas User Guide*. To request a quota increase, see [Requesting a quota increase](#) in the *Service Quotas User Guide*.

AWS CLI

Run the following command to view the default Amazon Location quotas.

```
aws service-quotas list-aws-default-service-quotas \
  --query 'Quotas[*]'.
{Adjustable:Adjustable,Name:QuotaName,Value:Value,Code:QuotaCode}' \
  --service-code geo \
  --output table
```

To work more with service quotas using the AWS CLI, see the Service Quotas section in the *AWS CLI Command Reference*. To request a quota increase, see [request-service-quota-increase](#) in the *AWS CLI Command Reference*.

FAQ

What are the default quotas for Amazon Location Service resources?

Amazon Location Service sets default quotas for APIs to help manage service capacity, which can be viewed in the [AWS Service Quotas Management Console](#). You can find links to all of these in the [the section called “Monitoring your Amazon Location service quotas”](#) section below.

How can I request an increase in quotas for Amazon Location Service?

You can request an increase in Amazon Location Service quotas through the [self-service console](#), for up to 2x the default limit for each API. For quota limits exceeding 2x the default limit, request in the self service console and it will submit a support ticket. Alternately, you can connect your premium support team

Are there additional costs associated with requesting higher quotas for Amazon Location Service?

There are no direct charges for quota increase requests, but higher usage levels may lead to increased service costs based on the additional resources consumed.

Monitoring your Amazon Location service quotas

You can monitor your usage against your quotas with Amazon CloudWatch. For more information, see [the section called “Monitor with Amazon CloudWatch”](#).

Name	Default	Adjustable	Description
API Key resources per account	Each supported Region: 500	No	The maximum number of API key resources (active or expired) that you can have per account.
Geofence Collection resources per account	Each supported Region: 1,500	Yes	The maximum number of Geofence Collection resources that you can create per account.
Geofences per Geofence Collection	Each supported Region: 50,000	No	The maximum number of Geofences that you can create per Geofence Collection.
Map resources per account	Each supported Region: 40	Yes	The maximum number of Map resources that you can create per account.
Place Index resources per account	Each supported Region: 40	Yes	The maximum number of Place Index resources

Name	Default	Adjustable	Description
			that you can create per account.
Rate of AssociateTrackerConsumer API requests	Each supported Region: 10 per second	Yes	The maximum number of AssociateTrackerConsumer requests that you can make per second. Additional requests are throttled.
Rate of BatchDeleteDevicePositionHistory API requests	Each supported Region: 50 per second	Yes	The maximum number of BatchDeleteDevicePositionHistory requests that you can make per second. Additional requests are throttled.
Rate of BatchDeleteGeofence API requests	Each supported Region: 50 per second	Yes	The maximum number of BatchDeleteGeofence requests that you can make per second. Additional requests are throttled.
Rate of BatchEvaluateGeofences API requests	Each supported Region: 50 per second	Yes	The maximum number of BatchEvaluateGeofences requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of BatchGetDevicePosition API requests	Each supported Region: 50 per second	Yes	The maximum number of BatchGetDevicePosition requests that you can make per second. Additional requests are throttled.
Rate of BatchPutGeofence API requests	Each supported Region: 50 per second	Yes	The maximum number of BatchPutGeofence requests that you can make per second. Additional requests are throttled.
Rate of BatchUpdateDevicePosition API requests	Each supported Region: 50 per second	Yes	The maximum number of BatchUpdateDevicePosition requests that you can make per second. Additional requests are throttled.
Rate of CalculateRoute API requests	Each supported Region: 10 per second	Yes	The maximum number of CalculateRoute requests that you can make per second. Additional requests are throttled.
Rate of CalculateRouteMatrix API requests	Each supported Region: 5 per second	Yes	The maximum number of CalculateRouteMatrix requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of CreateGeofenceCollection API requests	Each supported Region: 10 per second	Yes	The maximum number of CreateGeofenceCollection requests that you can make per second. Additional requests are throttled.
Rate of CreateKey API requests	Each supported Region: 10 per second	Yes	The maximum number of CreateKey requests that you can make per second. Additional requests are throttled.
Rate of CreateMap API requests	Each supported Region: 10 per second	Yes	The maximum number of CreateMap requests that you can make per second. Additional requests are throttled.
Rate of CreatePlaceIndex API requests	Each supported Region: 10 per second	Yes	The maximum number of CreatePlaceIndex requests that you can make per second. Additional requests are throttled.
Rate of CreateRouteCalculator API requests	Each supported Region: 10 per second	Yes	The maximum number of CreateRouteCalculator requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of CreateTracker API requests	Each supported Region: 10 per second	Yes	The maximum number of CreateTracker requests that you can make per second. Additional requests are throttled.
Rate of DeleteGeofenceCollection API requests	Each supported Region: 10 per second	Yes	The maximum number of DeleteGeofenceCollection requests that you can make per second. Additional requests are throttled.
Rate of DeleteKey API requests	Each supported Region: 10 per second	Yes	The maximum number of DeleteKey requests that you can make per second. Additional requests are throttled.
Rate of DeleteMap API requests	Each supported Region: 10 per second	Yes	The maximum number of DeleteMap requests that you can make per second. Additional requests are throttled.
Rate of DeletePlaceIndex API requests	Each supported Region: 10 per second	Yes	The maximum number of DeletePlaceIndex requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of DeleteRouteCalculator API requests	Each supported Region: 10 per second	Yes	The maximum number of DeleteRouteCalculator requests that you can make per second. Additional requests are throttled.
Rate of DeleteTracker API requests	Each supported Region: 10 per second	Yes	The maximum number of DeleteTracker requests that you can make per second. Additional requests are throttled.
Rate of DescribeGeofenceCollection API requests	Each supported Region: 10 per second	Yes	The maximum number of DescribeGeofenceCollection requests that you can make per second. Additional requests are throttled.
Rate of DescribeKey API requests	Each supported Region: 10 per second	Yes	The maximum number of DescribeKey requests that you can make per second. Additional requests are throttled.
Rate of DescribeMap API requests	Each supported Region: 10 per second	Yes	The maximum number of DescribeMap requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of DescribePlaceIndex API requests	Each supported Region: 10 per second	Yes	The maximum number of DescribePlaceIndex requests that you can make per second. Additional requests are throttled.
Rate of DescribeRouteCalculator API requests	Each supported Region: 10 per second	Yes	The maximum number of DescribeRouteCalculator requests that you can make per second. Additional requests are throttled.
Rate of DescribeTracker API requests	Each supported Region: 10 per second	Yes	The maximum number of DescribeTracker requests that you can make per second. Additional requests are throttled.
Rate of DisassociateTrackerConsumer API requests	Each supported Region: 10 per second	Yes	The maximum number of DisassociateTrackerConsumer requests that you can make per second. Additional requests are throttled.
Rate of ForecastGeofenceEvents API requests	Each supported Region: 50 per second	Yes	The maximum number of ForecastGeofenceEvents requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of GetDevicePosition API requests	Each supported Region: 50 per second	Yes	The maximum number of GetDevicePosition requests that you can make per second. Additional requests are throttled.
Rate of GetDevicePositionHistory API requests	Each supported Region: 50 per second	Yes	The maximum number of GetDevicePositionHistory requests that you can make per second. Additional requests are throttled.
Rate of GetGeofence API requests	Each supported Region: 50 per second	Yes	The maximum number of GetGeofence requests that you can make per second. Additional requests are throttled.
Rate of GetMapGlyphs API requests	Each supported Region: 50 per second	Yes	The maximum number of GetMapGlyphs requests that you can make per second. Additional requests are throttled.
Rate of GetMapSprites API requests	Each supported Region: 50 per second	Yes	The maximum number of GetMapSprites requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of GetMapStyleDescriptor API requests	Each supported Region: 50 per second	Yes	The maximum number of GetMapStyleDescriptor requests that you can make per second. Additional requests are throttled.
Rate of GetMapTile API requests	Each supported Region: 500 per second	Yes	The maximum number of GetMapTile requests that you can make per second. Additional requests are throttled.
Rate of GetPlace API requests	Each supported Region: 50 per second	Yes	The maximum number of GetPlace requests that you can make per second. Additional requests are throttled.
Rate of ListDevicePositions API requests	Each supported Region: 50 per second	Yes	The maximum number of ListDevicePositions requests that you can make per second. Additional requests are throttled.
Rate of ListGeofenceCollections API requests	Each supported Region: 10 per second	Yes	The maximum number of ListGeofenceCollections requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of ListGeofences API requests	Each supported Region: 50 per second	Yes	The maximum number of ListGeofences requests that you can make per second. Additional requests are throttled.
Rate of ListKeys API requests	Each supported Region: 10 per second	Yes	The maximum number of ListKeys requests that you can make per second. Additional requests are throttled.
Rate of ListMaps API requests	Each supported Region: 10 per second	Yes	The maximum number of ListMaps requests that you can make per second. Additional requests are throttled.
Rate of ListPlaceIndexes API requests	Each supported Region: 10 per second	Yes	The maximum number of ListPlaceIndexes requests that you can make per second. Additional requests are throttled.
Rate of ListRouteCalculators API requests	Each supported Region: 10 per second	Yes	The maximum number of ListRouteCalculators requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of ListTagsForResource API requests	Each supported Region: 10 per second	Yes	The maximum number of ListTagsForResource requests that you can make per second. Additional requests are throttled.
Rate of ListTrackerConsumers API requests	Each supported Region: 10 per second	Yes	The maximum number of ListTrackerConsumers requests that you can make per second. Additional requests are throttled.
Rate of ListTrackers API requests	Each supported Region: 10 per second	Yes	The maximum number of ListTrackers requests that you can make per second. Additional requests are throttled.
Rate of PutGeofence API requests	Each supported Region: 50 per second	Yes	The maximum number of PutGeofence requests that you can make per second. Additional requests are throttled.
Rate of SearchPlaceIndexForPosition API requests	Each supported Region: 50 per second	Yes	The maximum number of SearchPlaceIndexForPosition requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of SearchPlaceIndexForSuggestions API requests	Each supported Region: 50 per second	Yes	The maximum number of SearchPlaceIndexForSuggestions requests that you can make per second. Additional requests are throttled.
Rate of SearchPlaceIndexForText API requests	Each supported Region: 50 per second	Yes	The maximum number of SearchPlaceIndexForText requests that you can make per second. Additional requests are throttled.
Rate of TagResource API requests	Each supported Region: 10 per second	Yes	The maximum number of TagResource requests that you can make per second. Additional requests are throttled.
Rate of UntagResource API requests	Each supported Region: 10 per second	Yes	The maximum number of UntagResource requests that you can make per second. Additional requests are throttled.
Rate of UpdateGeofenceCollection API requests	Each supported Region: 10 per second	Yes	The maximum number of UpdateGeofenceCollection requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of UpdateKey API requests	Each supported Region: 10 per second	Yes	The maximum number of UpdateKey requests that you can make per second. Additional requests are throttled.
Rate of UpdateMap API requests	Each supported Region: 10 per second	Yes	The maximum number of UpdateMap requests that you can make per second. Additional requests are throttled.
Rate of UpdatePlaceIndex API requests	Each supported Region: 10 per second	Yes	The maximum number of UpdatePlaceIndex requests that you can make per second. Additional requests are throttled.
Rate of UpdateRouteCalculator API requests	Each supported Region: 10 per second	Yes	The maximum number of UpdateRouteCalculator requests that you can make per second. Additional requests are throttled.
Rate of UpdateTracker API requests	Each supported Region: 10 per second	Yes	The maximum number of UpdateTracker requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of VerifyDevicePosition API requests	Each supported Region: 50 per second	Yes	The maximum number of VerifyDevicePosition requests that you can make per second. Additional requests are throttled.
Rate of geo-maps:GetStaticMap API requests	Each supported Region: 50 per second	Yes	The maximum number of geo-maps:GetStaticMap requests that you can make per second. Additional requests are throttled.
Rate of geo-maps:GetTile API requests	Each supported Region: 2,000 per second	Yes	The maximum number of geo-maps:GetTile requests that you can make per second. Additional requests are throttled.
Rate of geo-places:Autocomplete API requests	Each supported Region: 100 per second	Yes	The maximum number of geo-places:Autocomplete requests that you can make per second. Additional requests are throttled.
Rate of geo-places:Geocode API requests	Each supported Region: 100 per second	Yes	The maximum number of geo-places:Geocode requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of geo-places:GetPlace API requests	Each supported Region: 100 per second	Yes	The maximum number of geo-places:GetPlace requests that you can make per second. Additional requests are throttled.
Rate of geo-places:ReverseGeocode API requests	Each supported Region: 100 per second	Yes	The maximum number of geo-places:ReverseGeocode requests that you can make per second. Additional requests are throttled.
Rate of geo-places:SearchNearby API requests	Each supported Region: 100 per second	Yes	The maximum number of geo-places:SearchNearby requests that you can make per second. Additional requests are throttled.
Rate of geo-places:SearchText API requests	Each supported Region: 100 per second	Yes	The maximum number of geo-places:SearchText requests that you can make per second. Additional requests are throttled.
Rate of geo-places:Suggest API requests	Each supported Region: 100 per second	Yes	The maximum number of geo-places:Suggest requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Rate of geo-routes:CalculateSolines API requests	Each supported Region: 20 per second	Yes	The maximum number of geo-routes:CalculateSolines requests that you can make per second. Additional requests are throttled.
Rate of geo-routes:CalculateRouteMatrix API requests	Each supported Region: 5 per second	Yes	The maximum number of geo-routes:CalculateRouteMatrix requests that you can make per second. Additional requests are throttled.
Rate of geo-routes:CalculateRoutes API requests	Each supported Region: 20 per second	Yes	The maximum number of geo-routes:CalculateRoutes requests that you can make per second. Additional requests are throttled.
Rate of geo-routes:OptimizeWaypoints API requests	Each supported Region: 5 per second	Yes	The maximum number of geo-routes:OptimizeWaypoints requests that you can make per second. Additional requests are throttled.
Rate of geo-routes:SnapToRoads API requests	Each supported Region: 20 per second	Yes	The maximum number of geo-routes:SnapToRoads requests that you can make per second. Additional requests are throttled.

Name	Default	Adjustable	Description
Route Calculator resources per account	Each supported Region: 40	Yes	The maximum number of Route Calculator resources that you can create per account.
Tracker consumers per tracker	Each supported Region: 5	No	The maximum number of Geofence Collection that Tracker resource can be associated with.
Tracker resources per account	Each supported Region: 500	Yes	The maximum number of Tracker resources that you can create per account.

Learn more

To learn more about service quotas, see the [Service Quotas documentation](#).

Manage resources with Tags

Use resource tagging in Amazon Location to create tags to categorize your resources by purpose, owner, environment, or criteria. Tagging your resources helps you manage, identify, organize, search, and filter your resources. For example, with AWS Resource Groups, you can create groups of AWS resources based on one or more tags or portions of tags. You can also create groups based on their occurrence in an AWS CloudFormation stack. Using Resource Groups and Tag Editor, you can consolidate and view data for applications that consist of multiple services, resources, and Regions in one place. For more information on [Common Tagging Strategies](#), see the *AWS General Reference*.

Each tag is a label consisting of a key and value that you define:

- **Tag key** – A general label that categorizes the tag values. For example, CostCenter.
- **Tag value** – An optional description for the tag key category. For example, MobileAssetTrackingResourcesProd.

This topic helps you get started with tagging by reviewing tagging restrictions. It also shows you how to create tags and use tags to track your AWS cost for each active tag by using cost allocation reports.

For more information about:

- Tagging best practices, see [Tagging AWS resources](#) in the *AWS General Reference*.
- Using tags to control access to AWS resources, see [Controlling access to AWS resources using tags](#) in the *AWS Identity and Access Management User Guide*.

Restrictions

Note

If you add a new tag with the same tag key as an existing tag, the new tag overwrites the existing tag.

Tagging allows you to organize and manage your resources more effectively. This page outlines the specific rules and constraints that govern the use of tags within Amazon Location Service. By understanding these tagging restrictions, you can ensure compliance with best practices and avoid potential issues when implementing tagging strategies for your location-based resources and applications.

The following basic restrictions apply to tags:

- Maximum tags per resource: 50
- For each resource, each tag key must be unique, and each tag key can have only one value.
- Maximum key length: 128 Unicode characters in UTF-8
- Maximum value length: 256 Unicode characters in UTF-8
- The allowed characters across services are: letters, numbers, and spaces representable in UTF-8, and the following characters: + - = . _ : / @.
- Tag keys and values are case-sensitive.
- The `aws :` prefix is reserved for AWS use. If a tag has a tag key with this prefix, then you can't edit or delete the tag's key or value. Tags with the `aws:` prefix don't count against your tags per resource limit.

Grant permission to tag resources

You can use IAM policies to control access to your Amazon Location resources and grant permission to tag a resource on creation. In addition to granting permission to create resources, the policy can include Action permissions to allow tagging operations:

- `geo:TagResource` – Allows a user to assign one or more tags to a specified Amazon Location resource.
- `geo:UntagResource` – Allows a user to remove one or more tags from a specified Amazon Location resource.
- `geo:ListTagsForResource` – Allows a user to list all the tags assigned to an Amazon Location resource.

The following is a policy example to allow a user to create a geofence collection and tag resources:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowTaggingForGeofenceCollectionOnCreation",
      "Effect": "Allow",
      "Action": [
        "geo:CreateGeofenceCollection",
        "geo:TagResource"
      ],
      "Resource": "arn:aws:geo:region:accountID:geofence-collection/*"
    }
  ]
}
```

Add a tag to a resource

You can add tags when creating your resources using the Amazon Location console, the AWS CLI, or the Amazon Location APIs:

- [the section called “Manage resources”](#)
- [the section called “Create a tracker”](#)

To tag existing resources, edit or delete tags

1. Open the [Amazon Location console](#).
2. In the left navigation pane, choose the resource you want to tag. For example, **Maps**.
3. Choose a resource from the list.
4. Choose **Manage tags** to add, edit, or delete your tags.

How to use tags

You can use tags for cost allocation to track your AWS cost in detail. After you activate the cost allocation tags, AWS uses the cost allocation tags to organize your resource billing on your cost allocation report. This helps you categorize and track your usage costs.

Amazon Location supports [User-defined](#) – Tags. These are custom tags that you create. The user-defined tags use the `user:` prefix, for example, `user:CostCenter`.

You must activate each tag type individually. After your tags are activated, you can [enable AWS Cost Explorer](#) or view your monthly cost allocation report.

To activate user-defined tags

1. Open the [Billing and Cost Management console](#).
2. In the left navigation pane, choose **Cost Allocation Tags**.
3. Under the **User-Defined Cost Allocation Tags** tab, select the tag keys you want to activate.
4. Choose **Activate**.

After you activate your tags, AWS generates a [monthly Cost Allocation Report](#) for your resource usage and cost. This cost allocation report includes all of your AWS costs for each billing period, including tagged and untagged resources. For more information, see [Organizing and tracking costs using AWS cost allocation tags](#) in the *AWS Billing User Guide*.

Control access to resources using tags

AWS Identity and Access Management (IAM) policies support tag-based conditions, which enables you to manage authorization for your resources based on specific tags key and values. For example, an IAM role policy can include conditions to limit access to specific environments, such as development, test, or production, based on tags. For more information, see the topic on [control resource access based on tags](#).

Manage billing and costs with AWS Billing and Cost Management

AWS Billing and Cost Management is a web service that provides features that helps you pay your bills and optimize your costs. Amazon Web Services bills your account for usage, which ensures that you pay only for what you use.

How to see bills and manage cost

1. Open [Billing and Cost Management](#) in the AWS Management Console.
2. Search for location service in Amazon Web Services, Inc. charges by service
3. Choose **[+] Location service**
4. Choose **[+] Region Name**

To learn more, see [Billing and Cost Management](#) in the AWS Management Console.

Create resources with AWS CloudFormation

Amazon Location Service is integrated with AWS CloudFormation, a service that helps you to model and set up your AWS resources so that you can spend less time creating and managing your resources and infrastructure. You create a template that describes all the AWS resources that you want (such as Amazon Location resources), and AWS CloudFormation provisions and configures those resources for you.

When you use AWS CloudFormation, you can reuse your template to set up your Amazon Location resources consistently and repeatedly. Describe your resources once, and then provision the same resources over and over in multiple AWS accounts and Regions.

Related AWS CloudFormation templates

To provision and configure resources for Amazon Location and related services, you must understand [AWS CloudFormation templates](#). Templates are formatted text files in JSON or YAML. These templates describe the resources that you want to provision in your AWS CloudFormation stacks. If you're unfamiliar with JSON or YAML, you can use AWS CloudFormation Designer to help you get started with AWS CloudFormation templates. For more information, see [What is AWS CloudFormation Designer?](#) in the *AWS CloudFormation User Guide*.

Amazon Location supports creating the following resource types in AWS CloudFormation:

- [AWS::Location::Tracker](#)

- [AWS::Location::TrackerConsumer](#)
- [AWS::Location::GeofenceCollection](#)

For more information, including examples of JSON and YAML templates for Amazon Location resources, see the [Amazon Location Service resource type reference](#) in the *AWS CloudFormation User Guide*.

Learn more about AWS CloudFormation

To learn more about AWS CloudFormation, see the following resources:

- [AWS CloudFormation](#)
- [AWS CloudFormation User Guide](#)
- [AWS CloudFormation API Reference](#)
- [AWS CloudFormation Command Line Interface User Guide](#)

Monitoring and Auditing

Monitoring and Auditing provides capabilities to track, monitor, and log activities in your Amazon Location Services environment. With Amazon CloudWatch and AWS CloudTrail, you can ensure the reliability, security, and compliance of your applications.

These tools help you observe resource performance metrics, detect anomalies, and log API activity for auditing and troubleshooting. Use them to enhance operational insights, diagnose issues, and ensure adherence to compliance standards.

Topics

- [Monitor with Amazon CloudWatch](#)
- [Monitor and log with AWS CloudTrail](#)

Monitor with Amazon CloudWatch

Amazon CloudWatch monitors your AWS resources and the applications that you run on AWS in near-real time. You can monitor Amazon Location resources using CloudWatch, which collects raw data and processes metrics into meaningful statistics in near-real time. You can view historical information for up to 15 months, or search metrics to view in the Amazon CloudWatch console

for more perspective on how your application or service is performing. You can also set alarms by defining thresholds, and send notifications or take actions when those thresholds are met.

For more information, see the [Amazon CloudWatch User Guide](#)

Topics

- [Amazon Location Service metrics and dimensions](#)
- [View Amazon Location Service metrics](#)
- [Create CloudWatch alarms for Amazon Location Service metrics](#)
- [Use CloudWatch to monitor usage against quotas](#)
- [CloudWatch metric examples for Amazon Location Service](#)

Amazon Location Service metrics and dimensions

Metrics are time-ordered data points that are exported to CloudWatch. A dimension is a name/value pair that identifies the metric. For more information, see [Using CloudWatch metrics](#) and [CloudWatch dimensions](#) in the Amazon CloudWatch User Guide.

Note

The result is approximate because of the distributed architecture of Amazon Location Service. In most cases, the count should be close to the actual number of API operations being sent.

Amazon Location Service metrics

The following are metrics that Amazon Location Service exports to CloudWatch in the `AWS/Location` namespace.

Metric	Description	Dimensions
CallCount	The number of calls made to a given API endpoint.	OperationName OperationName, ResourceName
	Valid Statistic: Sum	ApiKeyName, OperationName

Metric	Description	Dimensions
	Units: Count	ApiKeyName, OperationName, ResourceName OperationName, OperationVersion OperationName, OperationVersion, ResourceName ApiKeyName, OperationName, OperationVersion ApiKeyName, OperationName, OperationVersion, ResourceName
ErrorCount	The number of error responses from calls made to a given API endpoint. Valid Statistic: Sum Units: Count	OperationName OperationName, ResourceName ApiKeyName, OperationName ApiKeyName, OperationName, ResourceName
SuccessCount	The number of successful calls made to a given API endpoint. Valid Statistic: Sum Units: Count	OperationName OperationName, ResourceName ApiKeyName, OperationName ApiKeyName, OperationName, ResourceName

Metric	Description	Dimensions
CallLatency	<p>The amount of time the operation takes to process and return a response when a call is made to a given API endpoint.</p> <p>Valid Statistic: Average</p> <p>Units: Milliseconds</p>	<p>OperationName</p> <p>OperationName, ResourceName</p> <p>ApiKeyName, OperationName</p> <p>ApiKeyName, OperationName, ResourceName</p>

Amazon Location Service dimensions for metrics

You can use the dimensions in the following table to filter Amazon Location Service metrics.

Dimension	Description
OperationName	Filters Amazon Location metrics for API operation with the specified operation name.
OperationName, ResourceName	Filter Amazon Location metrics for API operation with the specified operation name and resource name.
ApiKeyName, OperationName	Filter Amazon Location metrics for API operation with the specified operation name and using given API key name.
ApiKeyName, OperationName, ResourceName	Filter Amazon Location metrics for API operation with the specified operation name, resource name and using given API key name.
OperationName, OperationVersion	<p>Filters Amazon Location metrics for API operation with the specified operation name.</p> <p>Amazon Location Service standalone Maps, Places, and Routes will be export metric to this dimension.</p>

Dimension	Description
OperationName, OperationVersion, ResourceName	Filter Amazon Location metrics for API operation with the specified operation name, version, and Amazon Location resource name. Amazon Location standalone Maps, Places, and Routes will be export metric to this dimension.
ApiKeyName, OperationName, OperationVersion	Filter Amazon Location metrics for API operation with the specified operation name, version, and using given API key name. Amazon Location standalone Maps, Places, and Routes will be export metric to this dimension.
ApiKeyName, OperationName, OperationVersion, ResourceName	Filter Amazon Location metrics for API operation with the specified operation name, version, resource name and using given API key name. Amazon Location standalone Maps, Places, and Routes will be export metric to this dimension.

View Amazon Location Service metrics

You can view metrics for Amazon Location Service on the Amazon CloudWatch console or by using the Amazon CloudWatch API.

To view metrics using the CloudWatch console

Example

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Metrics**.
3. On the **All metrics tab**, choose the **Location** namespace.
4. Select the type of metric to view.
5. Select the metric and add it to the chart.

For more information, see [View Available Metrics](#) in the *Amazon CloudWatch User Guide*.

Create CloudWatch alarms for Amazon Location Service metrics

You can use CloudWatch to set alarms on your Amazon Location Service metrics. For example, you can create an alarm in CloudWatch to send an email whenever an error count spike occurs.

The following topics give you a high-level overview of how to set alarms using CloudWatch. For detailed instructions, see [Using Alarms](#) in the *Amazon CloudWatch User Guide*.

To set alarms using the CloudWatch console

Example

1. Open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. In the navigation pane, choose **Alarm**.
3. Choose **Create Alarm**.
4. Choose **Select metric**.
5. On the **All metrics** tab, select the **Location** namespace.
6. Select a metric category.
7. Find the row with the metric you want to create an alarm for, then select the check box next to this row.
8. Choose **Select metric**.
9. Under **Metric**, fill in the values.
10. Specify the alarm **Conditions**.
11. Choose **Next**.
12. If you want to send a notification when the alarm conditions are met:
 - Under **Alarm state trigger**, select the alarm state to prompt a notification to be sent.
 - Under **Select an SNS topic**, choose **Create new topic** to create a new Amazon Simple Notification Service (Amazon SNS) topic. Enter the topic name and the email to send the notification to.
 - Under **Send a notification to** enter additional email addresses to send the notification to.
 - Choose **Add notification**. This list is saved and appears in the field for future alarms.
13. When done, choose **Next**.
14. Enter a name and description for the alarm, then choose **Next**.
15. Confirm the alarm details, then choose **Next**.

Note

When creating a new Amazon SNS topic, you must verify the email address before a notification can be sent. If the email is not verified, the notification will not be received when an alarm is initiated by a state change.

For more information about how to set alarms using the CloudWatch console, see [Create an Alarm that Sends Email](#) in the *Amazon CloudWatch User Guide*.

Use CloudWatch to monitor usage against quotas

You can create Amazon CloudWatch alarms to notify you when your utilization of a given quota exceeds a configurable threshold. This enables you to recognize when you are close to your quota limits, and either adapt your utilization to avoid cost overruns, or request a quota increase, if needed.

For information about how to use CloudWatch to monitor quotas, see [Visualizing your service quotas and setting alarms](#) in the *Amazon CloudWatch User Guide*.

CloudWatch metric examples for Amazon Location Service

You can use the [GetMetricData](#) API to retrieve metrics for Amazon Location.

- For example, you can monitor `CallCount` and set an alarm for when a drop in number occurs.

Monitoring the `CallCount` metrics for `SendDeviceLocation` can help give you perspective on tracked assets. If the `CallCount` drops, it means that tracked assets, such as a fleet of trucks, have stopped sending their current locations. Setting an alarm for this can help notify you an issue has occurred.

- For another example, you can monitor `ErrorCount` and set an alarm for when a spike in number occurs.

Trackers must be associated with geofence collections in order for device locations to be evaluated against geofences. If you have a device fleet that requires continuous location updates, seeing the `CallCount` for `BatchEvaluateGeofence` or `BatchPutDevicePosition` drop to zero indicates that updates are no longer flowing.

The following is an example output for [GetMetricData](#) with the metrics for `CallCount` and `ErrorCount` for creating map resources.

```
{
  "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"
      }
    }
  ]
}
```



```
]
}
```

Monitor and log with AWS CloudTrail

AWS CloudTrail is a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail records all API calls as events. You can use Amazon Location Service with CloudTrail to monitor your API calls, which include calls from the Amazon Location Service console and AWS SDK calls to the Amazon Location Service API operations.

CloudTrail is automatically enabled when you create your AWS account. When activity occurs in Amazon Location Service, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download event history for the past 90 days per AWS Region.

For more information about CloudTrail, see the [AWS CloudTrail User Guide](#). There are no CloudTrail charges for viewing the **Event history**.

For an ongoing records of events in your AWS account past 90 days, including events from Amazon Location Service, create a trail or a [CloudTrail Lake](#) data store.

CloudTrail trails

A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. When you create a trail in AWS Management Console, the trail applies to all AWS Regions. The trail logs events from all regions in the AWS Partition and delivers the log files to the S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs.

For more information on how to create a trail, see [Overview for Creating a Trail](#).

For a list of CloudTrail supported services and integrations, see [CloudTrail Supported Services and Integrations](#).

You can deliver one copy of your ongoing management events to your Amazon S3 bucket at no charge from CloudTrail by creating a trail. However, there are Amazon S3 storage charges.

For more information about CloudTrail pricing, see [AWS CloudTrail pricing](#).

For information about Amazon S3 pricing, see [Amazon S3 pricing](#).

CloudTrail Lake event data stores

CloudTrail Lake lets you run SQL-based queries on your events. Events are aggregated into *event data stores*, which are immutable collections of events based on criteria that you select by applying [advanced event selectors](#). The selectors that you apply to an event data store control which events persist and are available for you to query.

For more information about CloudTrail Lake, see [Working with AWS CloudTrail Lake](#).

CloudTrail Lake event data stores and queries incur costs. When you create an event data store, you choose the [pricing option](#) you want to use for the event data store. The pricing option determines the cost for ingesting and storing events, and the default and maximum retention period for the event data store.

For more information about CloudTrail pricing, see [AWS CloudTrail pricing](#).

Topics

- [Amazon Location management events in CloudTrail](#)
- [Amazon Location data events in CloudTrail](#)
- [Learn about Amazon Location Service log file entries](#)
- [Example: CloudTrail log file entry for an Amazon Location management event](#)
- [Example: CloudTrail log file entry for an Amazon Location data event](#)
- [CalculateRouteMatrix examples](#)
- [CalculateRouteMatrix with a geometry-based routing boundary](#)

Amazon Location management events in CloudTrail

You can view Amazon Location management events in your CloudTrail event history. These events include all API calls that manage Amazon Location resources and configurations. For a complete list of supported actions, refer to the [Amazon Location Service API references](#).

Amazon Location data events in CloudTrail

Data events provide information about operations performed directly on a resource. These events, also known as data plane operations, can be high-volume. By default, CloudTrail does not log data events, and the CloudTrail Event History does not record them. You incur additional charges when you enable data events. For more information about CloudTrail pricing, see [AWS CloudTrail Pricing](#).

You can choose which Amazon Location resource types log data events by using the CloudTrail console, AWS CLI, or CloudTrail API operations. For instructions on how to enable and manage data

events, see [Logging data events with the AWS Management Console](#) and [Logging data events with the AWS Command Line Interface](#).

The following table lists the Amazon Location resource types for which you can log data events:

Supported Amazon Location Data Events

Data event type (console)	resources.type value	Data APIs logged to CloudTrail
Geo Maps	AWS::GeoMaps::Provider	See the Amazon GeoMaps API reference
Geo Places	AWS::GeoPlaces::Provider	See the Amazon GeoPlaces API reference
Geo Routes	AWS::GeoRoutes::Provider	See the Amazon GeoRoutes API reference

Note

Amazon Location does not publish CloudTrail events for the following GeoMaps APIs: `GetStyleDescriptor`, `GetGlyphs`, and `GetSprites`. These APIs are free of charge and do not require authentication.

You can configure advanced event selectors to filter events by `eventName`, `readOnly`, and `resources.ARN`. This helps you log only those events that matter to you. For more information, see [AdvancedFieldSelector](#).

Learn about Amazon Location Service log file entries

When you configure a trail, CloudTrail delivers events as log files to an S3 bucket that you specify, or to Amazon CloudWatch Logs. For more information, see [Working with CloudTrail log files](#) in the AWS CloudTrail User Guide.

CloudTrail log files can contain one or more log entries. Each event entry represents a single request from any source and includes details such as the requested operation, the date and time of the operation, request parameters, and more.

Note

CloudTrail log files are not an ordered stack trace of API calls. They do not appear in chronological order. To determine the order of operations, use [eventTime](#).

Every event or log entry contains information about who made the request. This identity information helps you determine:

- Whether the request was made with root or user credentials.
- Whether the request was made with temporary security credentials for a role or a federated user.
- Whether the request was made by another AWS service.

Example: CloudTrail log file entry for an Amazon Location management event

The following example shows a CloudTrail log entry for the `CreateTracker` operation, which creates a tracker resource.

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "111122223333",
    "arn": "arn:aws:geo:us-east-1:111122223333:tracker/ExampleTracker",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "111122223333",
        "arn": "arn:aws:geo:us-east-1:111122223333:tracker/ExampleTracker",
        "accountId": "111122223333",
        "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": "SAMPLE_IP_ADDRESS",
    "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": "111122223333"
}

```

Example: CloudTrail log file entry for an Amazon Location data event

The following example shows a CloudTrail log entry for the Geocode operation, which retrieves coordinates, addresses, and other details about a place.

```

{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "ARO60DU7M35SFGUCGXHMSAMPLE",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/vingu-Isengard",
    "accountId": "111122223333",
    "accessKeyId": "ASIA60DU7M352GLR5CFMSAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "ARO60DU7M35SFGUCGXHMSAMPLE",

```

```
    "arn": "arn:aws:iam::111122223333:role/Admin",
    "accountId": "111122223333",
    "userName": "Admin"
  },
  "attributes": {
    "creationDate": "2024-09-16T14:41:33Z",
    "mfaAuthenticated": "false"
  }
},
"eventTime": "2024-09-16T14:42:16Z",
"eventSource": "geo-places.amazonaws.com",
"eventName": "Geocode",
"awsRegion": "us-west-2",
"sourceIPAddress": "52.94.133.129",
"userAgent": "Amazon CloudFront",
"requestParameters": {
  "Query": "****",
  "Filter": {
    "IncludeCountries": [
      "USA"
    ]
  }
},
"responseElements": null,
"requestID": "1ef7e0b8-c9fc-4a20-80c3-b5340d634c4e",
"eventID": "913d256c-3a9d-40d0-9bdf-705f12c7659f",
"readOnly": true,
"resources": [
  {
    "accountId": "111122223333",
    "type": "AWS::GeoPlaces::Provider",
    "ARN": "arn:aws:geoplaces:us-west-2:111122223333:provider"
  }
],
"eventType": "AwsApiCall",
"managementEvent": false,
"recipientAccountId": "111122223333",
"eventCategory": "Data"
}
```

CalculateRouteMatrix examples

Use the following examples to understand how you can call the `CalculateRouteMatrix` operation with an unbounded routing boundary.

Sample request

```
{
  "Origins": [
    {
      "Position": [-123.11679620827039, 49.28147612192166]
    },
    {
      "Position": [-123.11179620827039, 49.3014761219]
    }
  ],
  "Destinations": [
    {
      "Position": [-123.112317039, 49.28897192166]
    }
  ],
  "DepartureTime": "2024-05-28T21:27:56Z",
  "RoutingBoundary": {
    "Unbounded": true
  }
}
```

Sample response

```
{
  "ErrorCount": 0,
  "RouteMatrix": [
    [
      {
        "Distance": 1907,
        "Duration": 343
      }
    ],
    [
      {
        "Distance": 5629,
        "Duration": 954
      }
    ]
  ]
}
```

```
    ]
  ],
  "RoutingBoundary": {
    "Unbounded": true
  }
}
```

cURL

```
curl --request POST \
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/route-matrix?key=Your_key' \
  --header 'Content-Type: application/json' \
  --data '{
  "Origins": [
    {
      "Position": [-123.11679620827039, 49.28147612192166]
    },
    {
      "Position": [-123.11179620827039, 49.3014761219]
    }
  ],
  "Destinations": [
    {
      "Position": [-123.112317039, 49.28897192166]
    }
  ],
  "DepartureTime": "2024-05-28T21:27:56Z",
  "RoutingBoundary": {
    "Unbounded": true
  }
}'
```

AWS CLI

```
aws geo-routes calculate-route-matrix --key ${YourKey} \
--origins '[{"Position": [-123.11679620827039, 49.28147612192166]}, {"Position": [-123.11179620827039, 49.3014761219]}]' \
--destinations '[{"Position": [-123.11179620827039, 49.28897192166]}]' \
--departure-time "2024-05-28T21:27:56Z" \
--routing-boundary '{"Unbounded": true}'
```


CalculateRouteMatrix with a geometry-based routing boundary

This example shows how you can specify a geometry-based routing boundary when you call `CalculateRouteMatrix`.

Sample request

```
{
  "Origins": [
    {
      "Position": [-123.11679620827039, 49.28147612192166]
    },
    {
      "Position": [-123.11179620827039, 49.3014761219]
    }
  ],
  "Destinations": [
    {
      "Position": [-123.112317039, 49.28897192166]
    }
  ],
  "DepartureTime": "2024-05-28T21:27:56Z",
  "RoutingBoundary": {
    "Geometry": {
      "AutoCircle": {
        "Margin": 10000,
        "MaxRadius": 30000
      }
    }
  }
}
```

Sample response

```
{
  "ErrorCount": 0,
  "RouteMatrix": [
    [
      {
        "Distance": 1907,
        "Duration": 344
      }
    ]
  ],
}
```

```

    [
      {
        "Distance": 5629,
        "Duration": 950
      }
    ],
    "RoutingBoundary": {
      "Geometry": {
        "Circle": {
          "Center": [
            -123.1142962082704,
            49.29147612191083
          ],
          "Radius": 11127
        }
      },
      "Unbounded": false
    }
  }
}

```

cURL

```

curl --request POST \
  --url 'https://routes.geo.eu-central-1.amazonaws.com/v2/route-matrix?key=Your_key' \
  --header 'Content-Type: application/json' \
  --data '{
    "Origins": [
      {
        "Position": [-123.11679620827039, 49.28147612192166]
      },
      {
        "Position": [-123.11179620827039, 49.3014761219]
      }
    ],
    "Destinations": [
      {
        "Position": [-123.112317039, 49.28897192166]
      }
    ],
    "DepartureTime": "2024-05-28T21:27:56Z",
    "RoutingBoundary": {

```

```
    "Geometry": {
      "AutoCircle": {
        "Margin": 10000,
        "MaxRadius": 30000
      }
    }
  }
}'
```

AWS CLI

```
aws geo-routes calculate-route-matrix --key ${YourKey} \
--origins '[{"Position": [-123.11679620827039, 49.28147612192166]}, {"Position": [-123.11179620827039, 49.3014761219]}]' \
--destinations '[{"Position": [-123.11179620827039, 49.28897192166]}]' \
--departure-time "2024-05-28T21:27:56Z" \
--routing-boundary '{"Geometry": {"AutoCircle": {"Margin": 10000, "MaxRadius": 30000}}}'
```

Best practices

The following are a few best practices for integrating with Amazon Location Service.

Resource management

To help effectively manage your location resources in Amazon Location Service, consider the following best practices:

- Use regional endpoints that are central to your expected user base to improve their experience. For information about region endpoints, see [the section called “Supported regions”](#).
- For resources that use data providers, such as map resources and place index resources, make sure to follow the terms of use agreement of the specific data provider. For more information, see [the section called “Terms of use and data attribution”](#).
- Minimize the creation of resources by having one resource for each configuration of map, place index, or routes. Within a region, you typically need only one resource per data provider or map style. Most applications use existing resources, and do not create resources at run time.
- When using different resources in a single application, such as a map resource and a route calculator, use the same data provider in each resource to ensure that the data matches. For

example, that a route geometry you create with your route calculator aligns with the streets on the map drawn using the map resource.

Billing and cost management

To help manage your costs and billing, consider the following best practice:

- Use monitoring tools, such as Amazon CloudWatch, to track your resource usage. You can set alerts that notify you when usage is about to exceed your specified limits. For more information, see [Creating a Billing Alarm to Monitor Your Estimated AWS Charges](#) in the *Amazon CloudWatch User Guide*.

Quotas and usage

Your AWS account includes quotas that set a default limit your usage amount. You can set up alarms to alert you when your usage is getting close to your limit, and you can request a raise to a quota, when you need it. For information about how to work with quotas, see the following topics.

- [the section called "Manage quotas"](#)
- [the section called "Create CloudWatch alarms"](#)
- [Visualizing your service quotas and setting alarms](#) in the *Amazon CloudWatch User Guide*.

You can create alarms to give you advance warning when you are close to exceeding your limits. We recommend setting alarms for each quota in each AWS Region where you use Amazon Location. For example, you can monitor your use of the `SearchPlaceIndexForText` operation, and create an alarm when you exceed 80 percent of your current quota.

When you get an alarm warning about your quota, you must decide what to do. You might be using additional resources because your customer base has grown. In that case you may want to request an increase to your quota, such as a 50 percent increase in the quota for an API call in that Region. Or, maybe there's an error in your service that causes you to make additional unnecessary calls to Amazon Location. In that case you'd want to solve the problem in your service.

Security in Amazon Location Service

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS Compliance Programs](#). To learn about the compliance programs that apply to Amazon Location Service, see [AWS Services in Scope by Compliance Program](#).
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon Location. The following topics show you how to configure Amazon Location to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon Location resources.

Topics

- [Data protection in Amazon Location Service](#)
- [Incident Response in Amazon Location Service](#)
- [Compliance validation for Amazon Location Service](#)
- [Resilience in Amazon Location Service](#)
- [Infrastructure security in Amazon Location Service](#)
- [Configuration and vulnerability analysis in Amazon Location](#)
- [Cross-service confused deputy prevention](#)
- [Best practices for Amazon Location Service](#)

Data protection in Amazon Location Service

The AWS [shared responsibility model](#) applies to data protection in Amazon Location Service. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see [Working with CloudTrail trails](#) in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-3](#).

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with Amazon Location or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Data privacy

With Amazon Location Service, you retain control of your organization's data. Amazon Location anonymizes all queries sent to data providers by removing customer metadata and account information.

Amazon Location doesn't use data providers for tracking and geofencing. This means your sensitive data remains in your AWS account. This helps shield sensitive location information, such as facility, asset, and personnel location, from third parties, protect user privacy, and reduce your application's security risk.

For additional information, see the [AWS Data Privacy FAQ](#).

Data retention in Amazon Location

The following characteristics relate to how Amazon Location collects and stores data for the service:

- **Amazon Location Service Trackers** – When you use the Trackers APIs to track the location of entities, their coordinates can be stored. Device locations are stored for 30 days before being deleted by the service.
- **Amazon Location Service Geofences** – When you use the Geofences APIs to define areas of interest, the service stores the geometries you provided. They must be explicitly deleted.

Note

Deleting your AWS account delete all resources within it. For additional information, see the [AWS Data Privacy FAQ](#).

Data encryption at rest for Amazon Location Service

Amazon Location Service provides encryption by default to protect sensitive customer data at rest using AWS owned encryption keys.

- **AWS owned keys** — Amazon Location uses these keys by default to automatically encrypt personally identifiable data. You can't view, manage, or use AWS owned keys, or audit their use. However, you don't have to take any action or change any programs to protect the keys that

encrypt your data. For more information, see [AWS owned keys](#) in the *AWS Key Management Service Developer Guide*.

Encryption of data at rest by default helps reduce the operational overhead and complexity involved in protecting sensitive data. At the same time, it enables you to build secure applications that meet strict encryption compliance and regulatory requirements.

While you can't disable this layer of encryption or select an alternate encryption type, you can add a second layer of encryption over the existing AWS owned encryption keys by choosing a customer managed key when you create your tracker and geofence collection resources:

- **Customer managed keys** — Amazon Location supports the use of a symmetric customer managed key that you create, own, and manage to add a second layer of encryption over the existing AWS owned encryption. Because you have full control of this layer of encryption, you can perform such tasks as:
 - Establishing and maintaining key policies
 - Establishing and maintaining IAM policies and grants
 - Enabling and disabling key policies
 - Rotating key cryptographic material
 - Adding tags
 - Creating key aliases
 - Scheduling keys for deletion

For more information, see [customer managed key](#) in the *AWS Key Management Service Developer Guide*.

The following table summarizes how Amazon Location encrypts personally identifiable data.

Data type	AWS owned key encryption	Customer managed key encryption (Optional)
Position	Enabled	Enabled
A point geometry containing the device position details .		

Data type	AWS owned key encryption	Customer managed key encryption (Optional)
PositionProperties A set of key-value pairs associated with the position update .	Enabled	Enabled
GeofenceGeometry A polygon geofence geometry representing the geofenced area.	Enabled	Enabled
DeviceId The device identifier specified when uploading a device position update to a tracker resource.	Enabled	Not supported
GeofenceId An identifier specified when storing a geofence geometry , or a batch of geofences in a given geofence collection.	Enabled	Not supported

Note

Amazon Location automatically enables encryption at rest using AWS owned keys to protect personally identifiable data at no charge.

However, AWS KMS charges apply for using a customer managed key. For more information about pricing, see the [AWS Key Management Service pricing](#).

For more information on AWS KMS, see [What is AWS Key Management Service?](#)

How Amazon Location Service uses grants in AWS KMS

Amazon Location requires a [grant](#) to use your customer managed key.

When you create a [tracker resource](#) or [geofence collection](#) encrypted with a customer managed key, Amazon Location creates a grant on your behalf by sending a [CreateGrant](#) request to AWS KMS. Grants in AWS KMS are used to give Amazon Location access to a KMS key in a customer account.

Amazon Location requires the grant to use your customer managed key for the following internal operations:

- Send [DescribeKey](#) requests to AWS KMS to verify that the symmetric customer managed KMS key ID entered when creating a tracker or geofence collection is valid.
- Send [GenerateDataKeyWithoutPlaintext](#) requests to AWS KMS to generate data keys encrypted by your customer managed key.
- Send [Decrypt](#) requests to AWS KMS to decrypt the encrypted data keys so that they can be used to encrypt your data.

You can revoke access to the grant, or remove the service's access to the customer managed key at any time. If you do, Amazon Location won't be able to access any of the data encrypted by the customer managed key, which affects operations that are dependent on that data. For example, if you attempt to [get device positions](#) from an encrypted tracker that Amazon Location can't access, then the operation would return an `AccessDeniedException` error.

Create a customer managed key

You can create a symmetric customer managed key by using the AWS Management Console, or the AWS KMS APIs.

To create a symmetric customer managed key

Follow the steps for [Creating symmetric customer managed key](#) in the *AWS Key Management Service Developer Guide*.

Key policy

Key policies control access to your customer managed key. Every customer managed key must have exactly one key policy, which contains statements that determine who can use the key and how

they can use it. When you create your customer managed key, you can specify a key policy. For more information, see [Managing access to customer managed keys](#) in the *AWS Key Management Service Developer Guide*.

To use your customer managed key with your Amazon Location resources, the following API operations must be permitted in the key policy:

- [kms:CreateGrant](#) – Adds a grant to a customer managed key. Grants control access to a specified KMS key, which allows access to [grant operations](#) Amazon Location requires. For more information about [Using Grants](#), see the *AWS Key Management Service Developer Guide*.

This allows Amazon Location to do the following:

- Call `GenerateDataKeyWithoutPlainText` to generate an encrypted data key and store it, because the data key isn't immediately used to encrypt.
- Call `Decrypt` to use the stored encrypted data key to access encrypted data.
- Set up a retiring principal to allow the service to `RetireGrant`.
- [kms:DescribeKey](#) – Provides the customer managed key details to allow Amazon Location to validate the key.

The following are policy statement examples you can add for Amazon Location:

```
"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" : "*"
}
]
```

For more information about [specifying permissions in a policy](#), see the *AWS Key Management Service Developer Guide*.

For more information about [troubleshooting key access](#), see the *AWS Key Management Service Developer Guide*.

Specifying a customer managed key for Amazon Location

You can specify a customer managed key as a second layer encryption for the following resources:

- [the section called "Create a tracker"](#)
- [the section called "Get started"](#)

When you create a resource, you can specify the data key by entering a **KMS ID**, which Amazon Location uses to encrypt the identifiable personal data stored by the resource.

- **KMS ID** — A [key identifier](#) for an AWS KMS customer managed key. Enter a key ID, key ARN, alias name, or alias ARN.

Amazon Location Service encryption context

An [encryption context](#) is an optional set of key-value pairs that contain additional contextual information about the data.

AWS KMS uses the encryption context as [additional authenticated data](#) to support [authenticated encryption](#). When you include an encryption context in a request to encrypt data, AWS KMS binds the encryption context to the encrypted data. To decrypt data, you include the same encryption context in the request.

Amazon Location Service encryption context

Amazon Location uses the same encryption context in all AWS KMS cryptographic operations, where the key is `aws:geo:arn` and the value is the resource [Amazon Resource Name](#) (ARN).

Example

```
"encryptionContext": {
  "aws:geo:arn": "arn:aws:geo:us-west-2:111122223333:geofence-collection/SAMPLE-GeofenceCollection"
}
```

Using encryption context for monitoring

When you use a symmetric customer managed key to encrypt your tracker or geofence collection, you can also use the encryption context in audit records and logs to identify how the customer managed key is being used. The encryption context also appears in [logs generated by AWS CloudTrail or Amazon CloudWatch Logs](#).

Using encryption context to control access to your customer managed key

You can use the encryption context in key policies and IAM policies as conditions to control access to your symmetric customer managed key. You can also use encryption context constraints in a grant.

Amazon Location uses an encryption context constraint in grants to control access to the customer managed key in your account or region. The grant constraint requires that the operations that the grant allows use the specified encryption context.

Example

The following are example key policy statements to grant access to a customer managed key for a specific encryption context. The condition in this policy statement requires that the grants have an encryption context constraint that specifies the encryption context.

```
{
  "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"
    }
  }
}
```

Monitoring your encryption keys for Amazon Location Service

When you use an AWS KMS customer managed key with your Amazon Location Service resources, you can use [AWS CloudTrail](#) or [Amazon CloudWatch Logs](#) to track requests that Amazon Location sends to AWS KMS.

The following examples are AWS CloudTrail events for `CreateGrant`, `GenerateDataKeyWithoutPlainText`, `Decrypt`, and `DescribeKey` to monitor KMS operations called by Amazon Location to access data encrypted by your customer managed key:

CreateGrant

When you use an AWS KMS customer managed key to encrypt your tracker or geofence collection resources, Amazon Location sends a CreateGrant request on your behalf to access the KMS key in your AWS account. The grant that Amazon Location creates are specific to the resource associated with the AWS KMS customer managed key. In addition, Amazon Location uses the RetireGrant operation to remove a grant when you delete a resource.

The following example event records the CreateGrant operation:

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAIQDTESTANDEXAMPLE:Sampleuser01",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAIQDTESTANDEXAMPLE: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

When you enable an AWS KMS customer managed key for your tracker or geofence collection resource, Amazon Location creates a unique table key. It sends a `GenerateDataKeyWithoutPlainText` request to AWS KMS that specifies the AWS KMS customer managed key for the resource.

The following example event records the `GenerateDataKeyWithoutPlainText` operation:

```

{
    "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

When you access an encrypted tracker or geofence collection, Amazon Location calls the Decrypt operation to use the stored encrypted data key to access the encrypted data.

The following example event records the Decrypt operation:

```

{
  "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 uses the `DescribeKey` operation to verify if the AWS KMS customer managed key associated with your tracker or geofence collection exists in the account and region.

The following example event records the DescribeKey operation:

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAIQDTESTANDEXAMPLE:Sampleuser01",
    "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAIQDTESTANDEXAMPLE: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"  
}
```

Learn more

The following resources provide more information about data encryption at rest.

- For more information about [AWS Key Management Service basic concepts](#), see the *AWS Key Management Service Developer Guide*.
- For more information about [Security best practices for AWS Key Management Service](#), see the *AWS Key Management Service Developer Guide*.

Data in transit encryption for Amazon Location Service

Amazon Location protects data in transit, as it travels to and from the service, by automatically encrypting all inter-network data using the Transport Layer Security (TLS) 1.2 encryption protocol. Direct HTTPS requests sent to the Amazon Location Service APIs are signed by using the [AWS Signature Version 4 Algorithm](#) to establish a secure connection.

Incident Response in Amazon Location Service

Security is the highest priority at AWS. As part of the AWS Cloud [shared responsibility model](#), AWS manages a data center and network architecture that meets the requirements of the most security-sensitive organizations. As an AWS customer, you share a responsibility for maintaining security in the cloud. This means you control the security you choose to implement from the AWS tools and features you have access to.

By establishing a security baseline that meets the objectives for your applications running in the cloud, you're able to detect deviations that you can respond to. Since security incident response can be a complex topic, we encourage you to review the following resources so that you are better able to understand the impact that incident response (IR) and your choices have on your corporate

goals: [AWS Security Incident Response Guide](#), [AWS Security Best Practices](#) whitepaper, and the [AWS Cloud Adoption Framework \(AWS CAF\)](#).

Logging and Monitoring in Amazon Location Service

Logging and monitoring are an important part of incident response. It lets you establish a security baseline to detect deviations that you can investigate and respond to. By implementing logging and monitoring for Amazon Location Service, you're able to maintain the reliability, availability, and performance for your projects and resources.

AWS provides several tools that can help you log and collect data for incident response:

AWS CloudTrail

Amazon Location Service integrates with AWS CloudTrail, which is a service that provides a record of actions taken by a user, role or AWS service. This includes actions from the Amazon Location Service console, and programmatic calls to Amazon Location API operations. These records of action are called events. For more information, see [Logging and monitoring Amazon Location Service with AWS CloudTrail](#).

Amazon CloudWatch

You can use Amazon CloudWatch to collect and analyze metrics related to your Amazon Location Service account. You can enable CloudWatch alarms to notify you if a metric meets certain conditions, and has reached a specified threshold. When you create an alarm, CloudWatch sends a notification to an Amazon Simple Notification Service that you define. For more information, see the [Monitoring Amazon Location Service with Amazon CloudWatch](#).

AWS Health Dashboards

Using [AWS Health Dashboards](#), you can verify the status of the Amazon Location Service service. You can also monitor and view historical data about any events or issues that might affect your AWS environment. For more information, see the [AWS Health User Guide](#).


Compliance validation for Amazon Location Service

To learn whether an AWS service is within the scope of specific compliance programs, see [AWS services in Scope by Compliance Program](#) and choose the compliance program that you are interested in. For general information, see [AWS Compliance Programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security Compliance & Governance](#) – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- [Architecting for HIPAA Security and Compliance on Amazon Web Services](#) – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

 **Note**

Not all AWS services are HIPAA eligible. For more information, see the [HIPAA Eligible Services Reference](#).

- [AWS Compliance Resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [AWS Customer Compliance Guides](#) – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- [Evaluating Resources with Rules](#) in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see [Security Hub controls reference](#).
- [Amazon GuardDuty](#) – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.
- [AWS Audit Manager](#) – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience in Amazon Location Service

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

In addition to the AWS global infrastructure, Amazon Location offers several features to help support your data resiliency and backup needs.

Infrastructure security in Amazon Location Service

As a managed service, Amazon Location Service is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see [AWS Cloud Security](#). To design your AWS environment using the best practices for infrastructure security, see [Infrastructure Protection](#) in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Amazon Location through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

Configuration and vulnerability analysis in Amazon Location

Configuration and IT controls are a shared responsibility between AWS and you, our customer. For more information, see the AWS [shared responsibility model](#).

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

Amazon Location Service does not act as a calling service on your behalf to other AWS services, so you do not need to add these protections in this case. To learn more about confused deputy, see [The confused deputy problem](#) in the *AWS Identity and Access Management User Guide*.

Best practices for Amazon Location Service

This topic provides best practices to help you use Amazon Location Service. While these best practices can help you take full advantage of the Amazon Location Service, they do not represent a complete solution. You should follow only the recommendations that are applicable for your environment.

Topics

- [Security](#)

Security

To help manage or even avoid security risks, consider the following best practices:

- Use identity federation and IAM roles to manage, control, or limit access to your Amazon Location resources. For more information, see [IAM Best Practices](#) in the *IAM User Guide*.
- Follow the Principle of Least Privilege to grant only the minimum required access to your Amazon Location Service resources.
- For Amazon Location Service resources used in web applications, restrict access using an `aws:referer` IAM condition, limiting use by sites other than those included in the allow-list.

- Use monitoring and logging tools to track resource access and usage. For more information, see [the section called “Logging and Monitoring”](#) and [Logging Data Events for Trails](#) in the AWS CloudTrail User Guide.
- Use secure connections, such as those that begin with `https://` to add security and protect users against attacks while data is being transmitted between the server and browser.

Detective security best practices for Amazon Location Service

The following best practices for Amazon Location Service can help detect security incidents:

Implement AWS monitoring tools

Monitoring is critical to incident response and maintains the reliability and security of Amazon Location Service resources and your solutions. You can implement monitoring tools from the several tools and services available through AWS to monitor your resources and your other AWS services.

For example, Amazon CloudWatch allows you to monitor metrics for Amazon Location Service and enables you to setup alarms to notify you if a metric meets certain conditions you've set and has reached a threshold you've defined. When you create an alarm, you can set CloudWatch to send a notification to alert using Amazon Simple Notification Service. For more information, see [the section called “Logging and Monitoring”](#).

Enable AWS logging tools

Logging provides a record of actions taken by a user, role or an AWS service in Amazon Location Service. You can implement logging tools such as AWS CloudTrail to collect data on actions to detect unusual API activity.

When you create a trail, you can configure CloudTrail to log events. Events are records of resource operations performed on or within a resource such as the request made to Amazon Location, the IP address from which the request was made, who made the request, when the request was made, along with additional data. For more information, see [Logging Data Events for Trails](#) in the AWS CloudTrail User Guide.

Preventive security best practices for Amazon Location Service

The following best practices for Amazon Location Service can help prevent security incidents:

Use secure connections

Always use encrypted connections, such as those that begin with `https://` to keep sensitive information secure in transit.

Implement least privilege access to resources

When you create custom policies to Amazon Location resources, grant only the permissions required to perform a task. It's recommended to start with a minimum set of permissions and grant additional permissions as needed. Implementing least privilege access is essential to reducing the risk and impact that could result from errors or malicious attacks. For more information, see [the section called "Use IAM"](#).

Use globally-unique IDs as device IDs

Use the following conventions for device IDs.

- Device IDs must be unique.
- Device IDs should not be secret, because they can be used as foreign keys to other systems.
- Device IDs should not contain personally-identifiable information (PII), such as phone device IDs or email addresses.
- Device IDs should not be predictable. Opaque identifiers like UUIDs are recommended.

Do not include PII in device position properties

When sending device updates (for example, using [DevicePositionUpdate](#)), do not include personally-identifiable information (PII) such as phone number or email address in the `PositionProperties`.

Document history

The following table describes the documentation for Amazon Location Service. For notification about updates you can subscribe to an RSS feed.

Change	Description	Date
Amazon Location Service releases enhanced version to general availability	Amazon Location Service now offers enhanced Places, Routes, and Maps functionality, enabling developers to add advanced location capabilities into their applications more easily. These improvements introduce new capabilities and a new streamlined developer experience to support location-based use cases across industries such as healthcare, transportation & logistics, and retail. For more information, see the release .	October 31, 2024

AWS Glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS Glossary Reference*.