Qualcomm Linux Location Guide

80-70018-25 AA

April 8, 2025

Contents

1 Location overview

Location represents a geographical point primarily defined by its latitude, longitude, timestamp, and accuracy. It may also include additional information such as bearing, altitude, and velocity.

Qualcomm® Linux® offers location features, APIs, and a test application to assist you in developing location-based applications on Qualcomm® RB3 Gen 2 Development Kit. It also provides logging capabilities to debug issues related to the location API. You can get started by using the location API test application.

Note: Location APIs that are not described in this guide are reserved for a future release. It is recommended not to invoke these APIs until further notice.

Note: See Hardware SoCs that are supported on Qualcomm Linux.

Note: The location subsystem is currently unavailable on QCS9075/QCS8275. Location functionality will be added in a future release. Location documentation should not be considered accurate for QCS9075/QCS8275 and will be updated in a future release.

1.1 Next steps

- Get started with location
- Run location API test application
- Sample codes for location API functions
- Debug location issues

2 Get started with location

Before you begin, see *Qualcomm Linux Build Guide* for common infrastructure setup and build workflows.

Qualcomm Linux provides a location API test application to run test scenarios for location single shot, tracking, batching, and geofencing sessions. The `location_client_api_testapp` is available at the `/home/root` directory of the device.

The following figure shows the workflow to get started with the location API test application.

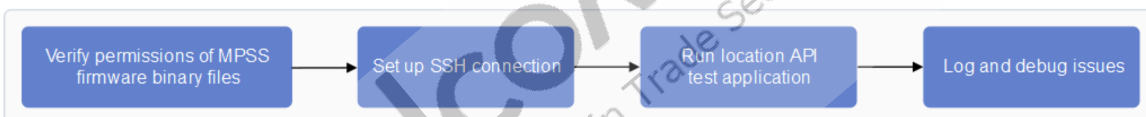

Figure1 Qualcomm Linux Location workflow

2.1 Verify permissions of MPSS firmware binary files

Prerequisites:

- The MPSS firmware binary files must have the correct permissions.
- The host PC used for the builds must have the correct umask setting.

To ensure that the location service is running in MPSS, do the following:

1. Ensure that the host PC used for the builds has the correct umask setting.

   For example:

   - Correct umask setting: `0002`, `0022`
   - Incorrect umask setting: `0027`

2. List the files available at `/usr/lib/modem_pr/so`.

```
mount -o remount rw /
ls -al /usr/lib/modem_pr/so/
-rwxr-x---. 1 root root 1547348 Mar  9  2018 544_0_0.mbn
```

```
-rwxr-x---. 1 root root 1209088 Mar  9  2018 544_0_5.mbn
-rwxr-x---. 1 root root 1551444 Mar  9  2018 544_0_8.mbn
-rwxr-x---. 1 root root  454580 Mar  9  2018 645_0_9.mbn
-rwxr-x---. 1 root root  456000 Mar  9  2018 645_0_a.mbn
-rwxr-x---. 1 root root   22148 Mar  9  2018 828_0.mbn
-rwxr-x---. 1 root root   22148 Mar  9  2018 829_0.mbn
-rwxr-x---. 1 root root  414132 Mar  9  2018 849_0_0.mbn
-rwxr-x---. 1 root root  426332 Mar  9  2018 849_0_2.mbn
-rwxr-x---. 1 root root  448224 Mar  9  2018 850_0_0.mbn
-rwxr-x---. 1 root root   22148 Mar  9  2018 881_0.mbn
-rwxr-x---. 1 root root   35888 Mar  9  2018 889_0_0.mbn
-rwxr-x---. 1 root root 1164128 Mar  9  2018 901_0_0.mbn
-rwxr-x---. 1 root root 1231824 Mar  9  2018 901_0_1.mbn
-rwxr-x---. 1 root root 1231840 Mar  9  2018 901_0_2.mbn
-rwxr-x---. 1 root root  837444 Mar  9  2018 931_0_0.mbn
-rwxr-x---. 1 root root   22148 Mar  9  2018 934_0.mbn
```

3. Provide read `r` permissions to these files.

```
chmod 644 /usr/lib/modem_pr/so/*
ls -al /usr/lib/modem_pr/so/*
-rw-r--r--. 1 root root 1547348 Mar  9  2018 /usr/lib/modem_pr/
so/544_0_0.mbn
-rw-r--r--. 1 root root 1209088 Mar  9  2018 /usr/lib/modem_pr/
so/544_0_5.mbn
-rw-r--r--. 1 root root 1551444 Mar  9  2018 /usr/lib/modem_pr/
so/544_0_8.mbn
-rw-r--r--. 1 root root  454580 Mar  9  2018 /usr/lib/modem_pr/
so/645_0_9.mbn
-rw-r--r--. 1 root root  456000 Mar  9  2018 /usr/lib/modem_pr/
so/645_0_a.mbn
-rw-r--r--. 1 root root   22148 Mar  9  2018 /usr/lib/modem_pr/
so/828_0.mbn
-rw-r--r--. 1 root root   22148 Mar  9  2018 /usr/lib/modem_pr/
so/829_0.mbn
-rw-r--r--. 1 root root  414132 Mar  9  2018 /usr/lib/modem_pr/
so/849_0_0.mbn
-rw-r--r--. 1 root root  426332 Mar  9  2018 /usr/lib/modem_pr/
so/849_0_2.mbn
-rw-r--r--. 1 root root  448224 Mar  9  2018 /usr/lib/modem_pr/
so/850_0_0.mbn
-rw-r--r--. 1 root root   22148 Mar  9  2018 /usr/lib/modem_pr/
so/881_0.mbn
-rw-r--r--. 1 root root   35888 Mar  9  2018 /usr/lib/modem_pr/
```

```
so/889_0_0.mbn
-rw-r--r--. 1 root root 1164128 Mar  9  2018 /usr/lib/modem_pr/
so/901_0_0.mbn
-rw-r--r--. 1 root root 1231824 Mar  9  2018 /usr/lib/modem_pr/
so/901_0_1.mbn
-rw-r--r--. 1 root root 1231840 Mar  9  2018 /usr/lib/modem_pr/
so/901_0_2.mbn
-rw-r--r--. 1 root root  837444 Mar  9  2018 /usr/lib/modem_pr/
so/931_0_0.mbn
-rw-r--r--. 1 root root   22148 Mar  9  2018 /usr/lib/modem_pr/
so/934_0.mbn
```

4. Reboot the device.

2.2 Set up SSH connection

To enable SSH and connect to the device, do the following:

1. Perform the steps mentioned in Sign in using SSH to enable SSH.
2. Connect to the device.

```
ssh root@<device_IP_address>
```

For example:

```
ssh root@10.92.168.185
```

3. Enter the following password to connect to SSH.

```
oelinux123
```


2.3 Next steps

- Run location test API application
- Sample codes for location API functions
- Debug location issues

3 Location features

Location API allows you to access the location information of a device and get location updates. Features available through location API are as follows:

- Single shot fix
- Location tracking
- Location batching
- Location geofencing

Note: Location features may vary based on the hardware and software capabilities of the device.

Before requesting location updates, consider the use cases, accuracy, and frequency requirements for these updates. Consider the following use cases and their requirements.

Use case	Requirement
Navigation applications	Require fine location information of the device every second.
Weather or camera applications	Require a one-time location update with a coarse level of position accuracy.
Applications that access device location history	May not require real time position reporting.

3.1 Single shot fix

You can request a single fix from a location service using the getSinglePosition API function. Based on the preferred Quality of Service (QoS) requirements such as request timeout (`timeoutMsec`) and accuracy (`horQos`), and available location technologies, the system invokes the single shot request.

Note: In this release, only GNSS technology is used to serve single shot location requests.

A position report is returned to the application through `positionCallback` when the accuracy threshold is reached or until the number of seconds specified in the timeout parameter elapses.

3.2 Location tracking

A location tracking session is used to request and obtain a continuous stream of position fixes from GNSS technologies. This type of session is commonly used in map services.

You can start a location tracking session using the startPositionSession (basic location information) API function. The time between fixes (TBF) specified in the `intervalInMs` parameter is used to set the location report rate. Consider the following conditions.

If	Then
The `intervalInMs` parameter is not set.	The default value of 1000 ms is used and the position report, if available, is reported every second.
The highest report rate supported by the system is 10 Hz.	The minimum `intervalInMs` is 100 ms.
The device supports only 1 Hz but the requested rate is 10 Hz.	The position is reported at a maximum rate that the device can support. In this case, the position is reported at 1 Hz.

Consider the following recommendations:

- To achieve good quality position information, set TBF < 5 s.
- If TBF > 30 s, request a single shot fix.
- If real time location updates are not mandatory, request a location batching session.

You can stop a location tracking session using the stopPositionSession API function. The tracking session continues to run the GNSS engine until the application sends a stop session request.

3.3 Location batching

Location batching feature stores the position fixes in the system without notifying the application for each position fix. This feature significantly reduces GNSS power consumption compared to a regular tracking session.

In a batching session, the GNSS engine generates position fixes at the TBF rate but stores each fix in its internal buffer. When the batch buffer is full (batch size of 20), all the batched positions are reported to the application through `batchingCallback`. A batching session is ideal in use cases where position accuracy is required but is not time sensitive, for example, recording fitness activity or tracking device location history.

You can start a batching session using the startRoutineBatchingSession API function. The TBF is specified in the `minInterval` parameter. If this parameter is not set, the default value of 1000 ms is used and passed to location HAL daemon as batching session parameter.

You can stop a batching session using the stopPositionSession API function.

3.4 Location geofencing

A geofence is a virtual perimeter on a geographic area using a location-based service. An application can define a geofence area and monitor the breach events for the area. When the device enters or exits the geofence area, a notification of the breach event is generated and sent to the application.

Location geofence feature is typically used to get breach event notification when the device enters or exits that special area (or inside/outside). Only circular geofences and up to 20 multiple geofence areas are supported.

You can add a circular location geofence using the addGeofences API function. After booting up the system, you must add the geofence again as the geofence area list is not stored in the system during the power cycle.

You can use the removeGeofences, modifyGeofences, pauseGeofences, and resumeGeofences API functions to remove, modify, pause, and resume geofences, respectively.

Consider the following terms related to geofencing:

Term	Description
Geofence area	• A circular geofence area is defined by latitude, longitude, and radius. • The minimum radius of a geofence area should be 50 m.
Breach event	An event when the device enters or exits (inside or outside) an area.
Breach confidence	• The probability of a breach event occurring at the exact geofence boundary for a given geofence breach. • The higher the confidence, the lower the false breach notifications, and vice versa.

Term	Description
Breach responsiveness	• The latency in breach detection by the location software. • Lower latency indicates higher responsiveness and vice versa.
Low-power geofencing	• In general, higher breach confidence and responsiveness can result in increased power consumption. • Low-power geofencing provides higher breach confidence and responsiveness at lower power due to close integration with the GNSS receiver.

3.5 Next steps

- Run location API test application
- Sample codes for location API functions

4 Location architecture

The following architecture diagram shows how a client application uses the location API with location HAL daemon. The location HAL daemon serves as a remote service for client applications that use tracking, batching, or geofencing functionalities.

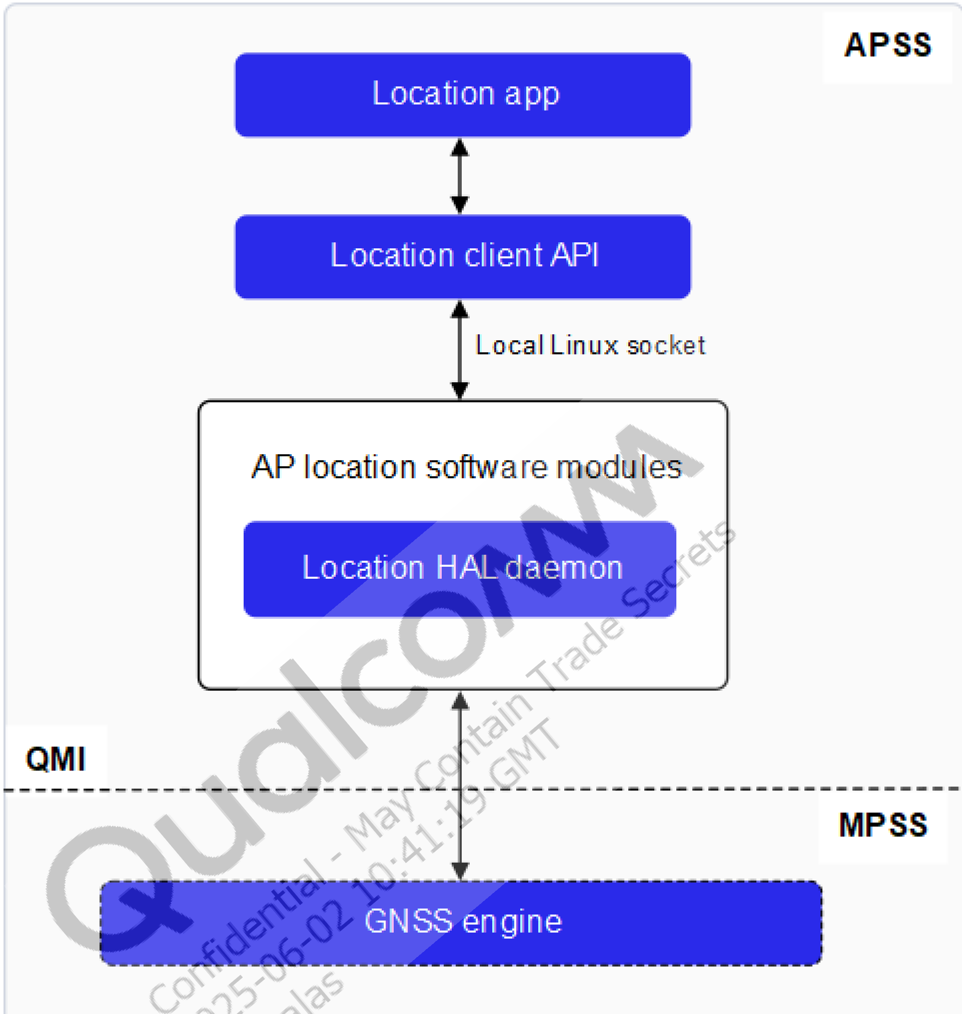

Figure1 Location API high-level architecture

4.1 Asynchronous messaging paradigm

Location APIs are designed specifically for asynchronous messaging, where API processing status and results are delivered asynchronously.

In general, most APIs return either `TRUE` or `FALSE`. While returning `FALSE`, a response callback or result callback is not invoked. However, while returning `TRUE`, an asynchronous response callback is invoked to deliver the processing status, and an optional asynchronous result callback may be invoked later to deliver the result.

Also, the QMI interface and interprocess communication mechanism used by the location API do not guarantee the delivery of every message, which may cause the messages to drop occasionally. Client applications must be designed in a way to recover from this rare event. An asynchronous interface allows for a more robust design, especially when the interface is not 100% reliable.

4.2 Process single shot API requests

getSinglePosition API is considered as single shot API. Invoke this API after invoking the CapabilitiesCb, which is registered with the `LocationClient` constructor. If this API is invoked before invoking the registered `CapabilitiesCb`, the app receives `LOCATION_ERROR_SYSTEM_NOT_READY` via `ResponseCb` that is registered.

4.3 Process tracking session, batching, and geofence API requests

Invoke these APIs after invoking `CapabilitiesCb`, which is registered with the `LocationClient` constructor.

4.4 Process concurrent API calls

In certain scenarios, the location API allows calling of multiple APIs one after the other without having to wait for a response from previous calls. For example, a tracking session request can be made without waiting for the previous single shot fix request to finish processing. However, client applications must handle this carefully and avoid using the same callbacks between the two calls.

For position tracking session, batching, and geofence functionalities, the location API allows only one type of session to be in progress at any time. For example, a batching session request is not allowed when a position tracking session is in progress.

If tracking session, batching, and geofencing must be running concurrently in one process, then three location API objects can be created. One object for performing location tracking, another object for performing batching, and the other object for performing geofencing. An unlimited number of location API objects can be created on an application processor.

5 Location APIs

An Application Programming Interface (API) is a set of programming code that allows applications to access and exchange information in a secure and efficient manner. An API is implemented by function calls that request the software to perform actions such as start or stop sessions, and create, read, update, or remove operations.

Qualcomm location API allows client applications to command and control the Global Navigation Satellite System (GNSS) engine and receive GNSS engine output information.

Note: The following location API functions are not supported in this release:

- `startPositionSession (specific GNSS engines)`
- `startTripBatchingSession`
- `updateNetworkAvailability`
- `getGnssEnergyConsumed`
- `getYearOfHw`

The supported API functions, enumerations, structures, and function pointer definitions used for location API in single shot fix, positioning tracking, batching, and geofence sessions are described as follows.

5.1 Single shot API functions

Single shot API functions are used to retrieve single shot position using positioning technologies.

Invoke the single shot APIs after invoking the CapabilitiesCb, which is registered with the `LocationClient` constructor. If these APIs are invoked before invoking the registered `CapabilitiesCb`, the app receives `LOCATION_ERROR_SYSTEM_NOT_READY` via `ResponseCb` that is registered.

The following figure shows a sample call flow of `getSinglePosition`:

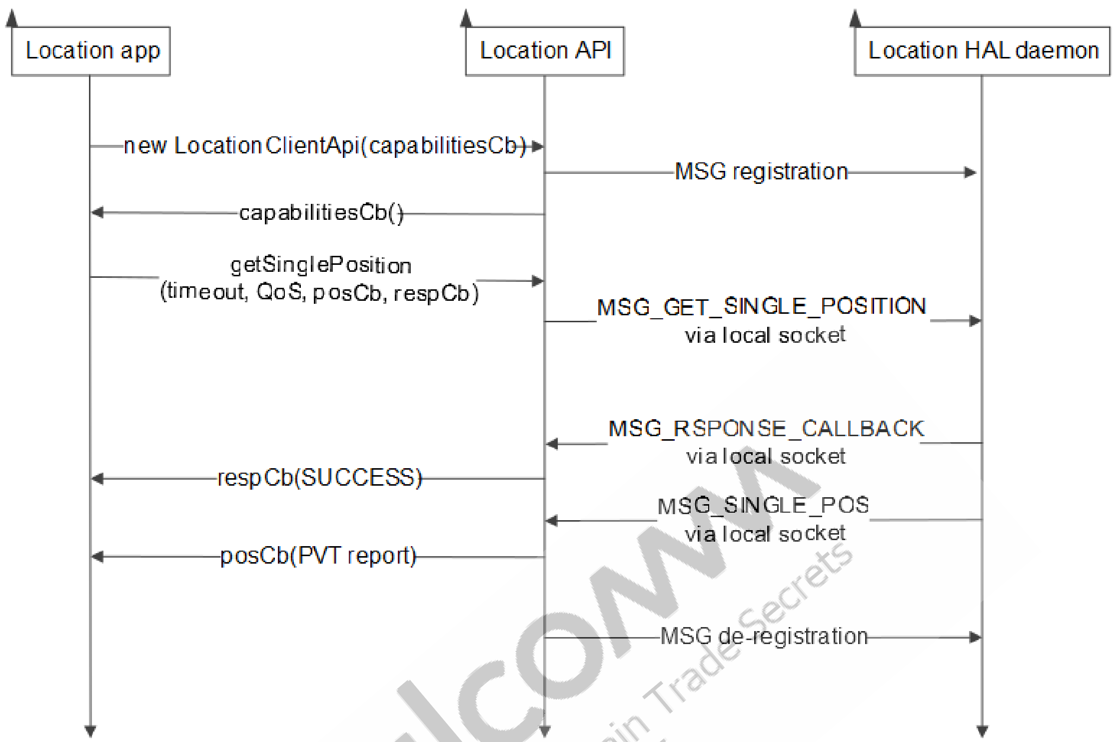

Figure1 Single shot fix call flow

getSinglePosition

Retrieves single shot position using the position technologies available, supported, and enabled on the device.

This API can be invoked with an ongoing tracking session initiated via startPositionSession.

Syntax

```
void getSinglePosition(uint32_t timeoutMsec, float horQos,
LocationCb PositionCallback, ResponseCb responseCallback);
```

Parameters

Parameter	Description
`timeoutMsec`	The amount of time that a user is willing to wait for the position to meet the QoS requirement. • When `timeoutMsec` is passed, the latest position received is delivered to the client and `responseCallback` is invoked with processing status set to `LOCATION_RESPONSE_TIMEOUT`. • If `timeoutMsec` is set to 0, `responseCallback` is invoked with `LOCATION_RESPONSE_PARAM_INVALID`.
`horQos`	The horizontal accuracy requirement for the fix. If `horQoS` is set to 0, `responseCallback` is invoked with `LOCATION_RESPONSE_PARAM_INVALID`.
`PositionCallback`	Callback to receive the position fix. • Some fields in `LocationClientApi::Location` such as speed, bearing, and their uncertainty may not be available. • `Check Location::flags` for the fields that are available. • This callback is invoked only when `responseCallback` is invoked with processing status set to `LOCATION_RESPONSE_SUCCESS` or `LOCATION_RESPONSE_TIMEOUT`. • Null `PositionCallback` cancels the current request. If `responseCallback` is not null, `LOCATION_RESPONSE_SUCCESS` is delivered.

Parameter	Description
`responseCallback`	Callback to receive processing status such as success or failure. An example of a failure code is timeout. • If a null `responseCallback` is passed, the processing status is not informed to the client. • When the processing status is `LOCATION_RESPONSE_SUCCESS`, the `PositionCallback` is invoked to deliver the single shot position report that meets the QoS requirement. • If this API is invoked with an invalid parameter, for example, 0 ms timeout, or `horQoS` set to a zero value, the `responseCallback` is invoked with `LOCATION_RESPONSE_PARAM_INVALID`. • When `timeoutMsec` has passed, the latest position received is delivered to the client and `responseCallback` is invoked with processing status set to `LOCATION_RESPONSE_TIMEOUT`. **Note:** The position received for the timeout scenario may not be recent enough to meet the QoS requirement. • If this API is invoked with a single shot position that is already in progress, the request fails and the `responseCallback` is invoked with `LOCATION_RESPONSE_REQUEST_ALREADY_IN_PROGRESS`.

Response

None

5.2 Tracking API functions

Position tracking API functions are used to retrieve basic and detailed location information.

Invoke the position tracking APIs after invoking the CapabilitiesCb, which is registered with the `LocationClient` constructor.

The following table provides a summary of the types of position tracking sessions.

Tracking session	Tracking type	Report types	PVT report info	PVT report engine
Simple location report	Time-based	Basic PVT report	Basic information such as timestamp, latitude, longitude, altitude, speed, bearing, uncertainty, tech mask.	Standard positioning engine (SPE)
Detailed location report	Time-based	• Detailed PVT report • SV report • NMEA report • Measurement report • Debug data report • DC message report	• Basic information • DOPs • Elliptical error • Measurement used in fix	SPE

The following figure shows the tracking call flow to receive a detailed location report. Note that the tracking sessions for basic report and detailed report from SPE engines have the same asynchronous call flow.

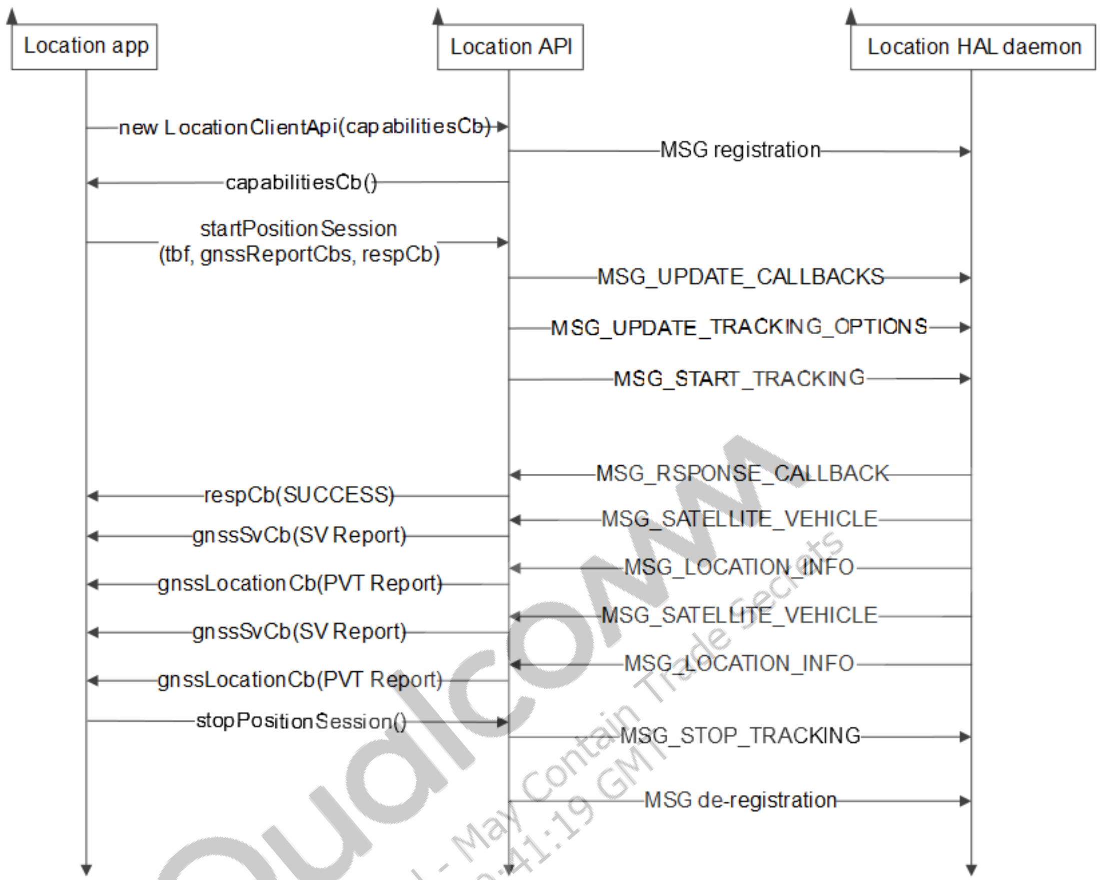

Figure2 Tracking session call flow

startPositionSession (basic location information)

Starts or updates a session with specified parameters to receive basic location information in the format of Location.

- If `locationCallback` is `nullptr`, this call is a no-op.
- If this API is called for the first time or after a previous position/batching/geofence session has been stopped, a position session will be started with the specified parameters and callbacks.
- If this API is called when the previous position/batching/geofence session has not yet received `responseCallback`, this API receives an error code of `LOCATION_RESPONSE_REQUEST_ALREADY_IN_PROGRESS` via its `responseCallback`.
- If this API is called during an ongoing session after the `responseCallback` has been received for the ongoing session, the parameters and callbacks will be updated, and the session continues but with a new set of parameters and callbacks.

Syntax

```
bool startPositionSession(uint32_t intervalInMs, uint32_t
distanceInMeters,
LocationCb locationCallback, ResponseCb responseCallback)
```

Parameters

Parameter	Description
`intervalInMs`	Time between fixes (TBF) in milliseconds (ms). • The underlying system determines the actual interval of reports received. • For example, if `intervalInMs` is specified as 10 ms, the report interval will be 100 ms if the highest report rate supported by the positioning engines is 10 Hz. • Also, if there is another application in the system having a session with a shorter interval, this client may benefit and receive reports at that interval.
`distanceInMeters`	Distance between fixes, in meters. 0 to indicate that the parameter does not take effect and the tracking is time-based. **Note:** By default, this parameter is always set to 0 for this release. It is not recommended to set this parameter to a nonzero value.
`locationCallback`	Callback to receive positions.
`responseCallback`	Callback to receive system responses. Value can be null.

Response

Returns `true` if a session is successfully started.

- If `responseCallback` is not null, it is invoked to deliver the processing status.
- If the processing status is `LOCATION_RESPONSE_SUCCESS`, LocationCb is invoked to deliver location information.

Returns `false` if a session is not started, that is, when `locationCallback` is `nullptr`.

- In this case, ResponseCb is not invoked.

startPositionSession (detailed location information)

Starts or updates a session with specified parameters to receive rich location information in the format of GnssLocation and other reports such as SV report and NMEA report.

- If `gnssReportCallbacks` is `nullptr`, this call is a no-op.
- If this API is called for the first time or after a previous position/batching/geofence session has been stopped, a position session will be started with the specified parameters and callbacks.
- If this API is called when the previous position/batching/geofence session has not yet received `responseCallback`, this API receives an error code of `LOCATION_RESPONSE_REQUEST_ALREADY_IN_PROGRESS` via its `responseCallback`.
- If called during an ongoing session after the `responseCb` has been received for the ongoing session, parameters and callbacks will be updated, and the session continues but with a new set of parameters and callbacks.

Syntax

```
bool startPositionSession(uint32_t intervalInMs, const GnssReportCbs
& gnssReportCallbacks,
ResponseCb responseCallback);
```

Parameters

Parameter	Description
intervalInMs	Time between fixes (TBF) in milliseconds (ms). • The underlying system determines the actual interval of reports received. • For example, if intervalInMs is specified as 10 ms, the report interval will be 100 ms if the highest report rate supported by the positioning engines is 10 Hz. • Also, if there is another application in the system having a session with a shorter interval, this client may benefit and receive reports at that interval.
gnssReportCallbacks	Callbacks to receive GNSS locations, SV information, NMEA report, or SV measurement report.
responseCallback	Callback to receive system responses. Value can be null.

Response

Returns `true` if a session is successfully started.

- If `responseCallback` is not null, it is invoked to deliver the processing status.
- If the processing status is `LOCATION_RESPONSE_SUCCESS`, GnssReportCbs is invoked to deliver registered reports.

Returns `false` if a session is not started, that is, when `locationCallback` is `nullptr`.

- In this case, ResponseCb is not invoked.

stopPositionSession

Stops the ongoing positioning session and deregisters the callbacks of a previous position tracking sessions.

No callback is issued regarding the processing status.

Syntax

```
void stopPositionSession();
```

Parameters

None

Response

None

5.3 Batching API functions

Batching API functions are used to start or stop location batching sessions.

Invoke the batching APIs after invoking CapabilitiesCb, which is registered with the `LocationClient` constructor.

startRoutineBatchingSession

Starts a routine mode batching session with specified parameters.

- If this API is called when idle, or after any other previous position/batching/geofence session is stopped, it delivers one of the following results:
  - If `batchingCallback` is `nullptr`, this call is a no-op.
  - If both `minInterval` and `tripDistance` do not take effect, this call is a no-op. Otherwise a batching session is started with the specified parameters and callbacks.
- If this API is called when any previous position/batching/geofence session has not yet received `responseCallback`, then this API receives an error code of `LOCATION_RESPONSE_REQUEST_ALREADY_IN_PROGRESS` via its `responseCallback`.
- If this API is called during an ongoing session after the `responseCallback` has been received for the ongoing session, the parameters/callback will be updated, and the session continues but with a new set of parameters/callback.
- Locations are reported on the `batchingCallback` in batches when the batch is full.

Syntax

```
bool startRoutineBatchingSession(uint32_t minInterval, uint32_t
minDistance,
BatchingCb batchingCallback, ResponseCb responseCallback);
```

Parameters

Parameter	Description
`minInterval`	Time between fixes (TBF) in milliseconds (ms). The actual interval of reports received is not larger than milliseconds. This value is rounded up by the next interval granularity supported by the underlying system. • 0 to indicate that the parameter does not take effect. • The underlying system may have a minimum interval threshold (for example, 100 ms or 1000 ms). Effective intervals are not smaller than this lower bound. • The effective intervals may have a granularity level higher than 1 ms, for example, 100 ms or 1000 ms. So milliseconds being 1559 may be honored at 1600 ms or 2000 ms, depending on the system. • Where there is another application in the system having a session with a shorter interval, this client may benefit and receive reports at that interval.
`minDistance`	Specifies the minimum distance, in meters, that should be traversed before a position should be batched. If 0, the positions are batched after the `minInterval` period expires. **Note:** By default, this parameter is always set to 0 for this release. Do not set this parameter to a nonzero value.

Parameter	Description
batchingCallback	Callback to receive batching positions and status.
responseCallback	(Optional) Callback to receive system responses.

Response

Returns `true` if a batching session is successfully started.

Returns `false` if a batching session is not started, that is, when `batchingCallback` is `nullptr`.

stopBatchingSession

Stops the ongoing batching session and deregisters the callbacks of a previous batching session.

No callback is issued regarding the processing status.

Syntax

```
void stopBatchingSession();
```

Parameters

None

Response

None

5.4 Geofence API functions

Geofence API functions are used to add, remove, modify, pause, or resume location geofencing sessions.

Invoke the geofence APIs after invoking CapabilitiesCb, which is registered with the `LocationClient` constructor.

addGeofences

Adds any number of geofences.

- The `geofenceBreachCallback` delivers the status of each geofence according to the geofence parameter.
- If this API is called when any previous position/batching/geofence session has not yet received `responseCallback`, then this API receives an error code of `LOCATION_RESPONSE_REQUEST_ALREADY_IN_PROGRESS` via its `responseCallback`.

Syntax

```
void addGeofences(std::vector<Geofence>& geofences, GeofenceBreachCb
gfBreachCb,
CollectiveResponseCb responseCallback);
```

Parameters

Parameter	Description
`geofences`	Geofence objects. When `addGeofences` is returned, each geofence object in the vector serves as the identifier throughout the remaining communication of that geofence. Such a geofence object can be copied or cloned, but they all reference the same geofence.
`gfBreachCb`	Callback to receive geofences state change. If `gfBreachCb` is null, `addGeofences` is no-op.
`responseCallback`	(Optional) Callback to receive geofence IDs and system responses.

Response

None

removeGeofences

Removes any number of geofences.

Syntax

```
void removeGeofences(std::vector<Geofence>& geofences);
```

Parameters

Parameter	Description
`geofences`	Geofence objects. This parameter must be originally added to the system; otherwise it is a no-op.

Response

None

modifyGeofences

Modifies any number of geofences.

Syntax

```
void modifyGeofences(std::vector<Geofence>& geofences);
```

Parameters

Parameter	Description
`geofences`	Geofence objects. Geofence objects must be originally added to the system; otherwise it is a no-op. The fields that can be modified include `breachTypeMask`, `responsiveness`, and `dwelltime`. A geofence that has been added to the system may have these fields modified. But it does not take any effect until `modifyGeofences()` is called with the changed geofence passed in.

Response

None

pauseGeofences

Pauses any number of geofences similar to removeGeofences, however, they can be resumed anytime.

Syntax

```
void pauseGeofences(std::vector<Geofence>& geofences);
```

Parameters

Parameter	Description
geofences	Geofence objects. Geofence objects must be originally added to the system; otherwise it is a no-op.

Response

None

resumeGeofences

Resumes any number of geofences that are currently paused.

Syntax

```
void resumeGeofences(std::vector<Geofence>& geofences);
```

Parameters

Parameter	Description
geofences	Geofence objects. This parameter must be originally added to the system; otherwise it is a no-op.

Response

None

5.5 Miscellaneous API functions

The `updateLocationSystemInfoListener` function registers/updates the listener to receive location system information that is not tied with a positioning session, for example, the next leap second event.

One set of callbacks can be registered per instance of the location API. The callback may be invoked multiple times to update the same or different pieces of system information.

Syntax

```
void updateLocationSystemInfoListener(LocationSystemInfoCb
locSystemInfoCallback,
ResponseCb responseCallback);
```

Parameters

Parameter	Description
`locSystemInfoCallback`	Callback to receive system information update. To stop receiving the update, pass a null callback.
`responseCallback`	Callback to receive the processing status of this API call. This callback can be null.

Response

None

5.6 API enums

Enumerations (enums) are a data type that have a limited set of possible values, usually identifiers, for a given property. Unlike other data types, enums are restricted to specific or predefined values for enhancing clarity and predictability in code.

The enums used for the location API are as follows.

LocationCapabilitiesMask

Specifies the set of location features supported by the location API.

Note: Features that are not listed in the table, though they are set to `true` in the returned `LocationCapabilitiesMask` enum, are not supported in this release.

Enum	Value	Description
LOCATION_CAPS_TIME_BASED_TRACKING_BIT	0x1	Supports time-based tracking via: `LocationClientApi::startPositionSession(uint32_t, const GnssReportCbs&, ResponseCb)` `LocationClientApi::startPositionSession(uint32_t, uint32_t, LocationCb, ResponseCb)` with `distanceInMeters` set to 0.
LOCATION_CAPS_TIME_BASED_BATCHING_BIT	0x2	Supports time-based batching via:`LocationClientApi::startRoutineBatchingSession()` with `minInterval` specified.
LOCATION_CAPS_GEOFENCE_BIT	0x10	Supports geofence via: `LocationClientApi::addGeofences()`.
LOCATION_CAPS_GNSS_MEASUREMENTS_BIT	0x40	Supports receiving `GnssMeasurements` data in `GnssMeasurementsCb` when `LocationClientApi` is in a positioning session.

PositioningEngineMask

Specifies the set of position engines supported by the location API.

Enum	Value	Description
STANDARD_POSITIONING_ENGINE	0x1	Mask for GNSS standard positioning engine (SPE).

Enum	Value	Description

GnssSvOptionsMask

Specifies the valid fields and additional status for GNSS SVs.

Enum	Value	Description
GNSS_SV_OPTIONS_HAS_EPHEMER_BIT	0x1	Indicates that ephemeris is available for this SV.
GNSS_SV_OPTIONS_HAS_ALMANAC_BIT	0x2	Indicates that an almanac is available for this SV.
GNSS_SV_OPTIONS_USED_IN_FIX_BIT	0x4	This SV is used in the position fix that has the output engine type set to LOC_OUTPUT_ENGINE_SPE.
GNSS_SV_OPTIONS_HAS_CARRIER_FREQUENCY_BIT	0x8	This SV has a valid carrierFrequencyHz field.
GNSS_SV_OPTIONS_HAS_GNSS_SIGNAL_TYPE_BIT	0x10	This SV has a valid gnssSignalTypeMask field.
GNSS_SV_OPTIONS_HAS_BASEBAND_CARRIER_TO_NOISE_BIT	0x20	This SV has a valid basebandCarrierToNoise field.
GNSS_SV_OPTIONS_HAS_ELEVATION_BIT	0x40	This SV has a valid GnssSv::elevation field.

Enum	Value	Description
GNSS_SV_OPTIONS_HAS_AZIMUTH_BIT	0x80	This SV has a valid GnssSv::azimuth field.

LocationFlagsMask

Specifies the valid fields in Location.

The user should determine the validity of a field in Location by checking whether its corresponding bit in Location::flags is set.

Enum	Value	Description
LOCATION_HAS_LAT_LONG_BIT	0x1	Location has valid latitude and longitude fields.
LOCATION_HAS_ALTITUDE_BIT	0x2	Location has valid altitude field.
LOCATION_HAS_SPEED_BIT	0x4	Location has valid speed field.
LOCATION_HAS_BEARING_BIT	0x8	Location has valid bearing field.
LOCATION_HAS_ACCURACY_BIT	0x10	Location has valid horizontalAccuracy field.
LOCATION_HAS_VERTICAL_ACCURACY_BIT	0x20	Location has valid verticalAccuracy field.
LOCATION_HAS_SPEED_ACCURACY_BIT	0x40	Location has valid speedAccuracy field.
LOCATION_HAS_BEARING_ACCURACY_BIT	0x80	Location has valid bearingAccuracy field.
LOCATION_HAS_TIMESTAMP_BIT	0x100	Location has valid timestamp field.
LOCATION_HAS_ELAPSED_REAL_TIME_BIT	0x200	Location has valid elapsedRealTime field.
LOCATION_HAS_ELAPSED_REAL_TIME_UNC_BIT	0x400	Location has valid elapsedRealTimeUnc field.

Enum	Value	Description
LOCATION_HAS_TIME_UNC_BIT	0x800	Location has valid timeUncMs field.

LocationTechnologyMask

Specifies the set of technologies that contribute to Location.

Enum	Value	Description
LOCATION_TECHNOLOGY_GNSS_BIT	0x1	GNSS-based technology is used to calculate location.
LOCATION_TECHNOLOGY_SENSORS_BIT	0x8	Sensor-based technology is used to calculate location.
LOCATION_TECHNOLOGY_PROPAGATION_BIT	0x800	Propagation logic data is used to calculate location.

Enum	Value	Description

GnssLocationNavSolutionMask

Specifies the set of navigation solutions that contribute to GnssLocation.

Enum	Value	Description
LOCATION_SBAS_CORRECTION_IONO_BIT	0x1	SBAS ionospheric correction is used to calculate GnssLocation.
LOCATION_SBAS_CORRECTION_FAST_BIT	0x2	SBAS fast correction is used to calculate GnssLocation.
LOCATION_SBAS_CORRECTION_LONG_BIT	0x4	SBAS long-term correction is used to calculate GnssLocation.
LOCATION_SBAS_INTEGRITY_BIT	0x8	SBAS integrity information is used to calculate GnssLocation.

Enum	Value	Description
LOCATION_NAV_CORRECTION_DGNSS_BIT	0x10	DGNSS correction is used to calculate GnssLocation.
LOCATION_NAV_CORRECTION_RTK_BIT	0x20	RTK correction is used to calculate GnssLocation.
LOCATION_NAV_CORRECTION_PPP_BIT	0x40	PPP correction is used to calculate GnssLocation.
LOCATION_NAV_CORRECTION_RTK_FIXED_BIT	0x80	RTK fixed correction is used to calculate GnssLocation.
LOCATION_NAV_CORRECTION_ONLY_SBAS_CORRECTED_SV_USED_BIT	0x100	Only SBAS corrected SVs are used to calculate GnssLocation.

GnssSignalTypeMask

Specifies the mask for available GNSS signal type and RF band used in GnssSv::gnssSignalTypeMask and GnssMeasUsageInfo::gnssSignalType.

Enum	Value	Description
GNSS_SIGNAL_GPS_L1CA_BIT	0x1	The GNSS signal is of the GPS L1CA RF band.
GNSS_SIGNAL_GPS_L1C_BIT	0x2	The GNSS signal is of the GPS L1C RF band.
GNSS_SIGNAL_GPS_L2_BIT	0x4	The GNSS signal is of the GPS L2 RF band.
GNSS_SIGNAL_GPS_L5_BIT	0x8	The GNSS signal is of the GPS L5 RF band.
GNSS_SIGNAL_GLONASS_G1_BIT	0x10	The GNSS signal is of the GLONASS G1 (L1OF) RF band.
GNSS_SIGNAL_GLONASS_G2_BIT	0x20	The GNSS signal is of the GLONASS G2 (L2OF) RF band.
GNSS_SIGNAL_GALILEO_E1_BIT	0x40	The GNSS signal is of the Galileo E1 RF band.
GNSS_SIGNAL_GALILEO_E5A_BIT	0x80	The GNSS signal is of the Galileo E5A RF band.
GNSS_SIGNAL_GALILEO_E5B_BIT	0x100	The GNSS signal is of the Galileo E5B RF band.
GNSS_SIGNAL_BEIDOU_B1_BIT	0x200	The GNSS signal is of the BeiDou B1 RF band.
GNSS_SIGNAL_BEIDOU_B2_BIT	0x400	The GNSS signal is of the BeiDou B2 RF band.
GNSS_SIGNAL_QZSS_L1CA_BIT	0x800	The GNSS signal is of the QZSS L1CA RF band.
GNSS_SIGNAL_QZSS_L1S_BIT	0x1000	The GNSS signal is of the QZSS L1S RF band.
GNSS_SIGNAL_QZSS_L2_BIT	0x2000	The GNSS signal is of the QZSS L2 RF band.
GNSS_SIGNAL_QZSS_L5_BIT	0x4000	The GNSS signal is of the QZSS L5 RF band.
GNSS_SIGNAL_SBAS_L1_BIT	0x8000	The GNSS signal is of the SBAS L1 RF band.
GNSS_SIGNAL_BEIDOU_B1I_BIT	0x10000	The GNSS signal is of the BeiDou B1I RF band.
GNSS_SIGNAL_BEIDOU_B1C_BIT	0x20000	The GNSS signal is of the BeiDou B1C RF band.

Enum	Value	Description
GNSS_SIGNAL_BEIDOU_B2I_BIT	0x40000	The GNSS signal is of the BeiDou B2I RF band.
GNSS_SIGNAL_BEIDOU_B2AI_BIT	0x80000	The GNSS signal is of the BeiDou B2AI RF band.
GNSS_SIGNAL_NAVIC_L5_BIT	0x100000	The GNSS signal is of the NavIC L5 RF band.
GNSS_SIGNAL_BEIDOU_B2AQ_BIT	0x200000	The GNSS signal is of the BeiDou B2A_Q RF band.

Enum	Value	Description

LocationResponse

Specifies the processing status of location API function call. The status is returned via ResponseCb.

Enum	Value	Description
LOCATION_RESPONSE_SUCCESS	0	The location API call is successful.
LOCATION_RESPONSE_UNKOWN_FAILURE	1	The location API call has failed.
LOCATION_RESPONSE_NOT_SUPPORTED	2	The location API call is not supported.
LOCATION_RESPONSE_PARAM_INVALID	3	The location API call has an invalid parameter.
LOCATION_RESPONSE_TIMEOUT	4	The location API call timeout.
LOCATION_RESPONSE_REQUEST_ALREADY_IN_PROGRESS	5	The location API is busy.
LOCATION_RESPONSE_SYSTEM_NOT_READY	6	The system is not ready, for example, the HAL daemon is not ready.
LOCATION_RESPONSE_EXCLUSIVE_SESSION_IN_PROGRESS	7	The location API does not support simultaneous tracking and batching session. Other session is ongoing.

GnssSvType

Specifies the SV constellation type in GnssSv and GnssMeasurementsData.

Enum	Value	Description
GNSS_SV_TYPE_UNKNOWN	0	The SV belongs to an unknown constellation.
GNSS_SV_TYPE_GPS	1	The SV belongs to the GPS constellation.
GNSS_SV_TYPE_SBAS	2	The SV belongs to the SBAS constellation.
GNSS_SV_TYPE_GLONASS	3	The SV belongs to the GLONASS constellation.
GNSS_SV_TYPE_QZSS	4	The SV belongs to the QZSS constellation.
GNSS_SV_TYPE_BEIDOU	5	The SV belongs to the BeiDou constellation.
GNSS_SV_TYPE_GALILEO	6	The SV belongs to the Galileo constellation.
GNSS_SV_TYPE_NAVIC	7	The SV belongs to the NavIC constellation.

Enum	Value	Description

GnssLocationInfoFlagMask

Specifies the valid fields in GnssLocation.

To determine if a field in `GnssLocation` is valid, check whether the corresponding bit in `GnssLocation::gnssInfoFlags` is set.

Enum	Value	Description
GNSS_LOCATION_INFO_ALTITUDE_MEAN_SEA_LEVEL_BIT	1ULL<<0	GnssLocation has valid altitudeMeanSeaLeve field.
GNSS_LOCATION_INFO_DOP_BIT	1ULL<<1	GnssLocation has valid pdop, hdop, and vdop fields.
GNSS_LOCATION_INFO_MAGNETIC_DEVIATION_BIT	1ULL<<2	GnssLocation has valid magneticDeviation field.
GNSS_LOCATION_INFO_HOR_RELIABILITY_BIT	1ULL<<3	GnssLocation has valid horReliability field.
GNSS_LOCATION_INFO_VER_RELIABILITY_BIT	1ULL<<4	GnssLocation has valid verReliability field.

Enum	Value	Description
GNSS_LOCATION_INFO_HOR_ACCURACY_ELIP_SEMI_MAJOR_BIT	1ULL<<5	GnssLocation has valid horUncEllipseSem field.
GNSS_LOCATION_INFO_HOR_ACCURACY_ELIP_SEMI_MINOR_BIT	1ULL<<6	GnssLocation has valid horUncEllipseSem field.
GNSS_LOCATION_INFO_HOR_ACCURACY_ELIP_AZIMUTH_BIT	1ULL<<7	GnssLocation has valid horUncEllipseOri field.
GNSS_LOCATION_INFO_GNSS_SV_USED_DATA_BIT	1ULL<<8	GnssLocation has valid svUsedInPosition field.
GNSS_LOCATION_INFO_NAV_SOLUTION_MASK_BIT	1ULL<<9	GnssLocation has valid navSolutionMask field.
GNSS_LOCATION_INFO_POS_TECH_MASK_BIT	1ULL<<10	GnssLocation has valid posTechMask field.
GNSS_LOCATION_INFO_POS_DYNAMICS_DATA_BIT	1ULL<<12	GnssLocation has valid bodyFrameData field.

Enum	Value	Description
`GNSS_LOCATION_INFO_EXT _DOP_BIT`	`1ULL<<13`	`GnssLocation` has valid `gdop` and `tdop` fields.
`GNSS_LOCATION_INFO_NORTH_STD_DEV_BIT`	`1ULL<<14`	`GnssLocation` has valid `northStdDeviation` field.
`GNSS_LOCATION_INFO_EAST_STD_DEV_BIT`	`1ULL<<15`	`GnssLocation` has valid `eastStdDeviation` field.
`GNSS_LOCATION_INFO_NORTH_VEL_BIT`	`1ULL<<16`	`GnssLocation` has valid `northVelocity` field.
`GNSS_LOCATION_INFO_EAST_VEL_BIT`	`1ULL<<17`	`GnssLocation` has valid `eastVelocity` field.
`GNSS_LOCATION_INFO_UP_VEL_BIT`	`1ULL<<18`	`GnssLocation` has valid `upVelocity` field.
`GNSS_LOCATION_INFO_NORTH_VEL_UNC_BIT`	`1ULL<<19`	`GnssLocation` has valid `northVelocityStdDev` field.

Enum	Value	Description
GNSS_LOCATION_INFO_EAST_VEL_UNC_BIT	1ULL<<20	GnssLocation has valid eastVelocityStdDe field.
GNSS_LOCATION_INFO_UP_VEL_UNC_BIT	1ULL<<21	GnssLocation has valid upVelocityStdDevi field.
GNSS_LOCATION_INFO_LEAP_SECONDS_BIT	1ULL<<22	GnssLocation has valid leapSeconds field.
GNSS_LOCATION_INFO_TIME _UNC_BIT	1ULL<<23	GnssLocation has valid timeUncMs field.
GNSS_LOCATION_INFO_NUM_SV_USED_IN_POSITION_BIT	1ULL<<24	GnssLocation has valid numSvUsedInPositi field.
GNSS_LOCATION_INFO_CALIBRATION_STATUS_BIT	1ULL<<26	GnssLocation has valid calibrationStatus field.
GNSS_LOCATION_INFO_OUTPUT_ENG_TYPE_BIT	1ULL<<27	GnssLocation has valid locOutputEngType field.
GNSS_LOCATION_INFO_OUTPUT_ENG_MASK_BIT	1ULL<<28	GnssLocation has valid locOutputEngMask field.

Enum	Value	Description
GNSS_LOCATION_INFO_CONFORMITY_INDEX_BIT	1ULL<<29	GnssLocation has valid conformityIndex field.
GNSS_LOCATION_INFO_ALTITUDE_ASSUMED_BIT	1ULL<<33	GnssLocation has valid altitudeAssumed field.
GNSS_LOCATION_INFO_SESSION_STATUS_BIT	1ULL<<34	GnssLocation has valid sessionStatus field.
GNSS_LOCATION_INFO_DGNSS_STATION_ID_BIT	1ULL<<39	GnssLocation has valid dgnssStationId field.

LocationReliability

Specifies the reliability level of GnssLocation horizontal and vertical reliability.

Enum	Value	Description
LOCATION_RELIABILITY_NOT_SET	0	GnssLocation reliability is not set.
LOCATION_RELIABILITY_VERY_LOW	1	GnssLocation reliability is very low. Use it at your own risk.
LOCATION_RELIABILITY_LOW	2	GnssLocation reliability is low. Little or no cross-checking is possible.
LOCATION_RELIABILITY_MEDIUM	3	GnssLocation reliability is medium. Limited cross-check has passed.
LOCATION_RELIABILITY_HIGH	4	GnssLocation reliability is high. A strong cross-check passed.

Gnss_LocSvSystemEnumType

Specifies to which SV a constellation belongs in GnssMeasUsageInfo and GnssSystemTime.

Enum	Value	Description
GNSS_LOC_SV_SYSTEM_GPS	1	The SV belongs to the GPS constellation.
GNSS_LOC_SV_SYSTEM_GALILEO	2	The SV belongs to the Galileo constellation.
GNSS_LOC_SV_SYSTEM_SBAS	3	The SV belongs to the SBAS constellation.
GNSS_LOC_SV_SYSTEM_GLONASS	4	The SV belongs to the GLONASS constellation.
GNSS_LOC_SV_SYSTEM_BDS	5	The SV belongs to the BDS constellation.
GNSS_LOC_SV_SYSTEM_QZSS	6	The SV belongs to the QZSS constellation.
GNSS_LOC_SV_SYSTEM_NAVIC	7	The SV belongs to the NavIC constellation.

GnssSystemTimeStructTypeFlags

Specifies the valid fields in GnssSystemTimeStructType.

Enum	Value	Description
GNSS_SYSTEM_TIME_WEEK_VALID	0x1	GnssSystemTimeStructType has valid systemWeek field.
GNSS_SYSTEM_TIME_WEEK_MS_VALID	0x2	GnssSystemTimeStructType has valid systemMsec field.
GNSS_SYSTEM_CLK_TIME_BIAS_VALID	0x4	GnssSystemTimeStructType has valid systemClkTimeBias field.
GNSS_SYSTEM_CLK_TIME_BIAS_UNC_VALID	0x8	GnssSystemTimeStructType has valid systemClkTimeUncMs field.
GNSS_SYSTEM_REF_FCOUNT_VALID	0x10	GnssSystemTimeStructType has valid refFCount field.
GNSS_SYSTEM_NUM_CLOCK_RESETS_VALID	0x20	GnssSystemTimeStructType has valid numClockResets field.

Enum	Value	Description

GnssGloTimeStructTypeFlags

Specifies the valid fields in GnssGloTimeStructType.

Enum	Value	Description
GNSS_GLO_DAYS_VALID	0x1	GnssGloTimeStructType has valid gloDays field.
GNSS_GLO_MSEC_VALID	0x2	GnssGloTimeStructType has valid gloMsec field.
GNSS_GLO_CLK_TIME_BIAS_VALID	0x4	GnssGloTimeStructType has valid gloClkTimeBias field.
GNSS_GLO_CLK_TIME_BIAS_UNC_VALID	0x8	GnssGloTimeStructType has valid gloClkTimeUncMs field.
GNSS_GLO_REF_FCOUNT_VALID	0x10	GnssGloTimeStructType has valid refFCount field.
GNSS_GLO_NUM_CLOCK_RESETS_VALID	0x20	GnssGloTimeStructType has valid numClockResets field.
GNSS_GLO_FOUR_YEAR_VALID	0x40	GnssGloTimeStructType has valid gloFourYear field.

LocOutputEngineType

Specifies the type of position engine that produced GnssLocation.

Enum	Value	Description
LOC_OUTPUT_ENGINE_FUSED	0	In this release, this output is the same as the SPE report.
LOC_OUTPUT_ENGINE_SPE	1	This fix is the unmodified fix from the modem GNSS engine.
LOC_OUTPUT_ENGINE_COUNT	4	Entry count of this enum.

LocSessionStatus

Specifies the status of the location session.

Enum	Value	Description
LOC_SESS_SUCCESS	0	Indicates that the session is successful.
LOC_SESS_INTERMEDIATE	1	Indicates that the session is in progress, and the reported session has not yet achieved the need criteria.
LOC_SESS_FAILURE	2	Indicates that the session has failed.

GnssSignalTypes

Specifies the GNSS signal type and RF band for jammer information and automatic gain control metric in GnssData.

To find out the jammer information and automatic gain control metric for a particular GNSS signal type, refer to the array element with index set to the signal type.

Enum	Value	Description
GNSS_SIGNAL_TYPE_GPS_L1CA	0	The GNSS signal belongs to the GPS L1CA RF band.
GNSS_SIGNAL_TYPE_GPS_L1C	1	The GNSS signal belongs to the GPS L1C RF band.
GNSS_SIGNAL_TYPE_GPS_L2C_L	2	The GNSS signal belongs to the GPS L2C_L RF band.
GNSS_SIGNAL_TYPE_GPS_L5_Q	3	The GNSS signal belongs to the GPS L5_Q RF band.
GNSS_SIGNAL_TYPE_GLONASS_G1	4	The GNSS signal belongs to the GLONASS G1 (L1OF) RF band.
GNSS_SIGNAL_TYPE_GLONASS_G2	5	The GNSS signal belongs to the GLONASS G2 (L2OF) RF band.
GNSS_SIGNAL_TYPE_GALILEO_E1_C	6	The GNSS signal belongs to the Galileo E1_C RF band.
GNSS_SIGNAL_TYPE_GALILEO_E5A_Q	7	The GNSS signal belongs to the Galileo E5A_Q RF band.
GNSS_SIGNAL_TYPE_GALILEO_E5B_Q	8	The GNSS signal belongs to the Galileo E5B_Q RF band.
GNSS_SIGNAL_TYPE_BEIDOU_B1_I	9	The GNSS signal belongs to the BeiDou B1_I RF band.
GNSS_SIGNAL_TYPE_BEIDOU_B1C	10	The GNSS signal belongs to the BeiDou B1C RF band.
GNSS_SIGNAL_TYPE_BEIDOU_B2_I	11	The GNSS signal belongs to the BeiDou B2_I RF band.

Enum	Value	Description
GNSS_SIGNAL_TYPE_BEIDOU_B2A_I	12	The GNSS signal belongs to the BeiDou B2A_I RF band.
GNSS_SIGNAL_TYPE_QZSS_L1CA	13	The GNSS signal belongs to the QZSS L1CA RF band.
GNSS_SIGNAL_TYPE_QZSS_L1S	14	The GNSS signal belongs to the QZSS L1S RF band.
GNSS_SIGNAL_TYPE_QZSS_L2C_L	15	The GNSS signal belongs to the QZSS L2C_L RF band.
GNSS_SIGNAL_TYPE_QZSS_L5_Q	16	The GNSS signal belongs to the QZSS L5_Q RF band.
GNSS_SIGNAL_TYPE_SBAS_L1_CA	17	The GNSS signal belongs to the SBAS L1_CA RF band.
GNSS_SIGNAL_TYPE_NAVIC_L5	18	The GNSS signal belongs to the NavIC L5 RF band.
GNSS_SIGNAL_TYPE_BEIDOU_B2A_Q	19	The GNSS signal belongs to the BeiDou B2A_Q RF band.
GNSS_MAX_NUMBER_OF_SIGNAL_TYPES	19	Indicates a maximum number of signal types.

GnssDataMask

Specifies the valid mask data fields in GnssData.

Enum	Value	Description
GNSS_DATA_JAMMER_IND_BIT	1ULL << 0	Indicates that the jammer indicator is available.
GNSS_DATA_AGC_BIT	1ULL << 1	Indicates that the AGC is available.

GnssDCReportType

Specifies the types of disaster and crisis reports currently supported by the GNSS engine.

Enum	Value	Description
QZSS_JMA_DISASTER_PREVENTION_INFO	43	Disaster prevention information provided by the Japan meteorological agency.
QZSS_NON_JMA_DISASTER_PREVENTION_INFO	44	Disaster prevention information provided by other organizations.

GnssMeasurementsDataFlagsMask

Specifies the valid fields in GnssMeasurementsData.

Enum	Value	Description
GNSS_MEASUREMENTS_DATA_SV_ID_BIT	0x1	GnssMeasureme has valid svId field.
GNSS_MEASUREMENTS_DATA_SV_TYPE_BIT	0x2	GnssMeasureme has valid svType field.
GNSS_MEASUREMENTS_DATA_STATE_BIT	0x4	GnssMeasureme has valid stateMask field.
GNSS_MEASUREMENTS_DATA_RECEIVED_SV_TIME_BIT	0x8	GnssMeasureme has valid receivedSvTir and receivedSvTir fields.
GNSS_MEASUREMENTS_DATA_RECEIVED_SV_TIME_UNCERTAINTY_BIT	0x10	GnssMeasureme has valid receivedSvTir field.
GNSS_MEASUREMENTS_DATA_CARRIER_TO_NOISE_BIT	0x20	GnssMeasureme has valid carrierToNois field.
GNSS_MEASUREMENTS_DATA_PSEUDORANGE_RATE_BIT	0x40	GnssMeasureme has valid pseudorangeRa field.

Enum	Value	Description
GNSS_MEASUREMENTS_DATA_PSEUDORANGE_RATE_UNCERTAINTY_BIT	0x80	GnssMeasurement has valid pseudorangeRate field.
GNSS_MEASUREMENTS_DATA_ADR_STATE_BIT	0x100	GnssMeasurement has valid adrStateMask field.
GNSS_MEASUREMENTS_DATA_ADR_BIT	0x200	GnssMeasurement has valid adrMeters field.
GNSS_MEASUREMENTS_DATA_ADR_UNCERTAINTY_BIT	0x400	GnssMeasurement has valid adrUncertaintyM field.
GNSS_MEASUREMENTS_DATA_CARRIER_FREQUENCY_BIT	0x800	GnssMeasurement has valid carrierFrequenc field.
GNSS_MEASUREMENTS_DATA_CARRIER_CYCLES_BIT	0x1000	GnssMeasurement has valid carrierCycles field.
GNSS_MEASUREMENTS_DATA_CARRIER_PHASE_BIT	0x2000	GnssMeasurement has valid carrierPhase field.
GNSS_MEASUREMENTS_DATA_CARRIER_PHASE_UNCERTAINTY_BIT	0x4000	GnssMeasurement has valid carrierPhaseUnc field.

Enum	**Value**	**Description**
GNSS_MEASUREMENTS_DATA_MULTIPATH_INDICATOR_BIT	0x8000	GnssMeasureme has valid multipathIndi field.
GNSS_MEASUREMENTS_DATA_SIGNAL_TO_NOISE_RATIO_BIT	0x10000	GnssMeasureme has valid signalToNoise field.
GNSS_MEASUREMENTS_DATA_AUTOMATIC_GAIN_CONTROL_BIT	0x20000	GnssMeasureme has valid agcLevelDb field.
GNSS_MEASUREMENTS_DATA_FULL_ISB_BIT	0x40000	GnssMeasureme has valid fullInterSign field.
GNSS_MEASUREMENTS_DATA_FULL_ISB_UNCERTAINTY_BIT	0x80000	GnssMeasureme has valid fullInterSign field.
GNSS_MEASUREMENTS_DATA_CYCLE_SLIP_COUNT_BIT	0x100000	GnssMeasureme has valid cycleslipCoun field.
GNSS_MEASUREMENTS_DATA_GNSS_SIGNAL_TYPE_BIT	0x200000	GnssMeasureme has valid gnssSignalTyp field.
GNSS_MEASUREMENTS_DATA_BASEBAND_CARRIER_TO_NOISE_BIT	0x400000	GnssMeasureme has valid basebandCarri field.

GnssMeasurementsStateMask

Specifies the GNSS measurement state in GnssMeasurementsData::stateMask.

Enum	Value	Description
GNSS_MEASUREMENTS_STATE_UNKNOWN_BIT	0	The GNSS measurement state is unknown.
GNSS_MEASUREMENTS_STATE_CODE_LOCK_BIT	0x1	The GNSS measurement state is code lock.
GNSS_MEASUREMENTS_STATE_BIT_SYNC_BIT	0x2	The GNSS measurement state is bit sync.
GNSS_MEASUREMENTS_STATE_SUBFRAME_SYNC_BIT	0x4	The GNSS measurement state is subframe sync.
GNSS_MEASUREMENTS_STATE_TOW_DECODED_BIT	0x8	The GNSS measurement state is tow decoded.
GNSS_MEASUREMENTS_STATE_MSEC_AMBIGUOUS_BIT	0x10	The GNSS measurement state is msec ambiguous.
GNSS_MEASUREMENTS_STATE_SYMBOL_SYNC_BIT	0x20	The GNSS measurement state is symbol sync.
GNSS_MEASUREMENTS_STATE_GLO_STRING_SYNC_BIT	0x40	The GNSS measurement state is GLONASS string sync.

Enum	Value	Description
GNSS_MEASUREMENTS_STATE_GLO_TOD_DECODED_BIT	0x80	The GNSS measurement state is GLONASS TOD decoded.
GNSS_MEASUREMENTS_STATE_BDS_D2_BIT_SYNC_BIT	0x100	The GNSS measurement state is BDS D2 bit sync.
GNSS_MEASUREMENTS_STATE_BDS_D2_SUBFRAME_SYNC_BIT	0x200	The GNSS measurement state is BDS D2 subframe sync.
GNSS_MEASUREMENTS_STATE_GAL_E1BC_CODE_LOCK_BIT	0x400	The GNSS measurement state is Galileo E1BC code lock.
GNSS_MEASUREMENTS_STATE_GAL_E1C_2ND_CODE_LOCK_BIT	0x800	The GNSS measurement state is Galileo E1C second code lock.
GNSS_MEASUREMENTS_STATE_GAL_E1B_PAGE_SYNC_BIT	0x1000	The GNSS measurement state is Galileo E1B page sync.

Enum	Value	Description
GNSS_MEASUREMENTS_STATE_SBAS_SYNC_BIT	0x2000	The GNSS measurement state is SBAS sync.

GnssMeasurementsAdrStateMask

Specifies the accumulated delta range (ADR) state in GnssMeasurementsData::adrStateMask.

Enum	Value	Description
GNSS_MEASUREMENTS_ACCUMULATED_DELTA_RANGE_STATE_UNKNOWN	0	ADR state is unknown.
GNSS_MEASUREMENTS_ACCUMULATED_DELTA_RANGE_STATE_VALID_BIT	0x1	ADR state is valid.
GNSS_MEASUREMENTS_ACCUMULATED_DELTA_RANGE_STATE_RESET_BIT	0x2	ADR state is reset.
GNSS_MEASUREMENTS_ACCUMULATED_DELTA_RANGE_STATE_CYCLE_SLIP_BIT	0x4	ADR state is cycle slip.

Enum	Value	Description

GnssMeasurementsMultipathIndicator

Specifies the GNSS multipath indicator state in GnssMeasurementsData::multipathIndicator.

Enum	Value	Description
`GNSS_MEASUREMENTS_MULTIPATH _INDICATOR_UNKNOWN`	`0`	The GNSS multipath indicator is unknown.
`GNSS_MEASUREMENTS_MULTIPATH _INDICATOR_PRESENT`	`1`	The GNSS multipath indicator is available.
`GNSS_MEASUREMENTS_MULTIPATH _INDICATOR_NOT_PRESENT`	`2`	The GNSS multipath indicator is not available.

GnssMeasurementsClockFlagsMask

Specifies the valid fields in GnssMeasurementsClock.

Enum	Value	Description
`GNSS_MEASUREMENTS_CLOCK_FLAGS_LEAP_SECOND_BIT`	`0x1`	`GnssMeasu` has valid `leapSecon` field.
`GNSS_MEASUREMENTS_CLOCK_FLAGS_TIME_BIT`	`0x2`	`GnssMeasu` has valid `timeNs` field.
`GNSS_MEASUREMENTS_CLOCK_FLAGS_TIME_UNCERTAINTY_BIT`	`0x4`	`GnssMeasu` has valid `timeUncer` field.

Enum	Value	Description
GNSS_MEASUREMENTS_CLOCK_FLAGS_FULL_BIAS_BIT	0x8	GnssMeasure has valid fullBiasNs field.
GNSS_MEASUREMENTS_CLOCK_FLAGS_BIAS_BIT	0x10	GnssMeasure has valid biasNs field.
GNSS_MEASUREMENTS_CLOCK_FLAGS_BIAS_UNCERTAINTY_BIT	0x20	GnssMeasure has valid biasUncerta field.
GNSS_MEASUREMENTS_CLOCK_FLAGS_DRIFT_BIT	0x40	GnssMeasure has valid driftNsps field.
GNSS_MEASUREMENTS_CLOCK_FLAGS_DRIFT_UNCERTAINTY_BIT	0x80	GnssMeasure has valid driftUncert field.
GNSS_MEASUREMENTS_CLOCK_FLAGS_HW_CLOCK_DISCONTINUITY_COUNT_BIT	0x100	GnssMeasure has valid hwClockDisc field.

Enum	Value	Description

LeapSecondSysInfoMask

Specifies the valid fields in LeapSecondSystemInfo.

Enum	Value	Description
LEAP_SECOND_SYS_INFO_CURRENT_LEAP_SECONDS_BIT	1ULL << 0	LeapSecondSystemInfo has valid leapSecondCurrent field.
LEAP_SECOND_SYS_INFO_LEAP_SECOND_CHANGE_BIT	1ULL << 1	LeapSecondSystemInfo has valid leapSecondChangeInfo field.

LocationSystemInfoMask

Specifies the set of valid fields in LocationSystemInfo.

Enum	Value	Description
LOC_SYS_INFO_LEAP_SECOND	1ULL << 0	LocationSystemInfo has valid leapSecondSysInfo field.

BatchingStatus

Specifies the batching status in BatchingCb.

Enum	Value	Description
BATCHING_STATUS_INACTIVE	0	The service is unable to compute positions for batching.
BATCHING_STATUS_ACTIVE	1	The service is able to compute positions for batching.

GeofenceBreachTypeMask

Specifies the geofence breach or dwell event in GeofenceBreachCb.

Enum	Value	Description
GEOFENCE_BREACH_ENTER_BIT	0x1	Indicates that a client entered the geofence.
GEOFENCE_BREACH_EXIT_BIT	0x2	Indicates that a client left the geofence.
GEOFENCE_BREACH_DWELL_IN_BIT	0x4	Indicates that a client dwelt inside the geofence.
GEOFENCE_BREACH_DWELL_OUT_BIT	0x8	Indicates that a client dwelt outside the geofence.

5.7 API structures

Structures define the information held by the API input and output parameters. An API structure allows developers to define the number of API message parameters required and how they should be structured.

The structures used for the location API are as follows.

Field	Type	Description

GnssLocationSvUsedInPosition

Specifies the set of SVs that are used to calculate GNSS location.

Field	Type	Description
gpsSvUsedIdsMask	UINT64	Specifies the set of SVs from the GPS constellation that are used to compute the position. Bit 0 to bit 31 corresponds to GPS SV ID 1 to 32.
gloSvUsedIdsMask	UINT64	Specifies the set of SVs from the GLONASS constellation that are used to compute the position. Bit 0 to bit 31 corresponds to GLO SV ID 65 to 96.
galSvUsedIdsMask	UINT64	Specifies the set of SVs from the Galileo constellation that are used to compute the position. Bit 0 to bit 35 corresponds to GAL SV ID 301 to 336.
bdsSvUsedIdsMask	UINT64	Specifies the set of SVs from the BeiDou constellation that are used to compute the position. Bit 0 to bit 62 corresponds to BDS SV ID 201 to 263.
qzssSvUsedIdsMask	UINT64	Specifies the set of SVs from the QZSS constellation that are used to compute the position. Bit 0 to bit 4 corresponds to QZSS SV ID 193 to 197.
navicSvUsedIdsMask	UINT64	Specifies the set of SVs from the NavIC constellation that are used to compute the position. Bit 0 to bit 13 corresponds to NavIC SV ID 401 to 414.

GnssMeasUsageInfo

Specifies the SV measurements used for calculating GNSS location.

Field/Method	Type	Description
gnssConstellation	Gnss_LocSvSystemEnumType	Specifies GNSS constellation Type for the SV
gnssSvId	UINT64	Specifies satellite vehicle ID number. For information on the SV ID range of each supported constellation, see GnssSv::svId.
gnssSignalType	GnssSignalTypeMask	Specifies the signal type mask of the SV.

Field/Method	Type	Description
toString	STRING	Method to print the structure in human readable form for logging purposes.

GnssSystemTimeStructType

Provides information about non-GLONASS GNSS system time.

Field	Type	Description
validityMask	GnssSystemTimeStructTypeFlags	Specifies valid fields in GnssSystemTimeStructType.
systemWeek	UINT16	Extended week number at reference tick, in unit of week. • Set the value to 65535 if the week number is unknown. • For GPS, it is calculated from midnight, January 6, 1980. OTA decoded 10-bit GPS week is extended to map between [NV6264 to (NV6264 + 1023)]. • For BDS, it is calculated from 00:00:00 on January 1, 2006 of Coordinated Universal Time (UTC). • For GAL, it is calculated from 00:00 UT on Sunday August 22, 1999 (midnight between August 21 and August 22).
systemMsec	UINT32	Time in to the current week at the reference tick, in milliseconds. The range is 0 to 604799999.
systemClkTimeBias	FLOAT	System clock time bias, in milliseconds. System time (TOW millisecond) = systemMsec - systemClkTimeBias
systemClkTimeUncMs	FLOAT	Single-sided maximum time bias uncertainty, in milliseconds.

Field	Type	Description
refFCount	UINT32	FCount (free running hardware timer) value, in milliseconds.
numClockResets	UINT32	Number of clock resets or discontinuities detected, which affects the local hardware counter value.

GnssGloTimeStructType

Provides information about the GLONASS system time.

Field	Type	Description
validityMask	GnssGloTimeStructTypeFlags	Specifies the valid fields in GnssGloTimeStructType.
gloFourYear	UINT8	GLONASS four-year number from 1996. Refer to GLONASS ICD. This field is applicable only for GLONASS and shall be ignored for other constellations.
gloDays	UINT16	GLONASS day number in 4 years. Refer to GLONASS ICD. Set the value to 65535 if the day number is unknown.
gloMsec	UINT32	GLONASS time of day in milliseconds. Refer to GLONASS ICD.
gloClkTimeBias	FLOAT	GLONASS clock time bias, in milliseconds. GLO time (TOD Millisecond) = gloMsec - gloClkTimeBias Check for gloClkTimeUncMs before using this field.
gloClkTimeUncMs	FLOAT	Single-sided maximum time bias uncertainty, in milliseconds.

Field	Type	Description
refFCount	UINT32	FCount (free running hardware timer) value, in milliseconds. Do not use this field for relative time purposes due to possible discontinuities.
numClockResets	UINT32	Number of clock resets/discontinuities detected, affecting the local hardware counter value.

SystemTimeStructUnion

Holds the GNSS system time from different constellations in GnssSystemTime.

Field	Type	Description
gpsSystemTime	GnssSystemTimeStructType	System time information from GPS constellation.
galSystemTime	GnssSystemTimeStructType	System time information from Galileo constellation.
bdsSystemTime	GnssSystemTimeStructType	System time information from BeiDou constellation.
qzssSystemTime	GnssSystemTimeStructType	System time information from QZSS constellation.
gloSystemTime	GnssSystemTimeStructType	System time information from GLONASS constellation.
navicSystemTime	GnssSystemTimeStructType	System time information from NavIC constellation.

GnssSystemTime

Specifies the GNSS system in GnssLocation.

Field/Method	Type	Description
gnssSystemTimeSrc	Gnss_LocSvSystemEnumType	Specifies the source constellation for GNSS system time.
u	SystemTimeStructUnion	Specifies the GNSS system time corresponding to the source.

Location

Specifies the location information received by the client via `startPositionSession (uint32_t, uint32_t LocationCb, ResponseCb).`

Field	Type	Description
`flags`	LocationFlagsMask	Specifies the valid fields.
`timestamp`	`UINT64`	UTC timestamp for location fix since January 1, 1970, in milliseconds.
`latitude`	`DOUBLE`	Latitude, in degrees. The range is [-90.0, 90.0].
`longitude`	`DOUBLE`	Longitude, in degrees. The range is [-180.0, 180.0]
`altitude`	`DOUBLE`	Altitude above the WGS 84 reference ellipsoid, in meters.
`speed`	`FLOAT`	Horizontal speed, in meters/second.
`bearing`	`FLOAT`	Bearing, in degrees. The range is [0, 360).
`horizontalAccuracy`	`FLOAT`	Horizontal accuracy, in meters. Uncertainty is defined with a 68% confidence level.
`verticalAccuracy`	`FLOAT`	Vertical accuracy, in meters. Uncertainty is defined with a 68% confidence level.
`speedAccuracy`	`FLOAT`	Horizontal speed uncertainty, in meters/second. Uncertainty is defined with a 68% confidence level.
`bearingAccuracy`	`FLOAT`	Bearing uncertainty, in degrees. The range is (0 to 359.999). Uncertainty is defined with a 68% confidence level.
`techMask`	LocationTechnologyMask	Sets of technology that contributed to the fix.

Field	Type	Description
elapsedRealTimeNs	UINT64	Boot timestamp, in nanoseconds, corresponding to the UTC timestamp for location fix. This field may not always be available. Check for the presence of LOCATION_HAS_ELAPSED_REAL_TIME_BIT in location::flags before retrieving this field.
elapsedRealTimeUncNs	UINT64	Uncertainty for the boot timestamp, in nanoseconds. This field may not always be available. Check for the presence of LOCATION_HAS_ELAPSED_REAL_TIME_UNC_BIT in location::flags before retrieving this field.
timeUncMs	FLOAT	Time uncertainty associated with this position, in milliseconds. This field may not always be available. Check for the presence of LOCATION_HAS_TIME_UNC_BIT in location::flags before retrieving this field.

GnssLocation

Specifies the location information received by the client via `startPositionSession(uint32_t, const GnssReportCbs&, ResponseCb)` and `startPositionSession(uint32_t, LocReqEngineTypeMask, const EngineReportCbs&, ResponseCb)`.

Note: Unsupported fields for this release are not listed in the table.

Field	Type	Description
gnssInfoFlags	GnssLocationInfoFlagMask	Specifies the validity of parameters.

Field	Type	Description
altitudeMeanSeaLevel	FLOAT	Altitude with respect to mean sea level, in meters.
pdop	FLOAT	Position dilution of precision (PDOP). Ranges from 0 (highest accuracy) to 50 (lowest accuracy).
hdop	FLOAT	Horizontal dilution of precision (HDOP). Ranges from 0 (highest accuracy) to 50 (lowest accuracy).
vdop	FLOAT	Vertical dilution of precision (VDOP). Ranges from 0 (highest accuracy) to 50 (lowest accuracy).
gdop	FLOAT	Geometric dilution of precision (GDOP). Ranges from 0 (highest accuracy) to 50 (lowest accuracy).
tdop	FLOAT	Time dilution of precision (TDOP). Ranges from 0 (highest accuracy) to 50 (lowest accuracy).
magneticDeviation	FLOAT	The difference between the bearing to true north and the bearing shown on a magnetic compass. The deviation is positive when the magnetic north is east of the true north.
horReliability	LocationReliability	Horizontal reliability.
verReliability	LocationReliability	Vertical reliability.
horUncEllipseSemiMajor	FLOAT	Horizontal elliptical accuracy semi-major axis, in meters. Uncertainty is defined with a 39% confidence level.
horUncEllipseSemiMinor	FLOAT	Horizontal elliptical accuracy semi-minor axis, in meters. Uncertainty is defined with a 39% confidence level.
horUncEllipseOrientAzimuth	FLOAT	Horizontal elliptical accuracy azimuth, in degrees. The range is [0, 180]. Confidence for uncertainty is not specified.
northStdDeviation	FLOAT	North standard deviation, in meters. Uncertainty is defined with a 68% confidence level.
eastStdDeviation	FLOAT	East standard deviation, in meters. Uncertainty is defined with a 68% confidence level.
northVelocity	FLOAT	North velocity, in meters/sec.
eastVelocity	FLOAT	East velocity, in meters/sec.
upVelocity	FLOAT	Up velocity, in meters/sec.
northVelocityStdDeviation	FLOAT	North velocity uncertainty, in meters/sec. Uncertainty is defined with a 68% confidence level.
eastVelocityStdDeviation	FLOAT	East velocity uncertainty, in meters/sec. Uncertainty is defined with a 68% confidence level.

Field	Type	Description
upVelocityStdDeviation	FLOAT	Up velocity uncertainty, in meters/sec. Uncertainty is defined with a 68% confidence level.
numSvUsedInPosition	UINT16	Number of SVs used in position report.
svUsedInPosition	GnssLocationSvUsed InPosition	GNSS SVs used in position data.
navSolutionMask	GnssLocationNavSolution Mask	Navigation solutions that are used to calculate the position report.
posTechMask	LocationTechnologyMask	Position technology used in computing this fix.
gnssSystemTime	GnssSystemTime	GNSS system time when this position is calculated.
measUsageInfo	std::vector <GnssMeasUsageInfo>	GNSS measurement usage information.
leapSeconds	UINT8	The number of leap seconds at the time when this position is generated.
locOutputEngType	LocOutputEngineType	Location engine type. When this field is set to LOC_ENGINE_SRC_FUSED, the fix is the propagated or aggregated reports from SPE. To check which location engine contributes to the fused output, check for locOutputEngMask.
locOutputEngMask	PositioningEngineMask	When locOutputEngType is set to fused, this field indicates the set of engines that contribute to the fix.
altitudeAssumed	BOOL	When this field is valid, it indicates whether altitude is assumed or calculated. • FALSE: The altitude is calculated. • TRUE: The altitude is assumed; there may not be enough satellites to determine the precise altitude.
sessionStatus	LocSessionStatus	Indicates whether the session is a success, failure, or intermediate.
dgnssStationId	std::vector <uint16_t>	List of DGNSS station IDs providing corrections. Range: • SBAS: 120 to 158 and 183 to 191 • Monitoring station: 1000 to 2023 (station ID biased by 1000) • Other values reserved

GnssSv

Specifies the GNSS SV report that comes when the client registers for `location_client::GnssSvCb`.

Field	Type	Description
`svId`	`UINT16`	Unique SV identifier. This field is always valid. The SV range for supported constellations is as follows: • For GPS: 1 to 32 • For GLONASS: 65 to 96 or FCN+104 • [65, 96] if orbital slot number (OSN) is known. • [97, 110] as frequency channel number (FCN) [-7, 6] plus 104, that is, encode FCN -7 as 97, 0 as 104, 6 as 110. • For SBAS: 120 to 158 and 183 to 191 • For QZSS: 193 to 197 • For BDS: 201 to 263 • For GAL: 301 to 336 • For NavIC: 401 to 414
`type`	`GnssSvType`	Constellation type of the SV (GPS, SBAS, GLONASS, QZSS, BeiDou, Galileo). This field is always valid.
`cN0Dbhz`	`FLOAT`	Carrier-to-noise ratio of the signal measured at the antenna, in dB Hz. `cN0Dbhz` of 0.0 indicates that this field is unknown.
`elevation`	`FLOAT`	Elevation of the SV, in degrees. This field is always valid.
`azimuth`	`FLOAT`	Azimuth of the SV, in degrees. This field is always valid.

Field	Type	Description
gnssSvOptionsMask	GnssSvOptionsMask	Specifies additional information and valid fields in GnssSv. This field is always valid.
carrierFrequencyHz	FLOAT	Carrier frequency of the signal tracked. This field is valid if gnssSvOptionsMask has GNSS_SV_OPTIONS_HAS_CARRIER_FREQUENCY_BIT set.
gnssSignalTypeMask	GnssSignalTypeMask	GNSS signal type mask of the SV. This field is valid if gnssSvOptionsMask has GNSS_SV_OPTIONS_HAS_GNSS_SIGNAL_TYPE_BIT set.
basebandCarrierToNoiseDbHz	DOUBLE	Carrier-to-noise ratio of the signal measured at baseband, in dB Hz. This field is valid if gnssSvOptionsMask has GNSS_SV_OPTIONS_HAS_BASEBAND_CARRIER_TO_NOISE_BIT set.
gloFrequency	UINT16	GLONASS frequency channel number. The range is [1, 14]. This field is valid only when SV is of GLONASS.

GnssData

Specifies the additional GNSS data that can be provided during a tracking session. Currently jammer data and automatic gain control data are available.

To find out the jammer information and automatic gain control metric for a particular GNSS signal type, refer to the array element with index set to the interested RF band.

- Determine whether GnssData::jammerInd is valid by checking if the element at the index of the specified RF band in GnssData::gnssDataMask has GNSS_DATA_JAMMER_IND_BIT set.
- Determine whether GnssData::agc is valid by checking if the element at the index of the specified RF band in GnssData::gnssDataMask has GNSS_DATA_AGC_BIT set.

Field	Type	Description
gnssDataMask [GNSS_MAX_NUMBER_OF_SIGNAL_TYPES]	GnssDataMask	Indicates valid data fields.
jammerInd [GNSS_MAX_NUMBER_OF_SIGNAL_TYPES]	DOUBLE	Jammer indication for each GNSS signal.
agc [GNSS_MAX_NUMBER_OF_SIGNAL_TYPES]	DOUBLE	Automatic gain control metric, in dB.

GnssDcReport

Specifies the type and data payload contained in the disaster and crisis report received from the GNSS engine.

Field	Type	Description
dcReportType	GnssDcReportType	Disaster and crisis report type, as defined in standard.
numValidBits	UINT32	Number of valid bits that the client should make use in the payload specified in GnssDcReport::dcReportData.
dcReportData	UINT8	Disaster and crisis report data packed into uint8_t.

Field	Type	Description

GnssMeasurementsData

Specifies the SV pseudo range and carrier phase measurement from a standard positioning engine (SPE).

Determine the validity of a field in GnssMeasurementsClock by checking whether its corresponding bit in `GnssMeasurementsClock::flags` is set.

Field	Type	Description
`flags`	GnssMeasurementsDataFlagsMask	Specifies the valid fields in `GnssMeasurementsData`.
`svId`	`INT16`	Specifies SV ID number. For the SV ID range of each supported constellation, refer to documentation in `GnssSv::svId`.
`svType`	GnssSvType	SV constellation type.
`timeOffsetNs`	`DOUBLE`	Time offset when the measurement was taken, in nanoseconds.
`stateMask`	GnssMeasurementsStateMask	Specifies the GNSS measurement state.
`receivedSvTimeNs`	`INT64`	Received GNSS time of the week in nanoseconds when the measurement was taken. For subnanoseconds part of the time, refer to field of `GnssMeasurementsData::receivedSvTimeSubNs`. Total time is: `receivedSvTimeNs + receivedSvTimeSubNs`.
`receivedSvTimeSubNs`	`FLOAT`	The subnanoseconds portion of the received GNSS time of the week when the measurement was taken. For a nanoseconds portion of the time, refer to field of `GnssMeasurementsData::receivedSvTimeSubNs`. Total time is: `receivedSvTimeNs + receivedSvTimeSubNs`.
`receivedSvTimeUncertaintyNs`	`INT64`	Satellite time. All SV times in the current measurement block are already propagated to a common reference time epoch, in nanoseconds.
`carrierToNoiseDbHz`	`DOUBLE`	Signal strength, carrier to noise ratio, in dB Hz.
`pseudorangeRateMps`	`DOUBLE`	Uncorrected pseudorange rate, in meters/second.
`pseudorangeRateUncertaintyMps`	`DOUBLE`	Uncorrected pseudorange rate uncertainty, in meters/sec.
`adrStateMask`	GnssMeasurementsAdrStateMask	Bitwise OR of `GnssMeasurementsAdrStateMask`.
`adrMeters`	`DOUBLE`	Accumulated delta range, in meters.
`adrUncertaintyMeters`	`DOUBLE`	Accumulated delta range uncertainty, in meters.
`carrierFrequencyHz`	`FLOAT`	Carrier frequency of the tracked signal, in Hz.
`carrierCycles`	`INT64`	The number of full carrier cycles between the receiver and the satellite.
`carrierPhase`	`DOUBLE`	The RF carrier phase that the receiver has detected.
`carrierPhaseUncertainty`	`DOUBLE`	The RF carrier phase uncertainty.

Field	Type	Description
`multipathIndicator`	GnssMeasurements MultipathIndicator	The multipath indicator could be unknown, present, or not present.
`signalToNoiseRatioDb`	`DOUBLE`	Signal to noise ratio, in dB.
`agcLevelDb`	`DOUBLE`	Automatic gain control level, in dB.
`basebandCarrierToNoiseDbHz`	`DOUBLE`	Baseband signal strength, in dB Hz. This field should always be available in the measurement report.
`gnssSignalType`	GnssSignalTypeMask	GNSS signal type mask of the SV. This field should always be available in the measurement report.
`fullInterSignalBiasNs`	`DOUBLE`	The full intersignal bias (ISB) in nanoseconds. This value is the sum of the estimated receiver-side and the space-segment-side intersystem bias, interfrequency bias, and intercode bias.
`fullInterSignalBiasUncertaintyNs`	`DOUBLE`	1-sigma uncertainty associated with the full intersignal bias in nanoseconds.
`cycleSlipCount`	`UINT8`	Increments when a cycle slip is detected.

GnssMeasurementsClock

Specifies GNSS measurements clock.

The following equation describes the relationship between various components:

```
utcTimeNs = timeNs - (fullBiasNs + biasNs) - leapSecond * 1,000,000,000.
```

Field	Type	Description
`flags`	GnssMeasurementsClockFlagsMask	Bitwise OR of `GnssMeasurementsClockFlagsMask`.
`leapSecond`	`INT16`	Leap second, in seconds.
`timeNs`	`INT64`	Time, monotonically increasing as long as the power is on, in nanoseconds.
`timeUncertaintyNs`	`DOUBLE`	Time uncertainty (one sigma), in nanoseconds
`fullBiasNs`	`INT64`	Full bias, in nanoseconds.
`biasNs`	`DOUBLE`	Subnanoseconds bias, in nanoseconds.
`biasUncertaintyNs`	`DOUBLE`	Bias uncertainty (one sigma), in nanoseconds.
`driftNsps`	`DOUBLE`	Clock drift, in nanoseconds/second.
`driftUncertaintyNsps`	`DOUBLE`	Clock drift uncertainty (one sigma), in nanoseconds/second.
`hwClockDiscontinuityCount`	`UINT32`	Hardware clock discontinuity count; incremented for each discontinuity in hardware clock.

GnssMeasurements

Specifies the GNSS measurements clock and data.

Field	Type	Description
`clock`	GnssMeasurementsClock	GNSS measurements clock information.
`measurements`	std::vector <GnssMeasurementsData>	GNSS measurements data.
`isNhz`	`BOOL`	NHz measurements indicator.

LeapSecondChangeInfo

Specifies the leap second change event information as part of `LeapSecondSystemInfo`.

Field	Type	Description
`gpsTimestampLsChange`	GnssSystemTimeStructType	GPS timestamp that corresponds to the last known leap second change event. The information is available in two scenarios: • This leap second change event has been scheduled and yet to happen and the GPS receiver has decoded this information since the device's last bootup. • This leap second change event happened after the device was last booted up and the GPS receiver has decoded this information. **Note:** If the device is rebooted after leap second change, this information becomes unavailable.
`leapSecondsBeforeChange`	UINT8	Number of leap seconds, in seconds, before the leap second change event that corresponds to the timestamp at `gpsTimestampLsChange`.
`leapSecondsAfterChange`	UINT8	Number of leap seconds, in seconds, after the leap second change event that corresponds to the timestamp at `gpsTimestampLsChange`.

Field	Type	Description

LeapSecondSystemInfo

Specifies leap second system information, including current leap second and leap second change event information, if available.

Determine the validity of a field in `LeapSecondSystemInfo` by checking whether its corresponding bit in `LeapSecondSystemInfo::leapSecondInfoMask` is set.

Field	Type	Description
`leapSecondInfoMask`	LeapSecondSysInfoMask	Specifies the valid fields in LeapSecondSystemInfo.
`leapSecondCurrent`	`UINT8`	Current leap seconds, in seconds. This information is available in two scenarios: • When the leap second change information is available, to figure out the current leap second information, compare the current GPS time with `LeapSecondChangeInfo::gpsTimestampLsChange` to choose `leapSecondBefore` or `leapSecondAfter` as the current leap second. • When the leap second change information is not available, then use this field to retrieve the current leap second.
`leapSecondChangeInfo`	LeapSecondChangeInfo	GPS timestamp that corresponds to the last known leap second change event. The information is available in two scenarios: • This leap second change event has been scheduled and yet to happen and the GPS receiver has decoded this information since the device's last bootup. • This leap second change event happened after the device was last booted up and the GPS receiver has decoded this information. **Note:** If the device is rebooted after leap second change, this information becomes unavailable. If leap second change information is available, compare the current GPS time with `LeapSecondChangeInfo::gpsTimestampLsChange` to figure out the current leap second information and choose `leapSecondBefore` or `leapSecondAfter` as the current leap second.

LocationSystemInfo

Specifies the location system information that can be received via LocationSystemInfoCb.

Determine the validity of a field in LocationSystemInfo by checking whether its corresponding bit in `LocationSystemInfo::flags` is set.

Field	Type	Description
`systemInfoMask`	LocationSystemInfoMask	Indicates the valid fields in `LocationSystemInfo`.
`leapSecondSysInfo`	LeapSecondSystemInfo	Current leap second and leap second information.

GnssReportCbs

Specifies the set of callbacks to receive the reports when invoking `startPositionSession (uint32_t, LocReqEngineTypeMask, const GnssReportCbs&, ResponseCb)` with `intervalInMs` specified.

Field	Type	Description
`gnssLocationCallback`	GnssLocationCb	Callback to receive GnssLocation. • When there are multiple engines running on the system, the received location is a fused report from all engines. • When there is only a standard SPE engine running on the system, the received location is from the modem GNSS engine.
`gnssSvCallback`	GnssSvCb	Callback to receive GnssSv from the modem GNSS engine.
`gnssNmeaCallback`	GnssNmeaCb	Callback to receive NMEA sentences.
`gnssDataCallback`	GnssDataCb	Callback to receive GnssData from the modem GNSS engine.
`gnssMeasurementsCallback`	GnssMeasurementsCb	Callback to receive 1 Hz GnssMeasurements from the modem GNSS engine.
`gnssNHzMeasurementsCallback`	`GnssMeasurementsCb`	Callback to receive NHz `GnssMeasurements` from the modem GNSS engine.
`gnssDcReportCallback`	GnssDcReportCb	Callback to receive disaster and crisis report from the modem GNSS engine.

5.8 API function pointer definitions

A function pointer is a pointer variable that holds the address of an API function. Function pointers are useful in creating a callback mechanism and passing the address of a function to another function.

The function pointer definitions used for the location API are as follows.

CapabilitiesCb

Provides the capabilities of the system to the location API client.

Syntax

```
typedef std::function<void( LocationCapabilitiesMask capsMask )> CapabilitiesCb;
```

Parameters

Parameter	Description
capsMask	Bitwise OR of LocationCapabilitiesMask

Response

None

ResponseCb

Receives the processing status of the location API function calls, as defined in LocationResponse.

Syntax

```
typedef std::function<void( LocationResponse response )> ResponseCb;
```

Parameters

Parameter	Description
response	Response of the function call.

Response

Returns `LOCATION_RESPONSE_SUCCESS` if successful. Else, the last API call has failed.

LocationCb

Receives basic location information when the location API is in a positioning session, as defined in Location.

- If there are multiple engines running on the system, the received location information is a fused report from all engines.
- If there is only SPE running on the system, the received location information is from the modem GNSS engine.

Syntax

```
typedef std::function<void( const Location& location )> LocationCb;
```

Parameters

Parameter	Description
`location`	Basic location information.

Response

None

GnssLocationCb

Receives GNSS location that has richer information than basic location when the location API is a positioning session, as defined in GnssLocation.

- If there are multiple engines running on the system, the received GNSS location information is a fused report from all engines.
- If there is only SPE running on the system, the received GNSS location information is from the modem GNSS engine.

Syntax

```
typedef std::function<void(const GnssLocation& gnssLocation)> GnssLocationCb;
```

Parameters

Parameter	Description
`gnssLocation`	Rich GNSS location information.

Response

None

GnssSvCb

Receives GNSS SV information when the location API is in a positioning session, as defined in GnssSv.

Syntax

```
typedef std::function<void(const std::vector<GnssSv>& gnssSvs)> GnssSvCb;
```

Parameters

Parameter	Description
gnssSvs	GNSS SV report information.

Response

None

GnssNmeaCb

Receives NMEA sentences when the location API is in a positioning session.

Syntax

```
typedef std::function<void(uint64_t timestamp, const std::string& nmea)> GnssNmeaCb;
```

Parameters

Parameter	Description
timestamp	The timestamp when the NMEA sentence is generated.
nmea	The NMEA strings generated from position and SV reports.

Response

None

GnssDataCb

Receives GNSS data such as jammer information when the location API is in a positioning session, as defined in GnssData.

Syntax

```
typedef std::function<void(const GnssData& gnssData)> GnssDataCb;
```

Parameters

Parameter	Description
gnssData	GNSS jammer and automatic gain control (AGC) information.

Response

None

GnssMeasurementsCb

Receives information about GNSS measurements when the location API is in a positioning session, as defined in GnssMeasurements.

Syntax

```
typedef std::function<void(const GnssMeasurements& gnssMeasurements)> GnssMeasurementsCb;
```

Parameters

Parameter	Description
`gnssMeasurements`	Information about GNSS SV measurements.

Response

None

GnssDcReportCb

Receives GNSS disaster and crisis (DC) report information when the location API is in a positioning session, as defined in GnssDcReport.

Syntax

```
typedef std::function<void(const GnssDcReport& gnssDcReport)> GnssDcReportCb;
```

Parameters

Parameter	Descriptiongit
`gnssDcReport`	GNSS disaster and crisis report information.

Response

None

LocationSystemInfoCb

Receives location system information update, which rarely occurs, as defined in LocationSystemInfo.

Syntax

```
typedef std::function<void(const LocationSystemInfo & locationSystemInfo)>
LocationSystemInfoCb;
```

Parameters

Parameter	Description
`locationSystemInfo`	Rare location system event information, for example, leap second change.

Response

None

BatchingCb

Receives the locations in a batching session, as defined in Location and BatchingStatus.

Syntax

```
typedef std::function<void(const std::vector<Location>& locations, BatchingStatus
batchStatus)> BatchingCb;
```

Parameters

Parameter	Description
`locations`	The locations batched in a session.
`batchStatus`	Batching status of the batching session: • `BATCHING_STATUS_INACTIVE`: If the session is unable to compute positions for batching. • `BATCHING_STATUS_ACTIVE`: If the session is able to compute positions for batching.

Response

None

CollectiveResponseCb

Receives collective response from geofence APIs, as defined in LocationResponse.

Syntax

```
typedef std::function<void(std::vector<std::pair<Geofence, LocationResponse>>& responses)>
CollectiveResponseCb;
```

Parameters

Parameter	Description
responses	Includes the geofence objects and corresponding responses.

Response

None

GeofenceBreachCb

Receives the geofences that have a state change, as defined in Location and GeofenceBreachTypeMask.

Syntax

```
typedef std::function<void(const std::vector<Geofence>& geofences, Location location,
GeofenceBreachTypeMask type,
uint64_t timestamp)> GeofenceBreachCb;
```

Parameters

Parameter	Description
geofences	Array of geofence objects that have breached.
location	Location associated with breach.
type	Type of breach.
timestamp	The timestamp of the breach.

Response

None

6 Run location API test application

The following figure shows the operations you can perform using the location API test application.

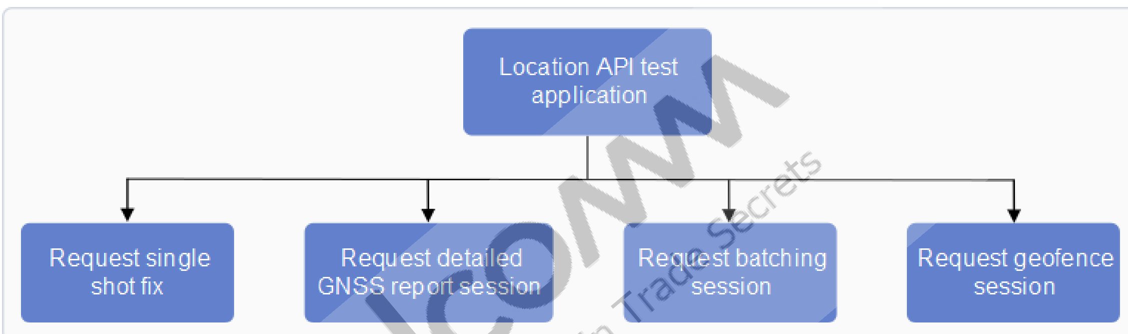

Figure1 Location API test application operations

You can run the `location_client_api_testapp` available at `/home/root` directory of the device using the command-line interface (CLI). Use `printHelp()` for options to explore the functions of the test application with the specified command-line syntax.

```
static void printHelp() {

# location_client_api_testapp
root@qcs6490:~# location_client_api_testapp
group ids: diag locclient
    Group diag = 53
    Group locclient = 4021

auto run 0, deleteAll 0, delete mask 0x0, session type 0,outputEnabled 1,
detailedOutputEnabled 0
<<< onCapabilitiesCb mask=0x2fff<<< onCapabilitiesCb mask string=capabMask: 0x2fff (TIME_
BASED_TRACKING | TIME_BASED_BATCHING | GEOFENCE | GNSS_MEASUREMENTS | GNSS_SINGLE_FREQ | GNSS_
MULTI_FREQ)

************* options *************
l tbf : Gnss location report session with tbf ms interval
g reporttype tbf : Gnss detailed reports session with tbf ms interval
b tbf: Position batching with tbf msec interval
s: Stop a session
q: Quit
r: delete client
disableReportOutput: supress output from various reports
enableReportOutput: enable output from various reports
enableDetailedReportOutput: enable detailed output from various reports
getSingleFusedFix: get single shot fix with qos and timeout
```

```
cancelSingleFusedFix: cancle single shot fix

addGeofences: add geofences with lat/lon/radius/breachtype/responsiveness/dwelltime
pauseGeofences: pause geofences with indexes
resumeGeofences: resume geofences with indexes
modifyGeofences: modify geofences with index/breachtype/responsiveness/dwelltime
removeGeofences: remove geofences with indexes
```

6.1 Sample commands for location test scenarios

The following tables list the command parameters and arguments that you can use to run test scenarios with the test application.

Command parameters

Parameter	Function used	Description
`l`	`startPositionSession(tbfMsec, 0, onLocationCb, onResponseCb);`	Starts a simple GNSS report session with a time between fixes (TBF) millisecond (ms) interval.
`g`	`startPositionSession(tbfMsec, reportcbs, onResponseCb);`	Starts a detailed GNSS report session with a TBF ms interval.
`b`	`startRoutineBatchingSession(tbfMsec, 0, onBatchingCb, onResponseCb)`	Starts a time-based batching with a TBF ms interval.
`s`	`stopPositionSession(); stopBatchingSession(); delete LCAClient instance`	Stops a session.
`r`	`delete LCAClient`	Deletes a client.

Command arguments

Argument	Description
`-a`	Auto starts a session.
`-i`	Sets the minimum interval value.
`-t`	Sets the timeout value.

Argument	Description
-r	Specifies the report type. Supported report types are as follows: • POSITION_REPORT = 1 << 0 • NMEA_REPORT = 1 << 1 • SV_REPORT = 1 << 2 • DATA_REPORT = 1 << 3 • MEAS_REPORT = 1 << 4 • NHZ_MEAS_REPORT = 1 << 5 • DC_REPORT = 1 << 6 • ENGINE_NMEA_REPORT = 1 << 7
-s g	Sets the session type to GNSS detailed location report.
-s l	Sets the session type to GNSS simple location report.
-l	Sets the fix count. • For a single shot session, set the value to 1. • For a tracking session, set the value to a larger number.
-V	Enables verbose logging to provide detailed information such as speed, bearing, uncertainty, DOP, and elliptical error.

Standalone single shot

Sample command to trigger a standalone single shot fix request is as follows:

```
location_client_api_testapp -a -i 1000 -t 30 -r 1 -s g -l 1
```

Where:

- -i is set to 1000 to start the session with a 1000 ms interval.
- -t is set to 30 to stop the session after 30 s.
- -s g is used to set the type of session to GNSS detailed location report.
- -l is set to 1 to stop the session after receiving one fix.

Tracking (simple location report)

Sample command to trigger GNSS 10 Hz tracking sessions is as follows:

```
location_client_api_testapp -a -i 100 -t 30 -s l -l 1000
```

Where:

- -i is set to 100 to start the session with a 100 ms interval.
- -t is set to 30 to stop the session after 30 s.
- -s l is used to set the type of session to GNSS simple location report.
- -l is set to 1000 to stop the session after receiving 1000 fixes.

Tracking (detailed location report)

Sample command to trigger GNSS 10 Hz tracking sessions is as follows:

```
location_client_api_testapp -a -V -i 100 -t 30 -r 7 -s g -l 1000
```

Where:

- `-i` is set to 100 to start the session with a 100 ms interval.
- `-t` is set to 30 to stop the session after 30 s.
- `-r` is set to 7 to register for location, SV, and NMEA reports. For more information on these types of reports, see Table : Command arguments.
- `-s g` is used to set the type of session to GNSS detailed location report.
- `-l` is set to 1000 to stop the session after receiving 1000 fixes.

Location batching

Sample command to trigger location batching in interactive mode is as follows:

```
location_client_api_testapp -b 1000 0 120
```

Where:

- `b` is set to 1000 to start routine batching session with a 1000 ms interval.
- The distance is set to 0 m.
- The duration is set to 120 s.

Location geofencing

Sample command to trigger location geofencing is as follows:

```
/ # location_client_api_testapp

addGeofences 41.374953 -121.984478 300 3 120000 5000, 41.376861 -121.9658 500 15 10000 20000,
41.1644 -121.916956 1000 7 1000 1000, 41.374253 -121.984478 300 3 120000 5000, 41.376261 -121.
9658 500 15 10000 20000
execute command addGeofences 41.374953 -121.984478 300 3 120000 5000, 41.376861 -121.9658 500
15 10000 20000, 41.1644 -121.916956 1000 7 1000 1000, 41.374253 -121.984478 300 3 120000 5000,
41.376261 -121.9658 500 15 10000 20000

usage: addGeofences [lat lon radius breachType responsiveness dwellTime ] ...
addGeofences, index: 0, latitude: 41.374954, longitude: -121.984482, radiusM: 300.000000,
breachType: 3, responsivenessMs: 120000, dwellTimeSec: 5000
addGeofences, index: 1, latitude: 41.376862, longitude: -121.965797, radiusM: 500.000000,
breachType: 15, responsivenessMs: 10000, dwellTimeSec: 20000
addGeofences, index: 2, latitude: 41.164398, longitude: -121.916954, radiusM: 1000.000000,
breachType: 7, responsivenessMs: 1000, dwellTimeSec: 1000
addGeofences, index: 3, latitude: 41.374252, longitude: -121.984482, radiusM: 300.000000,
breachType: 3, responsivenessMs: 120000, dwellTimeSec: 5000
addGeofences, index: 4, latitude: 41.376263, longitude: -121.965797, radiusM: 500.000000,
breachType: 15, responsivenessMs: 10000, dwellTimeSec: 20000
currently there are 5 geofences available, please input index and the latitude/longitude/
radius/breachType/responsiveness/dwelltime
<<< onCollectiveResponseCb, geofence cnt: 5
<<< onCollectiveResponseCb, lat=41.374954 lon=-121.984482 rad=300.000000, breachType: 3,
responsiveness: 120000, dwellTime: 5000, response: 0
<<< onCollectiveResponseCb, lat=41.376862 lon=-121.965797 rad=500.000000, breachType: 15,
```

```
responsiveness: 10000, dwellTime: 20000, response: 0
<<< onCollectiveResponseCb, lat=41.164398 lon=-121.916954 rad=1000.000000, breachType: 7,
responsiveness: 1000, dwellTime: 1000, response: 0
<<< onCollectiveResponseCb, lat=41.374252 lon=-121.984482 rad=300.000000, breachType: 3,
responsiveness: 120000, dwellTime: 5000, response: 0
<<< onCollectiveResponseCb, lat=41.376263 lon=-121.965797 rad=500.000000, breachType: 15,
responsiveness: 10000, dwellTime: 20000, response: 0
pauseGeofences 0
execute command pauseGeofences 0

usage: pauseGeofences gfIndex ...
currently there are 5 geofences available, please input index
pauseGeofences seqNum: 0, lat: 41.374954, lon: -121.984482, radiusM: 300.000000, breachType:
3,responsivenessMs: 120000, dwellTimeSec: 5000
<<< onCollectiveResponseCb, geofence cnt: 1
<<< onCollectiveResponseCb, lat=41.374954 lon=-121.984482 rad=300.000000, breachType: 3,
responsiveness: 120000, dwellTime: 5000, response: 0
```

7 Sample codes for location API functions

You can use the following sample codes for location API functions to perform certain operations such as requesting single shot fix, location tracking, location batching, and location geofencing sessions.

7.1 Request single shot fix

```
static void onCapabilitiesCb(location_client::LocationCapabilitiesMask mask) {
    //...
 }

 static void onSingleShotResponseCb(location_client::LocationResponse response) {
     printf("<<< onSingleShotResponseCb err=%u\n", response);
}

 static void onSingleShotLocationCb(const location_client::Location& location) {
         printf("<<< onSingleShotLocationCb time=%" PRIu64" mask=0x%x lat=%f lon=%f "
           "alt=%f accuracy=%f\n",
           location.timestamp,
           location.flags,
           location.latitude,
           location.longitude,
           location.altitude,
           location.horizontalAccuracy);
}
 void testSingleShotPositionApi() {
     LocationClientApi *pClient = new LocationClientApi(onCapabilitiesCb);
     if (nullptr == pClient) {
         LOC_LOGe("failed to create LocationClientApi instance");
         return;
     }
     uint32_t timeoutMsec = 60000,
     pClient->getSinglePosition(timeoutMsec, 0, onSingleShotLocationCb,
onSingleShotResponseCb);
     //...
 }
```

7.2 Request detailed GNSS report session

```
static void onCapabilitiesCb(location_client::LocationCapabilitiesMask mask)
{
    if (mask & LOCATION_CAPS_TIME_BASED_TRACKING_BIT)
    {
        // device has time based tracking capability
```

```
        // to start engine based location session
    }
static void onResponseCb(location_client::LocationResponse response)
{
    if (response == LOCATION_RESPONSE_SUCCESS)
    {
        // successfully started the tracking session
        // expecting to receive detailed GNSS PVT reports and other reports
        // the registered callbacks
    }
    else
    {
        // request to start the tracking session failed
        // detained GNSS PVT reports and other report callbacks will not be invoked
    }
}
static void onGnssLocationCb(const GnssLocation& location)
{
    //...
}
static void onGnssSvCb(const std::vector<location_client::GnssSv>& gnssSvs)
{
    //...
}
static void onGnssNmeaCb(uint64_t timestamp, const std::string& nmea)
{
    //...
}
void testDetailedGnssReportApi()
{
    LocationClientApi *pClient = new LocationClientApi(onCapabilitiesCb);
    if (nullptr == pClient)
    {
        LOC_LOGe("failed to create LocationClientApi instance");
        return;
    }
    uint32_t option = 0x111;
    uint32_t interval = 1000;
    // set callbacks
    GnssReportCbs reportcbs;
    if (option & 1<<0)
    {
        reportcbs.gnssLocationCallback = GnssLocationCb(onGnssLocationCb);
    }
    if (option & 1<<1)
    {
        reportcbs.gnssSvCallback = GnssSvCb(onGnssSvCb);
    }
    if (option & 1<<2)
    {
        reportcbs.gnssNmeaCallback = GnssNmeaCb(onGnssNmeaCb);
    }
    // start tracking session pClient->startPositionSession(interval, reportcbs,
onResponseCb);
    //...
    // stop session
    pClient->stopPositionSession();
//...
}
```

7.3 Request batching sessions

```
static void onCapabilitiesCb(location_client::LocationCapabilitiesMask mask)
{
confirm device has the needed batching capability
    if (mask & LOCATION_CAPS_TIME_BASED_BATCHING_BIT)
    {
        // device has time based batching capability
    }
    else if (mask & LOCATION_CAPS_DISTANCE_BASED_TRACKING_BIT)
    {
        // device has time distance batching capability
    }
    else if (mask & LOCATION_CAPS_OUTDOOR_TRIP_TRACKING_BIT)
    {
        // device has time distance batching capability
    }
}
static void onResponseCb(location_client::LocationResponse response)
{
    if (response == LOCATION_RESPONSE_SUCCESS)
    {
        // successfully made batching request
        // onBatchingCb() will be invoked to deliver batching status and
        // batched location
    }
    else
    {
        // batching request has failed
        // onBatchingCb() will not be invoked
    }
}
static void onBatchingCb(const std::vector<location_client::Location>& locations,
BatchingStatus status)
{
    if (status == BATCHING_STATUS_INACTIVE)
    {
        // device is unable to compute positions for batching
    }
    else if (status == BATCHING_STATUS_ACTIVE)
    {
        // device is able to compute positions for batching for (Locationloc : locations)
        {
            // retrieve each batch location
        }
    }
    //...
}
void testRoutineBatchingApi()
{
LocationClientApi *pClient = new LocationClientApi(onCapabilitiesCb);
    if (nullptr == pClient)
    {
        LOC_LOGe("failed to create LocationClientApi instance");
        return;
    }
    // batching session
    uint32_t intervalInMs = 0;
    uint32_t distanceInMeters = 0;
    pClient->startRoutineBatchingSession(intervalInMs, distanceInMeters, onBatchingCb,
onResponseCb);
    // ...
pClient->stopBatchingSession();
```

```
}
```

7.4 Request geofence sessions

```
vector<Geofence> sGeofences;
static void onGeofenceBreachCb( const std::vector<Geofence>& geofences,
Location location, GeofenceBreachTypeMask type, uint64_t timestamp)
{
    //...
}
static void onCapabilitiesCb(location_client::LocationCapabilitiesMask mask)
{
    // confirm device has the needed batching capability
    if (mask & LOCATION_CAPS_GEOFENCE_BIT)
    {
        // device has geofence capability
    }
}
static void onCollectiveResponseCb(std::vector<pair<Geofence, LocationResponse>>& responses)
{
    //...
}
void testGeofenceApi()
{
    double latitude = 32.896535;
    double longitude = -117.201025;
    double radius = 50;
    GeofenceBreachTypeMask type = (GeofenceBreachTypeMask)3;
    uint32_t responsiveness = 4000;
    uint32_t time = 0;
    Geofence gf(latitude, longitude, radius, type, responsiveness, time);
    sGeofences.push_back(gf);
    LocationClientApi *pClient = new LocationClientApi(onCapabilitiesCb);
    if (nullptr == pClient)
    {
        LOC_LOGe("failed to create LocationClientApi instance");
        return;
    }
    pClient->addGeofences(sGeofences, onGeofenceBreachCb, onCollectiveResponseCb);
    vector<Geofence> pauseGeofences;
    pauseGeofences.push_back(sGeofences[0]);
    pClient->pauseGeofences(pauseGeofences);
    vector<Geofence> resumeGeofences;
    resumeGeofences.push_back(sGeofences[0]);
    pClient->resumeGeofences(resumeGeofences);
    vector<Geofence> removeGeofences;
    for (int i=0; i<sGeofences.size(); ++i)
    {
    removeGeofences.push_back(sGeofences[i]);
    }
pClient->removeGeofences(removeGeofences);
}
```

8 Debug location issues

This information describes how to handle errors and enable logging to debug issues related to the location API. Sample logs are also provided to aid in debugging.

8.1 Logging and debugging

You can use console logs and sys logs for logging and debugging issues related to the location API.

Console logs

The `DEBUG_LEVEL` configuration item in the `/etc/gps.conf` file controls the logs from the location module. The default value for `DEBUG_LEVEL` is `3`.

To collect advanced logs for analyzing and debugging issues, set `DEBUG_LEVEL` to `5` in `gps.conf` and reboot the device.

Set debug level using SSH

You can also use SSH commands to set the debug level in `gps.conf`. After logging into the device via SSH, do the following:

1. Pull `gps.conf` to the current directory.

```
mount -o remount, rw /
```

```
scp -r root@< device_IP_address>:/etc/gps.conf .
```

2. Update the debug level to `5` in `gps.conf` and reboot the device.

```
scp -r gps.conf root@< device_IP_address >:/etc/gps.conf .
```

```
reboot
```

Note: When prompted for a password, enter `oelinux123` to authenticate the file transfer via the secure copy protocol (SCP).

You can now start capturing the logs with debug level 5.

sys logs

To capture sys logs, run the following command:

```
journalctl -f > <FILENAME>
```

For example:

```
journalctl -f > syslog_logs.txt
```

8.2 Sample logs for standalone single shot fix

Sample console log for standalone single shot fix is as follows:

```
location_client_api_testapp -a -i 1000 -t 30 -r 1 -s g -l 1
group ids: diag locclient
Group diag = 2901
Group locclient = 4021
tiemout: 30
report type: 1
session type: g
fix cnt: 1
auto run 1, deleteAll 0, delete mask 0x0, session type 2,outputEnabled 1,
detailedOutputEnabled 0 routeToNMEAPort 0pid: 2781
<<< onCapabilitiesCb mask=0xcff
<<< onCapabilitiesCb mask string=capabMask: 0xcff (TIME_BASED_TRACKING | TIME_BASED_BATCHING |
DIST_BASED_TRACKING | DIST_BASED_BATCHING | GEOFENCE | OUTDOOR_TRIP_BATCHING | GNSS_
MEASUREMENTS | CONSTELLATION_ENABLE | GNSS_SINGLE_FREQ | GNSS_MULTI_FREQ)
<<< onResponseCb err=0
<<< onGnssLocationCb cnt=(0/1): time=0 mask=0x873 lat=0.000000 lon=0.000000 alt=0.000000
diag: Diag_LSM_Init: invoked for pid: 2781 with init_count: 0
diag:successfully connected to socket 6
diag: Diag_LSM_Init: done for pid: 2781 with init_count: 1

<<< onGnssLocationCb cnt=(1/14): time=1708585393713 mask=0xf77 lat=17.429013 lon=78.379752
alt=635.276123
<<< onGnssLocationCb: numValidFixes:1 exceeds fixCnt:1
calling stopPosition and delete LCA client
```

Summary: Received one fix.

Sample sys log for standalone single shot fix is as follows:

```
LocSvc_LocationClientApi[1119]: startTrackingSync:1723] >>> StartTrackingReq Interval=1000
Distance=0, locReqEngTypeMask=0x0 rc=1
LocSvc_LocationClientApi[1119]: proc:2991] >-- onReceive Rcvd msg id: 5 E_LOCAPI_START_
TRACKING_MSG_ID, sockname: locapiservice, payload size: 148
LocSvc_LocationClientApi[1119]: proc:3034] <<< response message 5
LocSvc_LocationClientApi[1119]: proc:2991] >-- onReceive Rcvd msg id: 13 E_LOCAPI_LOCATION_
INFO_MSG_ID, sockname: locapiservice, payload size: 2752
LocSvc_LocationClientApi[1119]: proc:3217] <<< message = location info
LocSvc_LocationClientApi[1119]: proc:2991] >-- onReceive Rcvd msg id: 13 E_LOCAPI_LOCATION_
INFO_MSG_ID, sockname: locapiservice, payload size: 2752
LocSvc_LocationClientApi[1119]: proc:3217] <<< message = location info

LocSvc_LocApiBase[3504]: reportPosition:379]  flags: 0x217   source: 2   latitude: 17.429013
longitude: 78.379752   altitude: 635.276123   speed: 0.000000   bearing: 0.000000   accuracy:
3.790092   timestamp: 1708585393713   session status: 0   technology mask: 0x1   time bias unc
0.000029 msec   SV used in fix (gps/glo/bds/gal/qzss/navic) :  (0x20353162/0x0/0x0/0x40000/
0x0/0x0)
LocSvc_SystemStatus[3504]: eventPosition - lat=17.429013 lon=78.379752 alt=635.276123 speed=0.
```

```
000000

LocSvc_LocationClientApi[1119]: proc:2991] >-- onReceive Rcvd msg id: 13 E_LOCAPI_LOCATION_
INFO_MSG_ID, sockname: locapiservice, payload size: 2752
LocSvc_LocationClientApi[1119]: proc:3217] <<< message = location info

LocSvc_LocationClientApi[1119]: stopTrackingSync:1872] >>> StopTrackingReq rc=1
LocSvc_LocationClientApi[1119]: proc:1539] >>> DeregisterReq rc=1
```

8.3 Sample logs for location tracking

Sample command for location tracking (detailed location report) with verbose logging enabled is as follows:

```
location_client_api_testapp -a -V -i 1000 -t 30 -r 7 -s g -l 10
```

Sample output for verbose logging is as follows:

```
group ids: diag locclient
Group locclient = 4021
tiemout: 30
report type: 7
session type: g
fix cnt: 10
auto run 1, deleteAll 0, delete mask 0x0, session type 2,outputEnabled 1,
detailedOutputEnabled 1 routeToNMEAPort 0pid: 1519
<<< onCapabilitiesCb mask=0xff
<<< onCapabilitiesCb mask string=capabMask: 0xff (TIME_BASED_TRACKING | DIST_BASED_TRACKING)
<<< onResponseCb err=0
<<< onGnssSvCb cnt=1
<<< svId: 5
type: GPS
cN0Dbhz: 42.700001
elevation: 40.000000
azimuth: 185.000000
gnssSvOptionsMask: 0xdf (EPH | ALM | USED_IN_FIX | CARRIER_FREQ | SIG_TYPES | ELEVATION |
AZIMUTH)
carrierFrequencyHz: 1575420032.000000
gnssSignalTypeMask: 0x1 (GPS_L1CA)
basebandCarrierToNoiseDbHz: 39.100001
gloFrequency: 0

<<< onGnssLocationCb cnt=(0/1): flags: 0xf71 (LAT_LON | ACCURACY | VERT_ACCURACY | SPEED_
ACCURACY |  |  |  | )
timestamp: 1711532127182
latitude: 17.429032
longitude: 78.379732
altitude: 0.000000
speed: 0.000000
bearing: 0.000000
horizontalAccuracy: 2093.023193
verticalAccuracy: 0.000000
speedAccuracy: 9.548413
bearingAccuracy: 0.000000
```

Sample console log for location tracking is as follows:

```
group ids: diag locclient
Group locclient = 4021
interval: 1000
tiemout: 30
report type: 1
session type: g
fix cnt: 1200
auto run 1, deleteAll 0, delete mask 0x0, session type 2,outputEnabled 1,
detailedOutputEnabled 0 routeToNMEAPort 0pid: 3608
<<< onCapabilitiesCb mask=0xff
<<< onCapabilitiesCb mask string=capabMask: 0xff (TIME_BASED_TRACKING | TIME_BASED_BATCHING |
DIST_BASED_TRACKING | DIST_BASED_BATCHING | GEOFENCE | OUTDOOR_TRIP_BATCHING | GNSS_
MEASUREMENTS | CONSTELLATION_ENABLE)
<<< onResponseCb err=0
<<< onGnssLocationCb cnt=(1/1): time=1707456564547 mask=0xfff lat=17.429110 lon=78.379608
alt=645.985718
<<< onGnssLocationCb cnt=(2/2): time=1707456565679 mask=0xfff lat=17.429109 lon=78.379605
alt=646.641113
<<< onGnssLocationCb cnt=(3/3): time=1707456566677 mask=0xf77 lat=17.429126 lon=78.379681
alt=637.882446
<<< onGnssLocationCb cnt=(4/4): time=1707456567000 mask=0xf77 lat=17.429102 lon=78.379698
alt=637.032471

<<< onGnssLocationCb cnt=(1199/1199): time=1707457762000 mask=0xf77 lat=17.429111 lon=78.
379706 alt=634.181519
<<< onGnssLocationCb cnt=(1200/1200): time=1707457763000 mask=0xf77 lat=17.429112 lon=78.
379707 alt=634.100830
<<< onGnssLocationCb: numValidFixes:1200 exceeds fixCnt:1200
calling stopPosition and delete LCA client
```


Sample sys log for location tracking is as follows:

```
LocSvc_GnssAdapter[1354]: addClientCommand]: client 0xb5ddb4e0
LocSvc_GnssAdapter[1354]: startTrackingCommand]: client 0xb5ddb4e0 id 3 minInterval 1000
minDistance 0 mode 0 powermode 2 tbm 0
LocSvc_GnssAdapter[1354]: startTimeBasedTracking:3507] minInterval 1000 minDistance 0 mode 0
powermode 2 tbm 0

LocSvc_ApiV02[1354]: void LocApiV02::setOperationMode(GnssSuplMode):10653]: operationMode
STANDALONE
LocSvc_GnssAdapter[1354]: checkUpdateDgnssNtrip:8188] isInSession 1 mDgnssState 0x0
isLocationValid 0
LocSvc_GnssAdapter[1354]: reportResponse]: client 0xb5ddb4e0 id 3 err 0
LocSvc_ApiV02[1354]: reportPosition:3233] gnssSvIdUsed 3 gnssSignalTypeMask 0x1

LocSvc_ApiV02[1354]: reportPosition:3445] report position mask: 0x106187fc4fb3, dgnss info:
0x0 0 0 0 0,
LocSvc_GnssAdapter[1354]: reportPositionEvent:4155] reportPositionEvent, eng type: 1, unpro 0,
sess status 1 msInWeek -1

LocSvc_GnssAdapter[1354]: reportPosition:4445] reportToAllClients 0, reportToAnyClient 1,
status 1, eng type 1, precise location enabled 0
LocSvc_LocAdapterBase[1354]: virtual void loc_core::LocAdapterBase::reportPositionEvent(const
UlpLocation&, const GpsLocationExtended&, loc_sess_status, LocPosTechMask,
GnssDataNotification*, int): default implementation invoked

LocSvc_LocApiBase[1354]: flags: 0x217   source: 2   latitude: 17.429072   longitude: 78.379735
```

```
  altitude: 640.299561   speed: 0.000000   bearing: 0.000000   accuracy: 3.845669   timestamp:
1690562503000 Session status: 0  Technology mask: 0x1, time bias unc 0.000022 msec  SV used in
fix (gps/glo/bds/gal/qzss/navic) :(0x661081ce/0x403807/0x84480000/0xa01800196/0x0/0x0)

LocSvc_GnssAdapter[1354]: reportPositionEvent:4155] reportPositionEvent, eng type: 1, unpro 0,
sess status 0 msInWeek -1
LocSvc_SystemStatus[1354]: eventPosition - lat=17.429072 lon=78.379735 alt=640.299561 speed=0.
000000
LocSvc_GnssAdapter[1354]: logLatencyInfo:4382] mGnssLatencyInfoQueue.size is 0
```

8.4 Sample logs for location batching

Note: In this release, only time-based location batching is supported and distance-based batching is not supported.

Sample console log for location batching is as follows:

```
location_client_api_testapp

group ids: diag locclient
Group diag = 2901
Group locclient = 4021
auto run 0, deleteAll 0, delete mask 0x0, session type 0,outputEnabled 1,
detailedOutputEnabled 0 routeToNMEAPort 0<<< onCapabilitiesCb mask=0xff
<<< onCapabilitiesCb mask string=capabMask: 0xff (TIME_BASED_TRACKING | TIME_BASED_BATCHING |
DIST_BASED_TRACKING | DIST_BASED_BATCHING | GEOFENCE | OUTDOOR_TRIP_BATCHING | GNSS_
MEASUREMENTS | CONSTELLATION_ENABLE)

b 1000 10
execute command b 1000 0
start routine batching with interval 1000 msec, distance 0 meters
<<< onResponseCb err=0
diag: Diag_LSM_Init: invoked for pid: 3723 with init_count: 0
diag:successfully connected to socket 8
diag: Diag_LSM_Init: done for pid: 3723 with init_count: 1
<<< onBatchingCb, batching status: 1, pos cnt 20<<< onBatchingCb time=1690823085700 mask=0x177
lat=17.429066 lon=78.379707 alt=637.070984
<<< onBatchingCb time=1690823086700 mask=0x177 lat=17.429071 lon=78.379726 alt=636.896423
<<< onBatchingCb time=1690823088100 mask=0x177 lat=17.429073 lon=78.379725 alt=636.403137
<<< onBatchingCb, batching status: 1, pos cnt 20<<< onBatchingCb time=1690823106000 mask=0x1ff
lat=17.429088 lon=78.379714 alt=638.921814

calling stopPosition and delete LCA client
summary: received 20 fixes
```


Sample sys log for location batching is as follows:

```
LocSvc_BatchingAdapter[1271]: BatchingAdapter::BatchingAdapter()]: Constructor
LocSvc_BatchingAdapter[1271]: void BatchingAdapter::readConfigCommand()]:
LocSvc_BatchingAdapter[1271]: void BatchingAdapter::setConfigCommand()]:
LocSvc_BatchingAdapter[1271]: virtual void BatchingAdapter::readConfigCommand()::
MsgReadConfig::proc() const]: batchSize 20 tripBatchSize 600 batchingAccuracy 1
batchingTimeout 0
LocSvc_BatchingAdapter[1271]: void BatchingAdapter::restartSessions()]:
```

```
LocSvc_BatchingAdapter[1271]: uint32_t BatchingAdapter::startBatchingCommand(LocationAPI*,
BatchingOptions&)]: client 0xb5e6db40 id 3 minInterval 10000 minDistance 0 mode 0 Batching
Mode 0
LocSvc_BatchingAdapter[1271]: void BatchingAdapter::reportResponse(LocationAPI*,
LocationError, uint32_t)]: client 0xb5e6db40 id 3 err 0
LocSvc_BatchingAdapter[1271]: virtual void BatchingAdapter::reportLocationsEvent(const
Location*, size_t, BatchingMode)]: count 20 batchMode 0

LocSvc_ApiV02[1271]: void LocApiV02::readModemLocations(Location*, size_t, BatchingMode, size_
t&)] Read out 5 batched locations from modem.
LocSvc_ApiV02[1271]: void LocApiV02::readModemLocations(Location*, size_t, BatchingMode, size_
t&)] count 5.
LocSvc_ApiV02[1271]: void globalRespCb(locClientHandleType, uint32_t,
locClientRespIndUnionType, uint32_t, void*):275] client = 0xb6456a70, resp id = 121, client
cookie ptr = 0xb647a170
LocSvc_HalDaemon[1271]: onBatchingCb:827] --< onBatchingCb, client name /dev/socket/loc_
client/toclient_location_client_api_testapp.3568.1
LocSvc_HalDaemon[1271]: onBatchingCb:836] Batch count: 20
LocSvc_LocationApiPbMsgConv[1271]: convertLocAPIBatchingNotifMsgToPB:4072] LocApiPB:
locApiBatchNotifMsg - BatchStat: 1, Loc count:20

LocSvc_BatchingAdapter[1271]: virtual void BatchingAdapter::stopClientSessions(LocationAPI*,
bool)]: client 0xb5e6db40
LocSvc_BatchingAdapter[1271]: void BatchingAdapter::reportResponse(LocationAPI*,
LocationError, uint32_t)]: client 0xb5e6db40 id 3 err 0
```

8.5 Sample logs for location geofencing

Sample console log for location geofencing is as follows:

```
<<< onCapabilitiesCb mask=0xcff
<<< onCapabilitiesCb mask string=capabMask: 0xcff (TIME_BASED_TRACKING | TIME_BASED_BATCHING |
DIST_BASED_TRACKING | DIST_BASED_BATCHING | GEOFENCE | OUTDOOR_TRIP_BATCHING | GNSS_
MEASUREMENTS | CONSTELLATION_ENABLE | GNSS_SINGLE_FREQ | GNSS_MULTI_FREQ)
addGeofences 39.0072505228 116.0093202146 200 11 1000 1
execute command addGeofences 39.0072505228 116.0093202146 200 11 1000 1

usage: addGeofences [lat lon radius breachType responsiveness dwellTime ] ...
addGeofences, index: 0, latitude: 39.007252, longitude: 116.009323, radiusM: 200.000000,
breachType: 11, responsivenessMs: 1000, dwellTimeSec: 1
currently there are 1 geofences available, please input index and the latitude/longitude/
radius/breachType/responsiveness/dwelltime
<<< onCollectiveResponseCb, geofence cnt: 1
<<< onCollectiveResponseCb, lat=39.007252 lon=116.009323 rad=200.000000, breachType: 11,
responsiveness: 1000, dwellTime: 1, response: 0
<<< onGeofenceBreachCb, breach type: 2, gf cnt: 1, timestamp: 1694163617<<<
onGeofenceBreachCb, lat=39.007252 lon=116.009323 rad=200.000000, responsiveness: 1000,
dwellTime: 1
diag: Diag_LSM_Init: invoked for pid: 2658 with init_count: 0
diag:successfully connected to socket 8
diag: Diag_LSM_Init: done for pid: 2658 with init_count: 1
<<< onGeofenceBreachCb, breach type: 8, gf cnt: 1, timestamp: 1694163618<<<
onGeofenceBreachCb, lat=39.007252 lon=116.009323 rad=200.000000, responsiveness: 1000,
dwellTime: 1
<<< onGeofenceBreachCb, breach type: 1, gf cnt: 1, timestamp: 1694163628<<<
onGeofenceBreachCb, lat=39.007252 lon=116.009323 rad=200.000000, responsiveness: 1000,
dwellTime: 1
<<< onGeofenceBreachCb, breach type: 2, gf cnt: 1, timestamp: 1694163661<<<
onGeofenceBreachCb, lat=39.007252 lon=116.009323 rad=200.000000, responsiveness: 1000,
dwellTime: 1
```

```
<<< onGeofenceBreachCb, breach type: 8, gf cnt: 1, timestamp: 1694163662<<<
onGeofenceBreachCb, lat=39.007252 lon=116.009323 rad=200.000000, responsiveness: 1000,
dwellTime: 1
```

Sample sys log for location geofencing is as follows:

```
LocSvc_LocationClientApi[2658]: addGeofences:2132] >>> AddGeofencesReq count=1 rc=1
LocSvc_HalDaemon[1329]: processClientMsg:333] >-- onReceive Rcvd msg id: 24 E_LOCAPI_ADD_
GEOFENCES_MSG_ID, remote client: /dev/socket/loc_client/toclient_location_client_api_testapp.
2658.1, payload size: 1272
LocSvc_LocationApiPbMsgConv[1329]: pbConvertToGeofenceOption:6323] LocApiPB: pbGfOpt -
BreachTypeMask:b Resp:1000, DwellTime:1
LocSvc_LocationApiPbMsgConv[1329]: pbConvertToGeofenceInfo:6337] LocApiPB: pbGfInfo - Lat:39.
007252, Lon:116.009323, Rad:200.000000
LocSvc_GeofenceAdapter[1329]: uint32_t* GeofenceAdapter::addGeofencesCommand(LocationAPI*,
size_t, GeofenceOption*, GeofenceInfo*)]: client 0xb5e96080 count 1
LocSvc_ApiV02[1329]: LocApiV02::addGeofence(uint32_t, const GeofenceOption&, const
GeofenceInfo&, loc_core::LocApiResponseData<loc_core::LocApiGeofenceData>*)::<lambda()>]: lat=
  39.01 long=  116.01 radius   200.00 breach=11 respon=1000 dwell=1
[1329]: ---> locClientSendReq line 2462 QMI_LOC_ADD_CIRCULAR_GEOFENCE_REQ_V02

LocSvc_HalDaemon[1329]: addGeofences:395] start new geofence sessions: 0xb6416ef0
LocSvc_HalDaemon[1329]: addGeofences:1272] >-- add geofences
LocSvc_ApiV02[1329]: <--- globalRespCb line 270 QMI_LOC_ADD_CIRCULAR_GEOFENCE_REQ_V02
LocSvc_GeofenceAdapter[1329]: void GeofenceAdapter::saveGeofenceItem(LocationAPI*, uint32_t,
uint32_t, const GeofenceOption&, const GeofenceInfo&)]: hwId 0 client 0xb5e96080 clientId 3
LocSvc_GeofenceAdapter[1329]: HAL | hwId  | mask | respon | latitude | longitude | radius |
paused |  Id  | client
LocSvc_GeofenceAdapter[1329]:     |     0 |   11 |   1000 |    39.01 |    116.01 | 200.00 |
  0 | 0003 | 0xb5e96080
LocSvc_GeofenceAdapter[1329]: void GeofenceAdapter::reportResponse(LocationAPI*, size_t,
LocationError*, uint32_t*)]: client 0xb5e96080 ids [3 ] errs [0 ]
LocSvc_HalDaemon[1329]: onCollectiveResponseCallback:571] <-- addGeofence resp pending=24
LocSvc_LocationClientApi[2658]: proc:2965] >-- onReceive Rcvd msg id: 24 E_LOCAPI_ADD_
GEOFENCES_MSG_ID, sockname: locapiservice, payload size: 156
LocSvc_ApiV02[1329]: <--- globalEventCb line 242 QMI_LOC_EVENT_GEOFENCE_BATCHED_BREACH_
NOTIFICATION_IND_V02

LocSvc_LBSApiV02[1329]: virtual void lbs_core::LBSApiV02::eventCb(locClientHandleType, uint32_
t, locClientEventIndUnionType):59] client = 0xb63e4e30, event id = 128, event name = QMI_LOC_
EVENT_GEOFENCE_BATCHED_BREACH_NOTIFICATION_IND_V02 payload = 0xb5f1c2ac
LocSvc_ApiV02[1329]: eventCb:7535] event id = 0x80, event name QMI_LOC_EVENT_GEOFENCE_BATCHED_
BREACH_NOTIFICATION_IND_V02
LocSvc_ApiV02[1329]: void LocApiV02::geofenceBreachEvent(const
qmiLocEventGeofenceBatchedBreachIndMsgT_v02*)]: latitude=   39.01 longitude=  116.01
LocSvc_ApiV02[1329]: void LocApiV02::geofenceBreachEvent(const
qmiLocEventGeofenceBatchedBreachIndMsgT_v02*)]: discrete hwID 0 breachType 1
LocSvc_LocAdapterBase[1329]: virtual void loc_core::LocAdapterBase::geofenceBreachEvent(size_
t, uint32_t*, Location&, GeofenceBreachType, uint64_t): default implementation invoked
LocSvc_GeofenceAdapter[1329]: virtual void GeofenceAdapter::geofenceBreachEvent(size_t,
uint32_t*, Location&, GeofenceBreachType, uint64_t)]: breachType 1 count 1 ids [0 ]
LocSvc_HalDaemon[1329]: onGeofenceBreachCb:885] --< onGeofenceBreachCallback

LocSvc_HalDaemon[1329]: onGeofenceBreachCb:899] Gf Breach Notif count: 1, breachType: 1
LocSvc_LocationClientApi[2658]: proc:2965] >-- onReceive Rcvd msg id: 29 E_LOCAPI_GEOFENCE_
BREACH_MSG_ID, sockname: locapiservice, payload size: 280
LocSvc_LocationClientApi[2658]: proc:3153] <<< message = geofence breach
```

```
LocClientApiDiag[2658]: log:79] GeofenceBreachDiagReport::log
LocSvc_ApiV02[1329]: <--- globalEventCb line 242 QMI_LOC_EVENT_GEOFENCE_BATCHED_DWELL_
NOTIFICATION_IND_V02
```

9 References

9.1 Related documents

Title	Number
Qualcomm Technologies, Inc.	
Qualcomm RB3 Gen 2 Development Kit Guide	80-70018-251
Qualcomm Linux Build Guide	80-70018-254
Qualcomm Linux Kernel Guide	80-70018-3

9.2 Acronyms and terms

Acronym or term	Definition
AFLT	Advanced forward link trilateration
AGC	Automatic gain control
API	Application programming interface
AProc	Application processor
BDS	BeiDou navigation satellite system
DC	Disaster and crisis report
DGNSS	Differential global navigation satellite system
DOP	Dilution of precision
FCN	Frequency channel number
Galileo	European global navigation satellite system
GDOP	Geometric dilution of precision
GLONASS	Russian global navigation satellite system
GNSS	Global navigation satellite system
GPS	Global positioning system
GTP	Global terrestrial positioning
HAL	Hardware abstraction layer
HDOP	Horizontal dilution of precision
ICD	Interface control documents
ISB	Intersignal bias
MProc	Modem processor
NavIC	Indian regional navigation satellite system
NMEA	National marine electronics association
OTA	Over-the-air
PDOP	Position dilution of precision
PVT	Position velocity and time
QMI	Qualcomm messaging interface
QoS	Quality of service
QZSS	Quasi-zenith satellite system
RF	Radio frequency
RTK	Real-time kinematic positioning
SBAS	Satellite-based augmentation system
SCP	Secure copy protocol

Acronym or term	Definition
SPE	Standard positioning engine
SV	Satellite vehicle
TBF	Time between fixes
TDOP	Time dilution of precision
TOD	Time of day
VDOP	Vertical dilution of precision
WWAN	Wireless wide area network

LEGAL INFORMATION

Your access to and use of this material, along with any documents, software, specifications, reference board files, drawings, diagnostics and other information contained herein (collectively this "Material"), is subject to your (including the corporation or other legal entity you represent, collectively "You" or "Your") acceptance of the terms and conditions ("Terms of Use") set forth below. If You do not agree to these Terms of Use, you may not use this Material and shall immediately destroy any copy thereof.

1) Legal Notice.

This Material is being made available to You solely for Your internal use with those products and service offerings of Qualcomm Technologies, Inc. ("**Qualcomm Technologies**"), its affiliates and/or licensors described in this Material, and shall not be used for any other purposes. If this Material is marked as "**Qualcomm Internal Use Only**", no license is granted to You herein, and You must immediately (a) destroy or return this Material to Qualcomm Technologies, and (b) report Your receipt of this Material to qualcomm.support@qti.qualcomm.com. This Material may not be altered, edited, or modified in any way without Qualcomm Technologies' prior written approval, nor may it be used for any machine learning or artificial intelligence development purpose which results, whether directly or indirectly, in the creation or development of an automated device, program, tool, algorithm, process, methodology, product and/or other output. Unauthorized use or disclosure of this Material or the information contained herein is strictly prohibited, and You agree to indemnify Qualcomm Technologies, its affiliates and licensors for any damages or losses suffered by Qualcomm Technologies, its affiliates and/or licensors for any such unauthorized uses or disclosures of this Material, in whole or part.

Qualcomm Technologies, its affiliates and/or licensors retain all rights and ownership in and to this Material. No license to any trademark, patent, copyright, mask work protection right or any other intellectual property right is either granted or implied by this Material or any information disclosed herein, including, but not limited to, any license to make, use, import or sell any product, service or technology offering embodying any of the information in this Material.

THIS MATERIAL IS BEING PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESSED, IMPLIED, STATUTORY OR OTHERWISE. TO THE MAXIMUM EXTENT PERMITTED BY LAW, QUALCOMM TECHNOLOGIES, ITS AFFILIATES AND/OR LICENSORS SPECIFICALLY DISCLAIM ALL WARRANTIES OF TITLE, MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, SATISFACTORY QUALITY, COMPLETENESS OR ACCURACY, AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MOREOVER, NEITHER QUALCOMM TECHNOLOGIES, NOR ANY OF ITS AFFILIATES AND/OR LICENSORS, SHALL BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY EXPENSES, LOSSES, USE, OR ACTIONS HOWSOEVER INCURRED OR UNDERTAKEN BY YOU IN RELIANCE ON THIS MATERIAL.

Certain product kits, tools and other items referenced in this Material may require You to accept additional terms and conditions before accessing or using those items.

Technical data specified in this Material may be subject to U.S. and other applicable export control laws. Transmission contrary to U.S. and any other applicable law is strictly prohibited.

Nothing in this Material is an offer to sell any of the components or devices referenced herein.

This Material is subject to change without further notification.

In the event of a conflict between these Terms of Use and the *Website Terms of Use* on www.qualcomm.com, the *Qualcomm Privacy Policy* referenced on www.qualcomm.com, or other legal statements or notices found on prior pages of the Material, these Terms of Use will control. In the event of a conflict between these Terms of Use and any other agreement (written or click-through, including, without limitation any non-disclosure agreement) executed by You and Qualcomm Technologies or a Qualcomm Technologies affiliate and/or licensor with respect to Your access to and use of this Material, the other agreement will control.

These Terms of Use shall be governed by and construed and enforced in accordance with the laws of the State of California, excluding the U.N. Convention on International Sale of Goods, without regard to conflict of laws principles. Any dispute, claim or controversy arising out of or relating to these Terms of Use, or the breach or validity hereof, shall be adjudicated only by a court of competent jurisdiction in the county of San Diego, State of California, and You hereby consent to the personal jurisdiction of such courts for that purpose.

2) Trademark and Product Attribution Statements.

Qualcomm is a trademark or registered trademark of Qualcomm Incorporated. Arm is a registered trademark of Arm Limited (or its subsidiaries) in the U.S. and/or elsewhere. The Bluetooth® word mark is a registered trademark owned by Bluetooth SIG, Inc. Other product and brand names referenced in this Material may be trademarks or registered trademarks of their respective owners.

Snapdragon and Qualcomm branded products referenced in this Material are products of Qualcomm Technologies, Inc. and/or its subsidiaries. Qualcomm patented technologies are licensed by Qualcomm Incorporated.