IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
MARSHALL DIVISION

RYAN HARDIN AND ANDREW HILL, **Plaintiffs,** **vs.** **SAMSUNG ELECTRONICS CO., LTD., AND SAMSUNG ELECTRONICS AMERICA, INC.,** **Defendants.**	**Civil Action No. 2:22-cv-___** **JURY TRIAL**

COMPLAINT FOR PATENT INFRINGEMENT

Plaintiffs Ryan Hardin and Andrew Hill file this Complaint for Patent Infringement against Samsung Electronics Co., Ltd. and Samsung Electronics America, Inc. (collectively, "Samsung"), and allege as follows:

THE PARTIES

1. Plaintiff Ryan Hardin is a citizen of Texas with a principal residence in Houston, Texas. Mr. Hardin is a technology professional with over 20 years of experience.

2. Plaintiff Andrew Hill is a citizen of Texas with a principal residence in Houston, Texas. Mr. Hill is a technology professional with over 20 years of experience.

3. Defendant Samsung Electronics Co., Ltd. ("SEC") is a Korean company with its principal place of business in Suwon, South Korea. SEC has an "Information Technology & Mobile Communications" division that is responsible for the design, manufacture, and sale of

cellular network infrastructure equipment and components thereof around the world and in the United States.

4. Defendant Samsung Electronics America, Inc. ("SEA") is a New York corporation with its principal place of business in Ridgefield Park, New Jersey and an established place of business in Plano, Texas. SEA is a wholly-owned subsidiary of SEC. SEA imports into the United States and sells in the United States, including in this District, cellular network infrastructure equipment and components thereof.

JURISDICTION AND VENUE

5. This is an action arising under the patent laws of the United States, 35 U.S.C. § 271. This Court has subject matter jurisdiction under 28 U.S.C. §§ 1331 and 1338(a).

6. Venue is proper in this judicial district under 28 U.S.C. §§ 1391 and 1400(b) because Defendant has committed acts of infringement and has a regular and established place of business in this District. SEA, SEC's wholly-owned subsidiary, maintains an office in Plano, Texas. (*See* "Where to Find Us," Samsung, https://www.sra.samsung.com/locations/). In 2018, SEA relocated its North Texas-based teams in Richardson to Legacy Central in Plano. SEA described this new Plano office as its "flagship North Texas campus" and "home to Samsung Electronics America's second biggest employee population in the U.S. across multiple divisions – Customer Care, Mobile, Mobile R&D and Engineering." (*See* "Samsung Electronics America to Open Flagship North Texas Campus," Samsung Newsroom U.S., https://news.samsung.com/us/samsung-electronics-america-open-flagship-north-texas-campus/ (last visited July 27, 2021)). In a press release, SEA said its "move to Plano further demonstrates the company's dedication to Texas and being an invested corporate citizen." (*Id.*)

7. This Court has personal jurisdiction over Samsung. Samsung has continuous and systematic business contacts with the State of Texas. Samsung, directly and/or through subsidiaries or intermediaries (including distributors, retailers, and others), conducts its business extensively throughout Texas, by shipping, distributing, offering for sale, selling, and advertising (including the provision of an interactive web page) its products and/or services in the State of Texas and the Eastern District of Texas. SEA maintains an office in Plano, Texas and is responsible for importing and selling smartphones, tablets, other mobile devices, and cellular network infrastructure equipment that operate on cellular networks in the United States. SEC and SEA regularly do business or solicit business, engage in other persistent courses of conduct, and/or derive substantial revenue from products and/or services provided to individuals in the State of Texas.

8. SEC and SEA, directly and/or through subsidiaries or intermediaries (including distributors, retailers, and others), have purposefully and voluntarily placed one or more products and/or services in the stream of commerce that practice the Asserted Patents with the intention and expectation that they will be purchased and used by consumers in the Eastern District of Texas. These products and/or services have been and continue to be purchased and used by operators/carriers in the Eastern District of Texas.

9. Upon information and belief, the Samsung products accused of infringement in this case are manufactured by or on behalf of SEC and SEA.

BACKGROUND OF THE INVENTORS

10. Inventors Ryan Hardin and Andrew Hill both grew up in Henderson, Texas. After college, Hardin and Hill started an IT services company called Pronet Solutions Corp. in 2002 in East Texas. Upon establishing the business after a few years, Hardin and Hill began to investigate

and discuss emerging and future business opportunities that would leverage their technological backgrounds and expertise.

11. In the 2006-2007 timeframe, Hardin and Hill were interested in the digital billboard business. At that time, digital billboards were relatively new, and Hardin and Hill saw an opportunity to grow a business for displaying content (e.g., advertisements) on digital billboards, which they could efficiently manage and control remotely. However, Hardin and Hill ultimately saw significant barriers to entry for getting into the digital billboard business, including costs, regulatory considerations, and governmental issues.

12. Then, in late 2007, Hardin was at a meeting in Henderson, and when getting in his car to go to his next appointment, he looked at his phone and thought—why should he and Hill risk the expense to erect digital billboards if much of the public might soon be carrying digital billboards in their pockets? Of course, at that time, smartphones were in a period of rapid development,[1] and while a few phones had GPS, access to the internet, and apps, they were certainly not as commonplace as today. Hardin and Hill were betting on the future. Going forward, Hardin and Hill continued to fine-tune the implementation of their invention and applied for their first patent on May 1, 2009.

13. Separate from the patented invention, not only did Hardin and Hill realize a technical solution was needed—but a platform was also needed for the technology to be marketed to consumers. In this same timeframe, Hardin and Hill coincidentally received a phone call from

[1] For example, upon information and belief, the first version of the Samsung Galaxy smartphone was released in June 2009, and Google launched the Android Market in October 2008 as a way for users to download apps for Android-based phones (e.g., Samsung Galaxy).

the Henderson Area Chamber of Commerce, which requested a website be designed—but as cheaply as possible. Hardin and Hill ultimately saw this as an opportunity for the platform they were looking for. As a result, Hardin and Hill decided they would develop and host the website for free for the Henderson Area Chamber of Commerce, but then ultimately, build free websites for all Chambers of Commerce using the same platform. Then, through the access the chambers had with businesses, the websites could get advertising revenue—which is how Hardin and Hill could ultimately be compensated. Moreover, once setting up this network of chambers of commerce was complete, eventually the patented technology could also be marketed and used to create location-aware content delivery on mobile devices as they envisioned an increased demand from businesses for location-aware apps and from the public for mobile devices. Hardin and Hill called the platform "Chamber Planet."

14. Hardin and Hill spent years developing Chamber Planet and started to get numerous chambers of commerce on board. At one point, Hardin and Hill were working with the Henderson Area Chamber, Jacksonville Chamber, Bullard Area Chamber, Pittsburgh Camp County Chamber, and the Greater Marshall Chamber of Commerce. In fact, Hardin and Hill negotiated the purchase of the MarshallTexas.com domain from a third party and provided it to the Greater Marshall Chamber of Commerce. As more chambers got on board, more challenges arose, including that the chambers also desired an integrated billing system with their websites—which was complex and took significant time to develop.

15. On or around 2012, however, while Hardin and Hill were still developing and growing the Chamber Planet platform, their separate business Pronet also began to experience some challenges, largely due to the business disruptions created by the financial crisis. Hardin and

Hill had to focus more on Pronet, which was the business that provided revenue to them—and this slowed the growth of Chamber Planet. In addition, numerous members of Hardin's family began to have serious health issues. Hardin's wife had medical issues in pregnancies which required multiple surgeries and then she later developed cancer. Around the same time, Hardin's mother fell and broke her back, and Hardin's father required a lung transplant. Hardin needed to prioritize his family over the continued development of Chamber Planet for a period of time.

16. During this period of time, however, Hardin and Hill noticed that mobile devices were starting to utilize Hardin and Hill's invention. Moreover, these companies developing the mobile devices and related software and services had significantly more financial resources that enabled them to commercialize the invention much more quickly than Hardin or Hill could. As a result, Hardin and Hill then decided to focus more on protecting the intellectual property they had developed, while continuing to look out for opportunities to bounce back and continue to build the platform they had invented.

<u>THE ASSERTED PATENTS</u>

17. Plaintiffs are the owners in right, title, and interest in and to multiple United States patents and patent applications, including U.S. Patent No. 8,433,296 (the "'296 Patent) and U.S. Patent No. 8,977,247 (the "'247 Patent") (collectively, the "Asserted Patents"). The Asserted Patents are valid and enforceable, and the inventions claimed in the Asserted Patents were novel, non-obvious, unconventional, and non-routine at least as of May 1, 2009.

18. The '296 Patent, entitled "Exclusive Delivery of Content within Geographic Areas," was duly and legally issued to inventors Ryan Hardin and Andrew Hill on April 30, 2013.

Plaintiffs own the entire right, title, and interest in the ’296 Patent and are entitled to sue for past and future infringement. A true and correct copy of the ’296 Patent is attached as Exhibit A.

19. The ’247 Patent, entitled “Exclusive Delivery of Content within Geographic Areas,” was duly and legally issued to inventors Ryan Hardin and Andrew Hill on March 10, 2015. Plaintiffs own the entire right, title, and interest in the ’247 Patent and are entitled to sue for past and future infringement. A true and correct copy of the ’247 Patent is attached as Exhibit B.

OVERVIEW OF THE INVENTIONS

20. Mobile applications (“mobile apps” or “apps”) are computer programs designed to run on mobile devices. The advent and proliferation of apps for mobile devices significantly increased the opportunities for providing content to device users. In particular, the ability of mobile phones to monitor and make decisions based on the dynamic location of the user created additional opportunities for providing location-based content, including advertisements and other data or information.

21. Apps had no precise analog in physical media such as newspapers or magazines. Nor did physical media have the ability to determine a person’s location and proximity to other points of interest at any given time, such as when a user is moving from one location to another, or the length of time that a user spends in a certain location. Indeed, the location of a mobile device may change rapidly depending on, for example, whether the user is stationary or in a moving vehicle.

22. With these new opportunities came new technological challenges, including how to enable location-dependent features and functionality for apps on a mobile device while constrained by the limited computing power and storage of a mobile device. For instance, location

information can be obtained by apps on a mobile device in a number of suitable ways, including using a Global Positioning System (GPS) component on the mobile device. One way would be for each app to communicate with the GPS component to determine the location of the mobile device and deliver location-dependent content. Each app requiring location information would need to be configured to independently obtain location information. As such, if, for example, 10 mobile apps polled a GPS component for a location, the GPS component would need to obtain location information (e.g., through satellites) 10 different times.

23. However, given that mobile devices may include dozens of apps installed and simultaneously running on a mobile device, having apps independently requesting location information would be processor intensive and not scalable, especially for the mobile device to efficiently allow multitasking when several apps are simultaneously running in the foreground/background. This implementation could tie up the limited resources available on the mobile phone and also deplete the mobile phone's battery at a faster rate.

24. Plaintiffs invented a technological solution that solves these technical problems by implementing a centralized service for tracking and monitoring the location of the mobile device on behalf of apps on the mobile device. Apps can register with the centralized service to receive its benefits. The claimed inventions provide a mechanism whereby an app can register with the centralized service and request to be notified about events relating to a particular geographic area of interest, which is defined by a virtual perimeter. One kind of geographic area of interest is now commonly referred to as a "geofence" (the term is used herein as an exemplary geographic area of interest but not to suggest exact equivalence of the terms). As part of the request to be notified about events relating to a particular geofence, an app provides a unique identifier corresponding

to the app and a geofence. The centralized service receives and stores identifiers to keep track of app/geofence pairings for registered apps. The centralized service monitors the location of the mobile device and alerts the appropriate app(s), based on the identifier, when there is an event associated with a particular geofence (e.g., when the mobile device enters or exits a geofence registered to one or more apps on the mobile device, when the mobile device remains in a geofence for a predetermined amount of time). The alert from the centralized service includes the identifier provided by the app. Apps can advantageously limit the frequency of alerts received from the centralized service by, for example, specifying the time of day a mobile device must be in a particular area. In response to the alert, the app can, for example, display location-based information for the user. As such, for example, information shown is customized based on the mobile device's location, including based on the mobile device moving to a different location and/or the time of day or duration it remains within a particular area and/or the type of app receiving the alert. Notably, defining alerts and triggering content delivery based on geofence events was not feasible for physical media such as newspapers and the like, which cannot respond to dynamic movement.

25. By offloading the polling and monitoring of the mobile device's location from the individual apps to the centralized service, the claimed inventions improve the overall functioning of the mobile devices as well as the individual apps. As a result, the claimed inventions improve the overall functioning of the mobile device itself by increasing processing efficiency and reducing battery drain, as well as the functioning of the individual apps, especially when several apps are running simultaneously in the foreground or background. Revisiting the example described above, the centralized service could meet the needs of 10 apps by obtaining location information once for

all 10 of the apps. The centralized service, which is distinct from the apps on the mobile device, could be a part of the operating system (OS) of the mobile device or it may be a separate component of the mobile device or another system. By using a centralized service, the individual apps can take advantage of any additional information known to the mobile device (e.g., sensor data such as an accelerometer), thereby improving the functionality of the device and apps.

COUNT I: CLAIM FOR PATENT INFRINGEMENT OF THE ’296 PATENT

26. Plaintiffs repeat and reallege the allegations in paragraphs 1-25 as if fully set forth herein.

27. Samsung has infringed, contributed to the infringement of, and/or induced infringement of the ’296 Patent by making, using, selling, offering for sale, or importing into the United States, or by intending that others make, use, import into, offer for sale, or sell in the United States, products and/or methods covered by one or more claims of the ’296 Patent including, but not limited to, mobile devices and components thereof.

28. Upon information and belief, Samsung’s mobile devices, including smartphones, tablets, and watches, that infringe one or more claims of the ’296 Patent include at least the following products, as well as products with reasonably similar functionality:

- Galaxy S series, Galaxy Note series, Galaxy Z Flip series, Galaxy Fold series, and Galaxy A series smartphones;
- Galaxy Tab S series, Galaxy Tab A series, and Galaxy Book S series tablets; and

- Galaxy Watch 3 series, Galaxy Watch 3 Titanium series, Galaxy Watch Active2 series, Galaxy Watch Active series, Galaxy Watch series, and Galaxy Fit2 series watches.

Upon information and belief, these products are offered in several varieties, including different options for size, display, processor, storage, and battery. The products listed are exemplary, and Plaintiffs will be able to provide a more comprehensive list after discovery. Upon information and belief, Samsung also infringes one or more claims of the '296 Patent through its use of geofencing or location information in its advertising or content delivery efforts or its own applications it develops.

29. For example, upon information and belief, Samsung's mobile devices infringe at least claim 6 of the '296 Patent. Samsung makes, uses, sells, offers for sale, imports, exports, supplies, or distributes within the United States its mobile devices and thus directly infringes the '296 Patent.

30. Samsung indirectly infringes the '296 Patent as provided by 35 U.S.C. § 271(b) by inducing infringement by others, such as manufacturers, resellers, and end-user customers in this District and throughout the United States. For example, direct infringement is the result of activities performed by manufacturers, resellers, or operators/carriers of Samsung's mobile devices, who, upon information and belief, perform each step of the claimed invention as directed by Samsung. Samsung received actual notice of the '296 Patent at least as early as the filing of this Complaint.

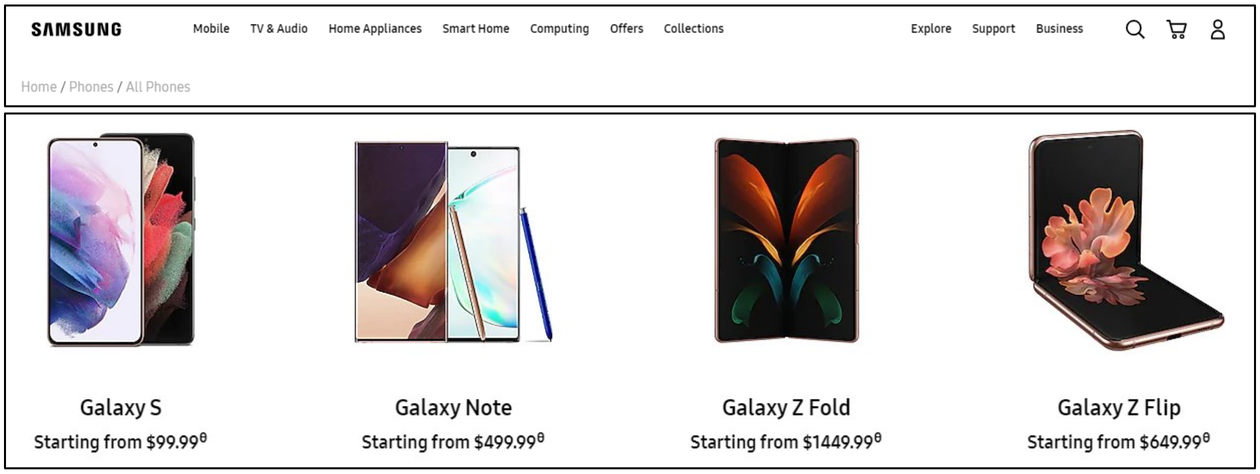
31. Upon information and belief, Samsung's affirmative acts of selling Samsung's mobile devices, causing Samsung's mobile devices to be manufactured, and providing directions,

instructions, schematics, diagrams, or designs to its manufacturers, resellers, or operators/carriers to make or use Samsung's mobile devices in a manner that directly infringes the '296 Patent induce infringement by others, such as manufacturers, resellers, or operators/carriers in this District and throughout the United States. Upon information and belief, through its manufacture and sales of Samsung's mobile devices, Samsung performed the acts that constitute induced infringement with knowledge or willful blindness that the induced acts would constitute infringement.

32. Samsung also indirectly infringes the '296 Patent by contributing to infringement by others, such as manufacturers, resellers, manufacturers, and operators/carriers, in accordance with 35 U.S.C. § 271(c) in this District and throughout the United States. Upon information and belief, direct infringement is the result of activities performed by manufacturers, resellers, or operators/carriers of Samsung's mobile devices.

33. Upon information and belief, Samsung's affirmative acts of selling Samsung's mobile devices and causing Samsung's mobile devices to be manufactured and sold contribute to Samsung's manufacturers, resellers, and operators/carriers making or using Samsung's mobile devices in a normal and customary way that infringes the '296 Patent. Upon information and belief, Samsung's mobile devices constitute the material part of Plaintiffs' patented invention, have no substantial non-infringing uses, and are known by Samsung to be especially made or especially adapted for use to infringe the '296 Patent.

34. Samsung provides a system comprising: a memory; a communications interface operably coupled to the memory; and at least one processor operably coupled to the memory and the communications interface, the processor configured to execute a program of instructions:


SAMSUNG Mobile TV & Audio Home Appliances Smart Home Computing Offers Collections Explore Support Business

Home / Phones / All Phones

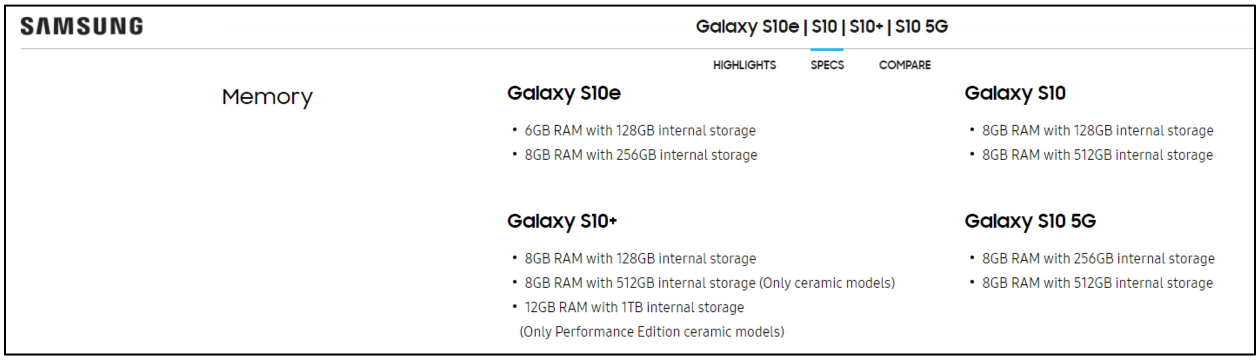
Galaxy S
Starting from $99.99⁸

Galaxy Note
Starting from $499.99⁸

Galaxy Z Fold
Starting from $1449.99⁸

Galaxy Z Flip
Starting from $649.99⁸

https://www.samsung.com/us/


SAMSUNG Galaxy S10e | S10 | S10+ | S10 5G

HIGHLIGHTS SPECS COMPARE

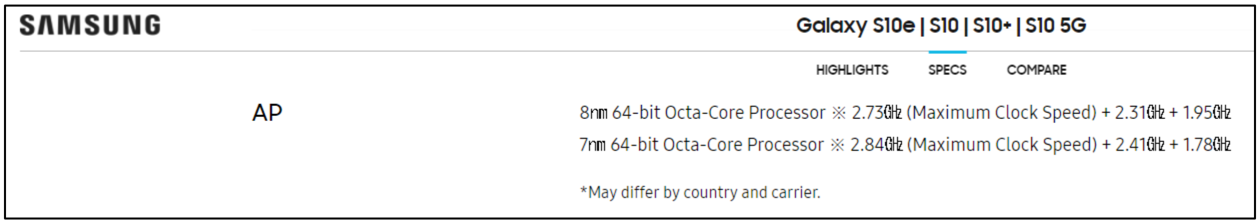
Memory

Galaxy S10e
- 6GB RAM with 128GB internal storage
- 8GB RAM with 256GB internal storage

Galaxy S10
- 8GB RAM with 128GB internal storage
- 8GB RAM with 512GB internal storage

Galaxy S10+
- 8GB RAM with 128GB internal storage
- 8GB RAM with 512GB internal storage (Only ceramic models)
- 12GB RAM with 1TB internal storage
  (Only Performance Edition ceramic models)

Galaxy S10 5G
- 8GB RAM with 256GB internal storage
- 8GB RAM with 512GB internal storage

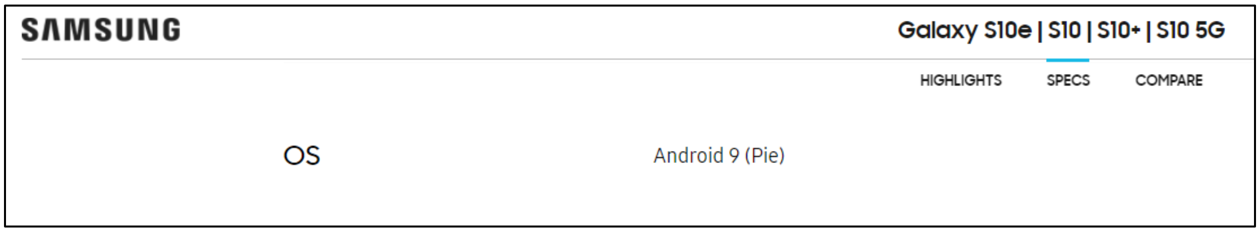

SAMSUNG Galaxy S10e | S10 | S10+ | S10 5G

HIGHLIGHTS SPECS COMPARE

AP

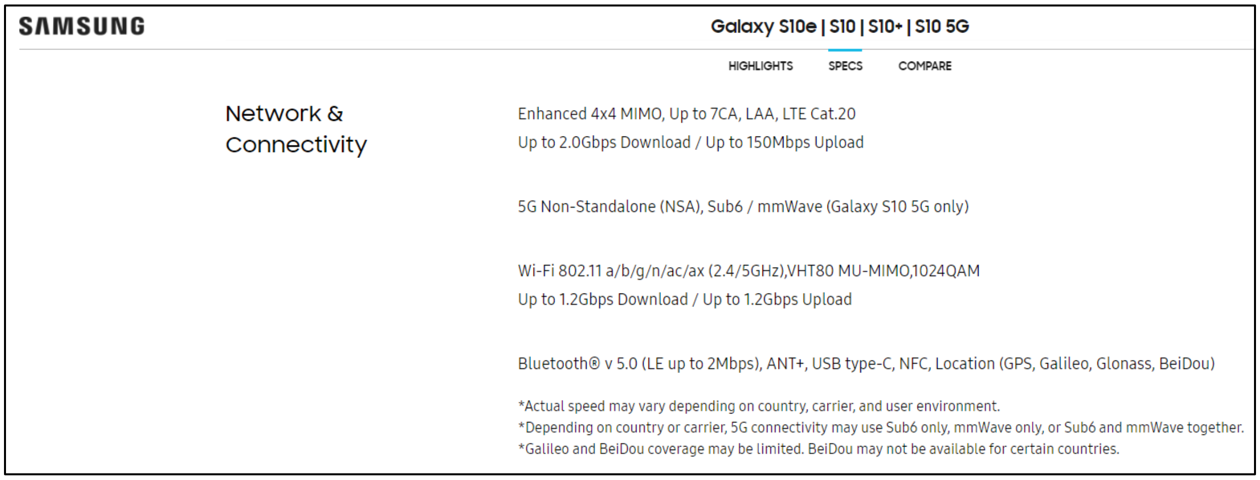
8㎚ 64-bit Octa-Core Processor ※ 2.73㎓ (Maximum Clock Speed) + 2.31㎓ + 1.95㎓
7㎚ 64-bit Octa-Core Processor ※ 2.84㎓ (Maximum Clock Speed) + 2.41㎓ + 1.78㎓

*May differ by country and carrier.


SAMSUNG Galaxy S10e | S10 | S10+ | S10 5G

HIGHLIGHTS SPECS COMPARE

OS Android 9 (Pie)


SAMSUNG Galaxy S10e | S10 | S10+ | S10 5G

HIGHLIGHTS SPECS COMPARE

Network & Connectivity

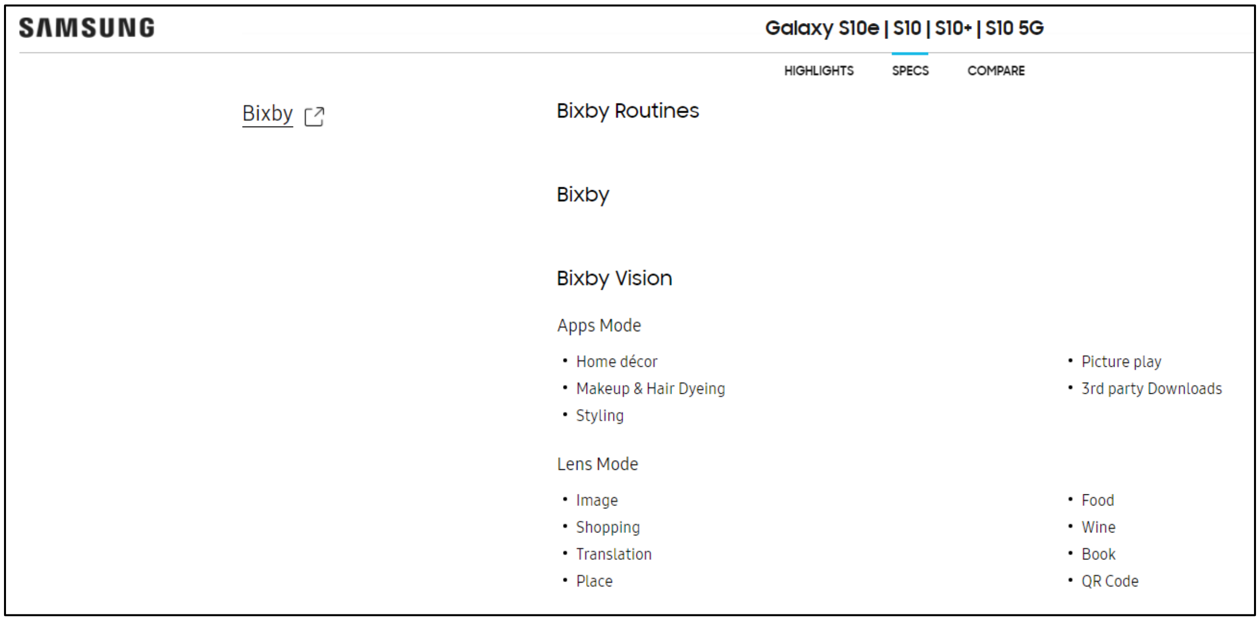
Enhanced 4x4 MIMO, Up to 7CA, LAA, LTE Cat.20
Up to 2.0Gbps Download / Up to 150Mbps Upload

5G Non-Standalone (NSA), Sub6 / mmWave (Galaxy S10 5G only)

Wi-Fi 802.11 a/b/g/n/ac/ax (2.4/5GHz),VHT80 MU-MIMO,1024QAM
Up to 1.2Gbps Download / Up to 1.2Gbps Upload

Bluetooth® v 5.0 (LE up to 2Mbps), ANT+, USB type-C, NFC, Location (GPS, Galileo, Glonass, BeiDou)

*Actual speed may vary depending on country, carrier, and user environment.
*Depending on country or carrier, 5G connectivity may use Sub6 only, mmWave only, or Sub6 and mmWave together.
*Galileo and BeiDou coverage may be limited. BeiDou may not be available for certain countries.


SAMSUNG Galaxy S10e | S10 | S10+ | S10 5G

HIGHLIGHTS SPECS COMPARE

Bixby

Bixby Routines

Bixby

Bixby Vision

Apps Mode

- Home décor
- Makeup & Hair Dyeing
- Styling
- Picture play
- 3rd party Downloads

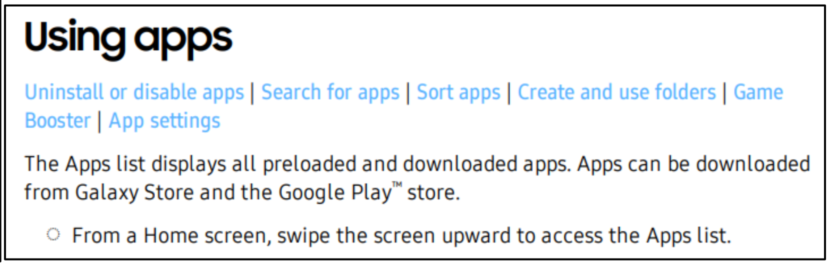
Lens Mode

- Image
- Shopping
- Translation
- Place
- Food
- Wine
- Book
- QR Code

https://www.samsung.com/global/galaxy/galaxy-s10/specs/

35. Samsung provides a system comprising a processor configured to execute a program of instructions, the program of instructions comprising: at least one instruction to register a plurality of application programs for use with a content delivery platform, wherein the plurality of registered application programs are each associated with at least one application program type; at least one instruction to establish a plurality of perimeters defining a plurality of geographic areas; at least one instruction to maintain at least one record indicating content delivery

reservations associating each of a plurality of sponsors with specific registered application program types and at least one of the plurality of geographic areas; at least one instruction to receive, from a sponsor, a request to obtain an interest in a selected one of the plurality of geographic areas; at least one instruction to provide the sponsor a response to the request, including as demonstrated in the exemplary text and images below:

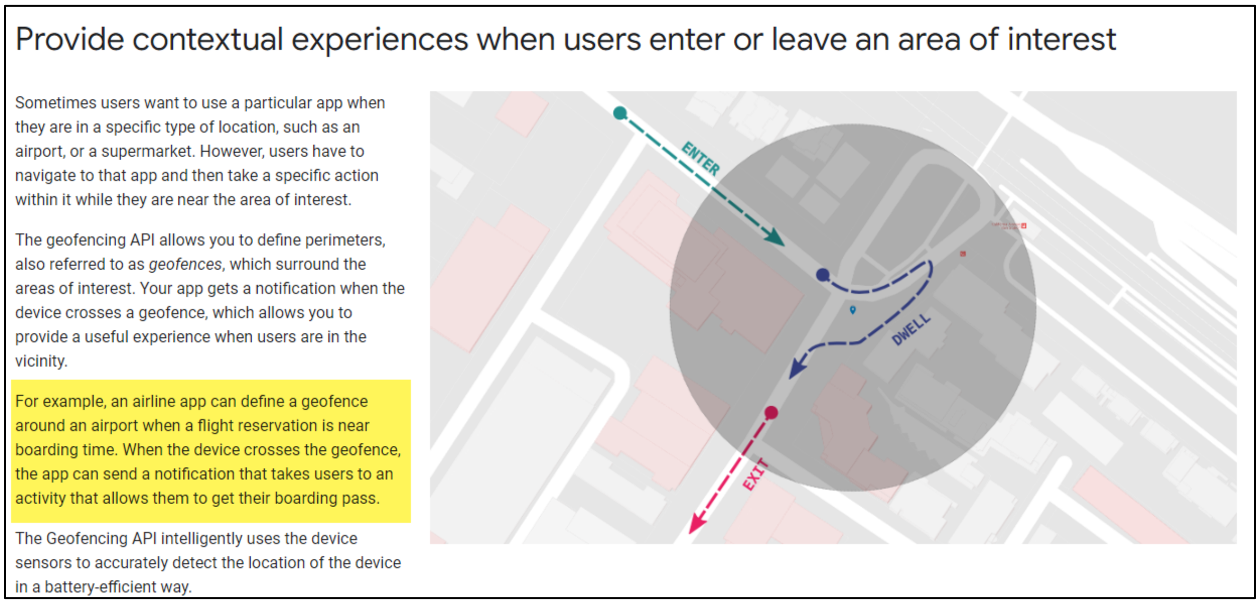

Using apps

Uninstall or disable apps | Search for apps | Sort apps | Create and use folders | Game Booster | App settings

The Apps list displays all preloaded and downloaded apps. Apps can be downloaded from Galaxy Store and the Google Play™ store.

◦ From a Home screen, swipe the screen upward to access the Apps list.

https://downloadcenter.samsung.com/content/UM/202103/20210304004310357/ATT_SM-G970U_G973U_G975U_EN_UM_R_11.0_022221_FINAL.pdf


Provide contextual experiences when users enter or leave an area of interest

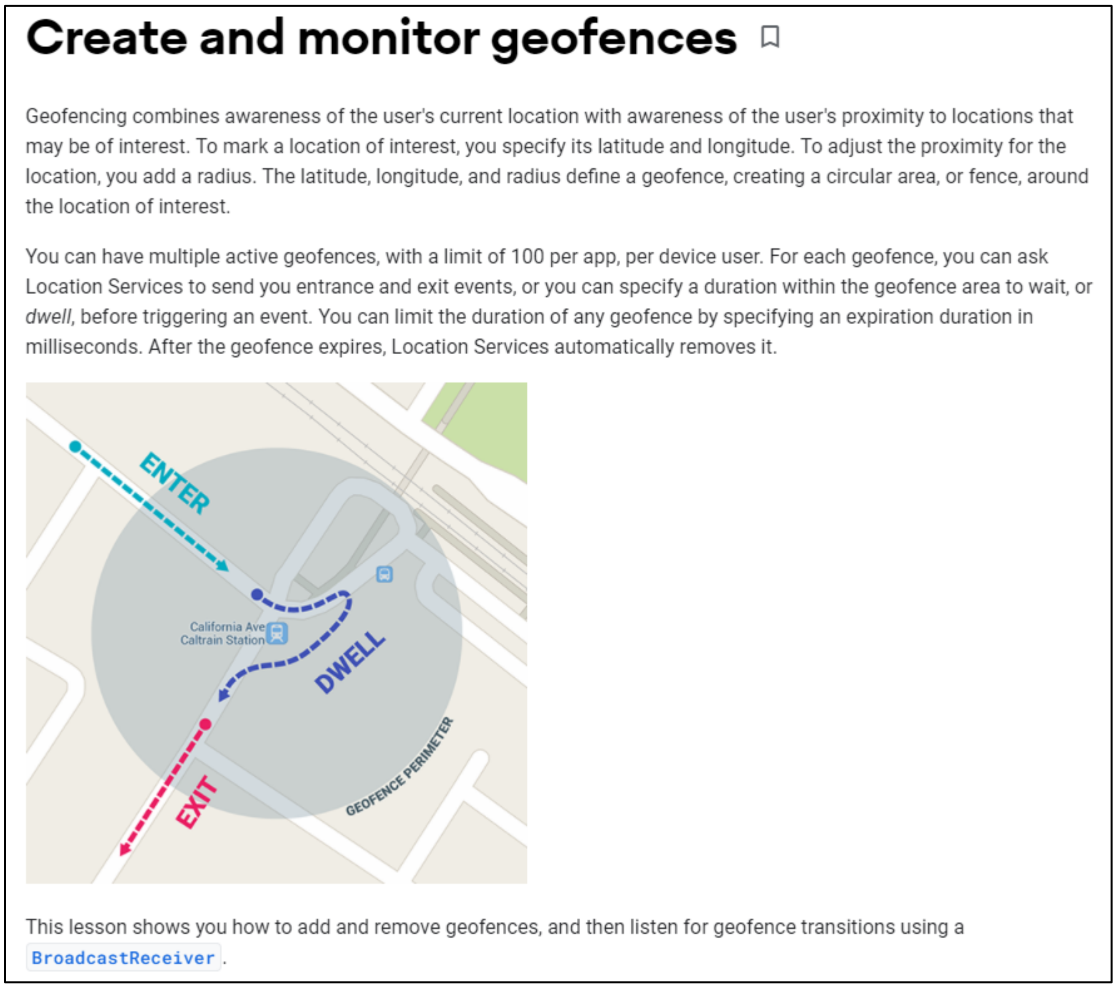
Sometimes users want to use a particular app when they are in a specific type of location, such as an airport, or a supermarket. However, users have to navigate to that app and then take a specific action within it while they are near the area of interest.

The geofencing API allows you to define perimeters, also referred to as *geofences*, which surround the areas of interest. Your app gets a notification when the device crosses a geofence, which allows you to provide a useful experience when users are in the vicinity.

For example, an airline app can define a geofence around an airport when a flight reservation is near boarding time. When the device crosses the geofence, the app can send a notification that takes users to an activity that allows them to get their boarding pass.

The Geofencing API intelligently uses the device sensors to accurately detect the location of the device in a battery-efficient way.

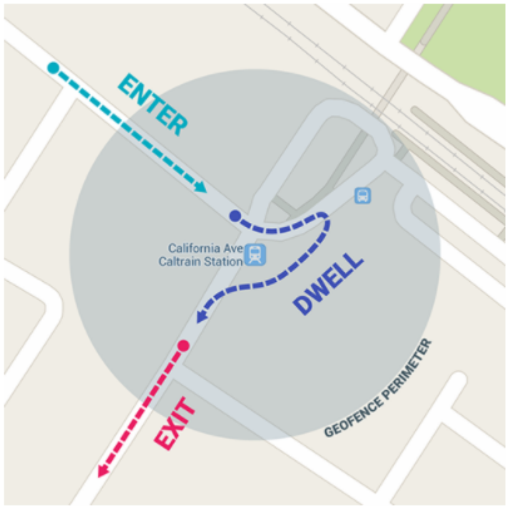
https://developers.google.com/location-context/geofencing#:~:text=The%20geofencing%20API%20allows%20you,users%20are%20in %20the%20vicinity;



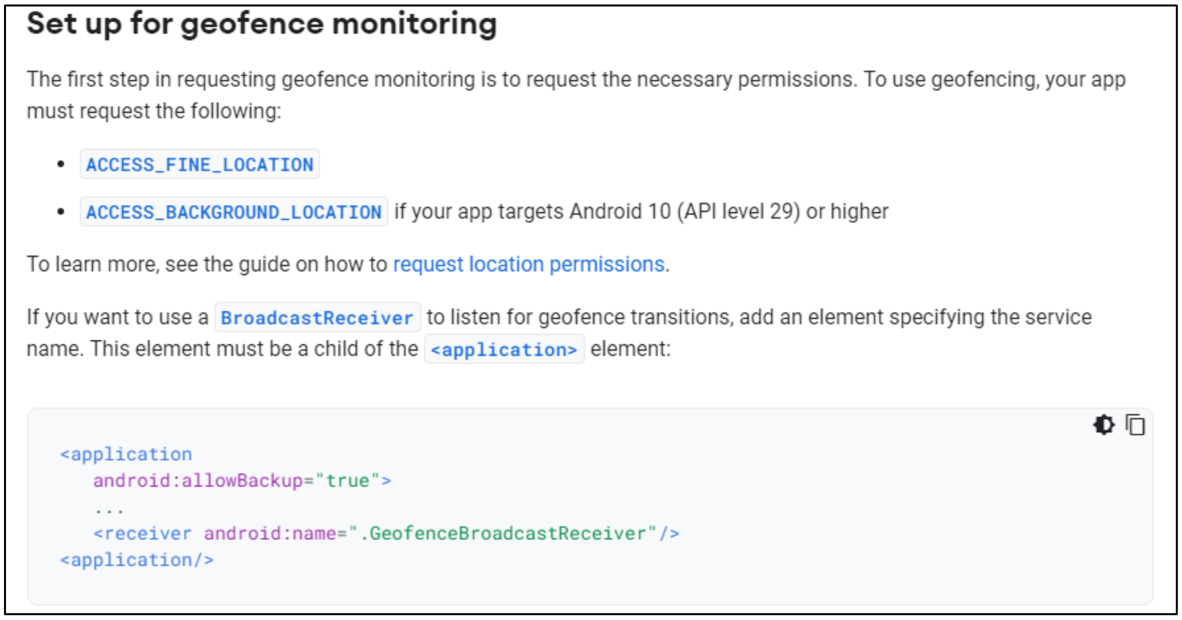
Create and monitor geofences

Geofencing combines awareness of the user's current location with awareness of the user's proximity to locations that may be of interest. To mark a location of interest, you specify its latitude and longitude. To adjust the proximity for the location, you add a radius. The latitude, longitude, and radius define a geofence, creating a circular area, or fence, around the location of interest.

You can have multiple active geofences, with a limit of 100 per app, per device user. For each geofence, you can ask Location Services to send you entrance and exit events, or you can specify a duration within the geofence area to wait, or *dwell*, before triggering an event. You can limit the duration of any geofence by specifying an expiration duration in milliseconds. After the geofence expires, Location Services automatically removes it.



This lesson shows you how to add and remove geofences, and then listen for geofence transitions using a `BroadcastReceiver`.



Set up for geofence monitoring

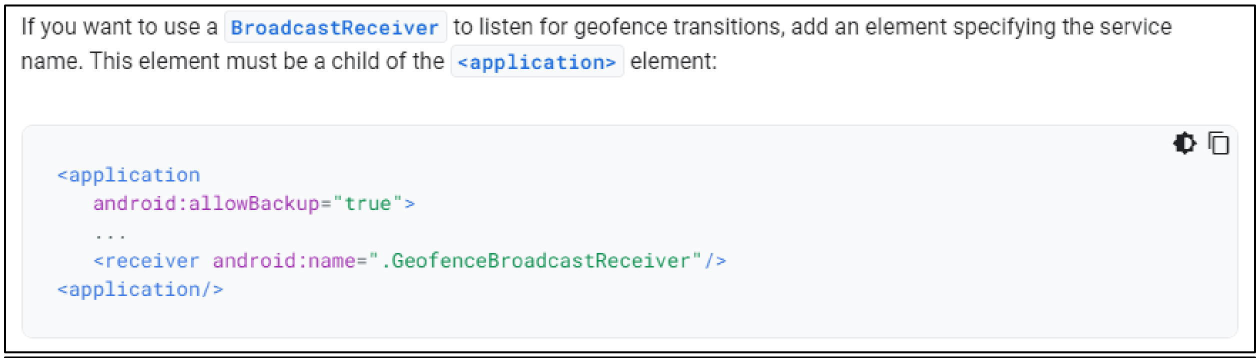
The first step in requesting geofence monitoring is to request the necessary permissions. To use geofencing, your app must request the following:

- `ACCESS_FINE_LOCATION`
- `ACCESS_BACKGROUND_LOCATION` if your app targets Android 10 (API level 29) or higher

To learn more, see the guide on how to request location permissions.

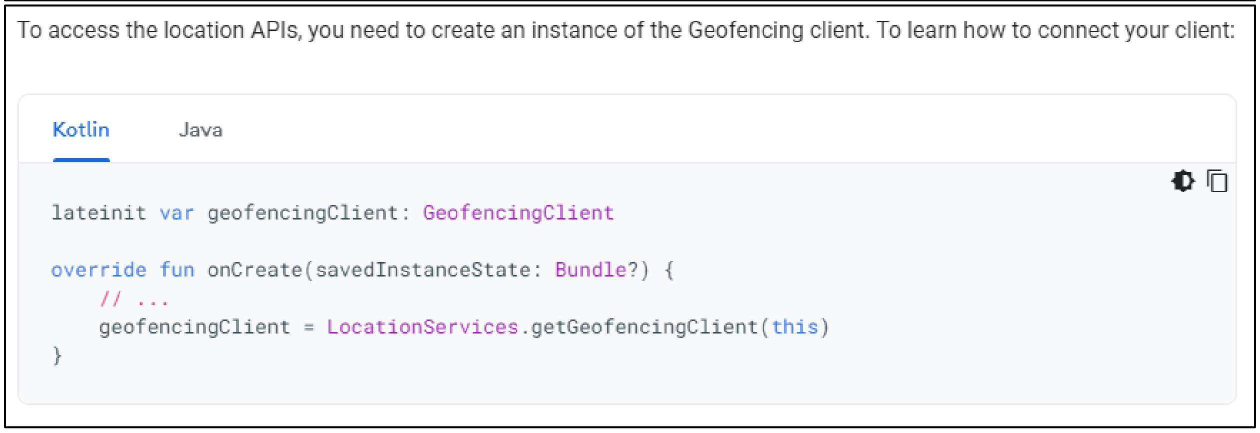
If you want to use a `BroadcastReceiver` to listen for geofence transitions, add an element specifying the service name. This element must be a child of the `<application>` element:

```
<application
   android:allowBackup="true">
   ...
   <receiver android:name=".GeofenceBroadcastReceiver"/>
<application/>
```



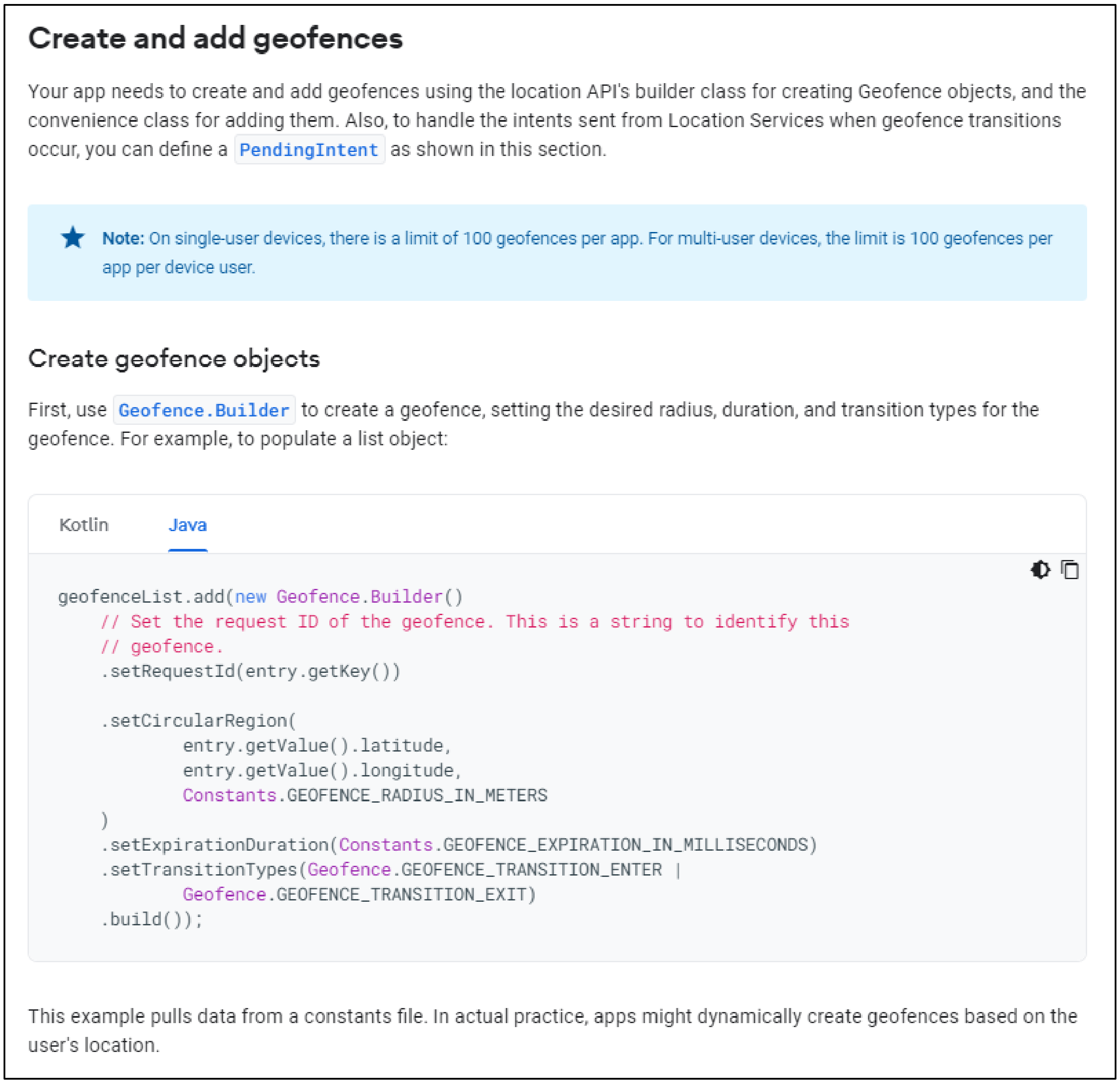
If you want to use a `BroadcastReceiver` to listen for geofence transitions, add an element specifying the service name. This element must be a child of the `<application>` element:

```
<application
   android:allowBackup="true">
   ...
   <receiver android:name=".GeofenceBroadcastReceiver"/>
<application/>
```



To access the location APIs, you need to create an instance of the Geofencing client. To learn how to connect your client:

```
lateinit var geofencingClient: GeofencingClient

override fun onCreate(savedInstanceState: Bundle?) {
    // ...
    geofencingClient = LocationServices.getGeofencingClient(this)
}
```



Create and add geofences

Your app needs to create and add geofences using the location API's builder class for creating Geofence objects, and the convenience class for adding them. Also, to handle the intents sent from Location Services when geofence transitions occur, you can define a `PendingIntent` as shown in this section.

Note: On single-user devices, there is a limit of 100 geofences per app. For multi-user devices, the limit is 100 geofences per app per device user.

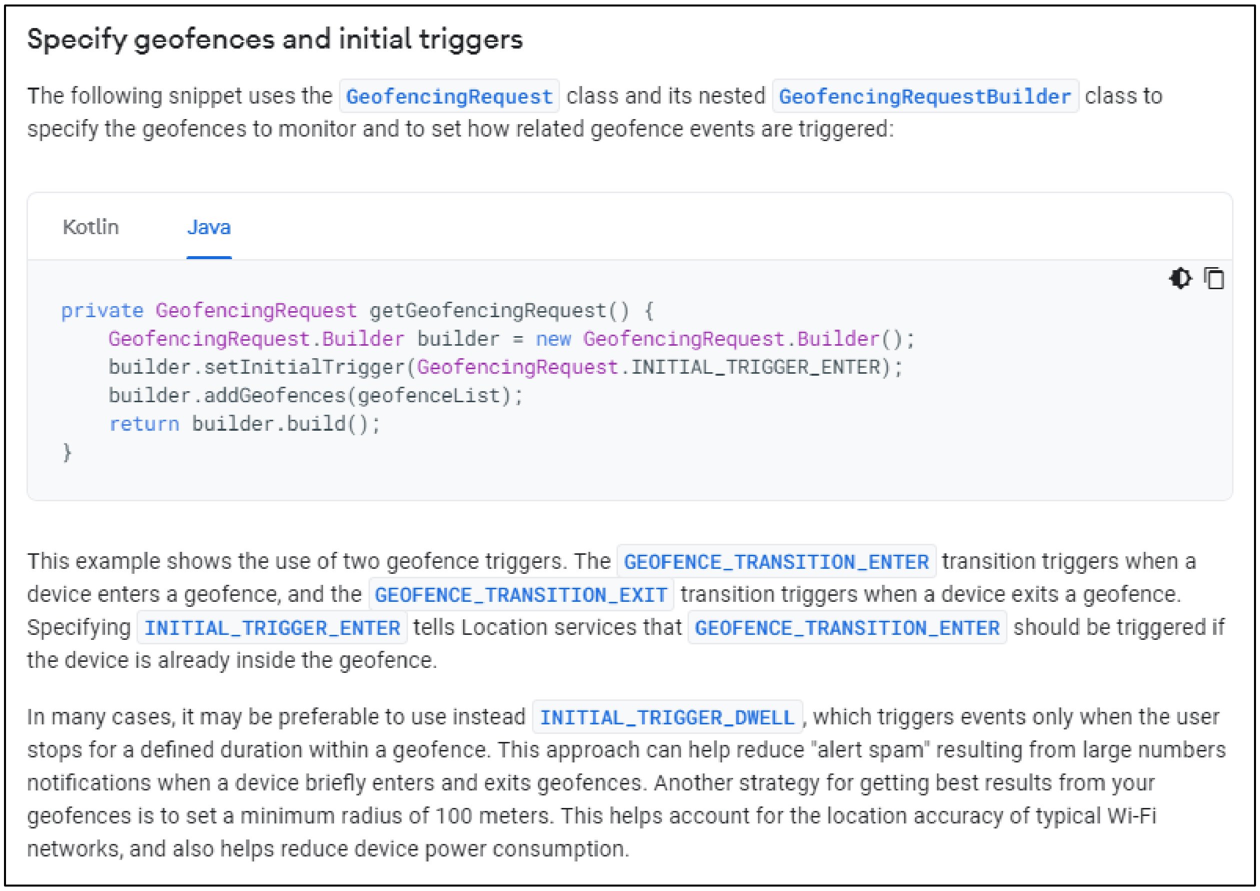
Create geofence objects

First, use `Geofence.Builder` to create a geofence, setting the desired radius, duration, and transition types for the geofence. For example, to populate a list object:

Kotlin Java

```
geofenceList.add(new Geofence.Builder()
    // Set the request ID of the geofence. This is a string to identify this
    // geofence.
    .setRequestId(entry.getKey())

    .setCircularRegion(
            entry.getValue().latitude,
            entry.getValue().longitude,
            Constants.GEOFENCE_RADIUS_IN_METERS
    )
    .setExpirationDuration(Constants.GEOFENCE_EXPIRATION_IN_MILLISECONDS)
    .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER |
            Geofence.GEOFENCE_TRANSITION_EXIT)
    .build());
```

This example pulls data from a constants file. In actual practice, apps might dynamically create geofences based on the user's location.



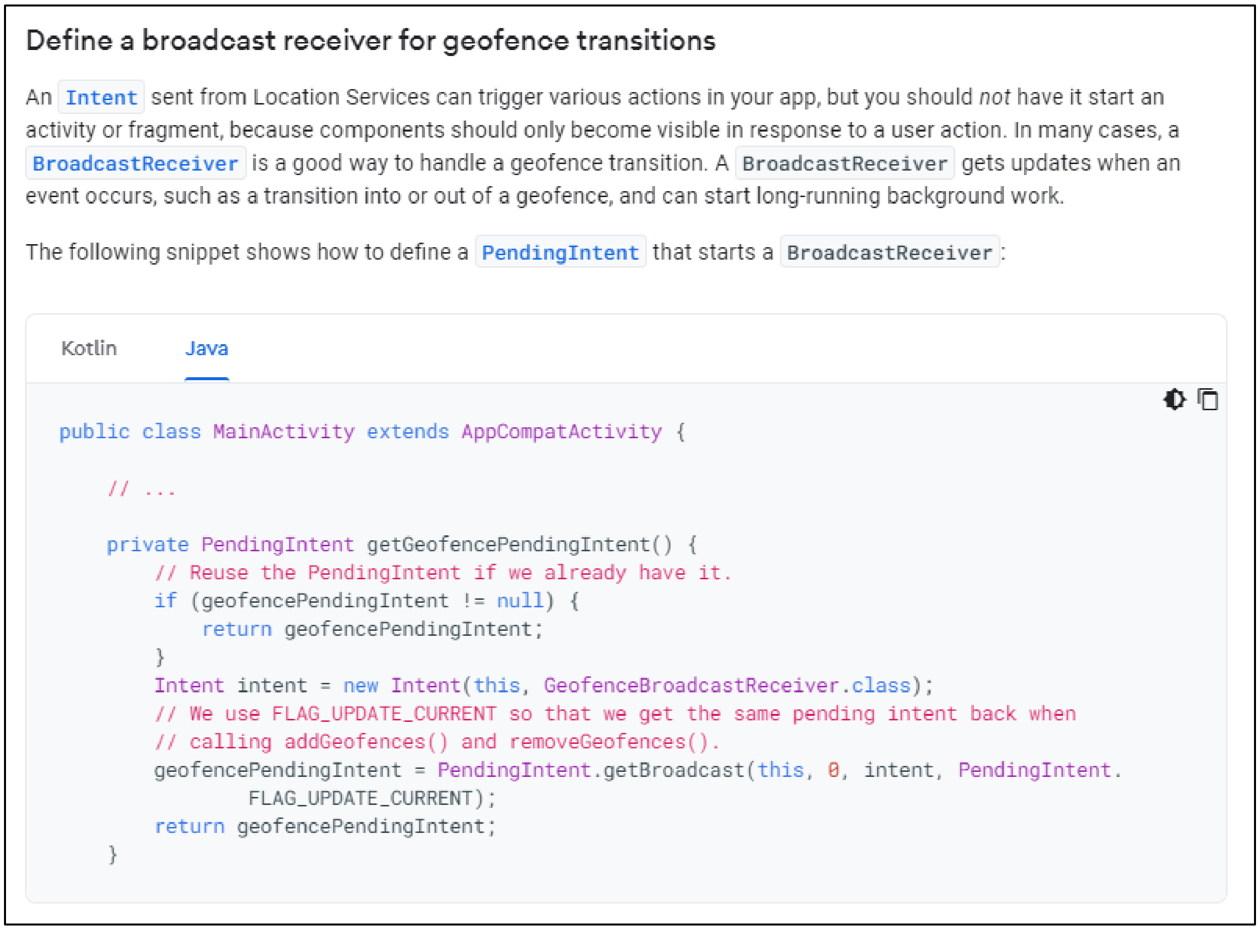
Specify geofences and initial triggers

The following snippet uses the `GeofencingRequest` class and its nested `GeofencingRequestBuilder` class to specify the geofences to monitor and to set how related geofence events are triggered:

```
private GeofencingRequest getGeofencingRequest() {
    GeofencingRequest.Builder builder = new GeofencingRequest.Builder();
    builder.setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER);
    builder.addGeofences(geofenceList);
    return builder.build();
}
```

This example shows the use of two geofence triggers. The `GEOFENCE_TRANSITION_ENTER` transition triggers when a device enters a geofence, and the `GEOFENCE_TRANSITION_EXIT` transition triggers when a device exits a geofence. Specifying `INITIAL_TRIGGER_ENTER` tells Location services that `GEOFENCE_TRANSITION_ENTER` should be triggered if the device is already inside the geofence.

In many cases, it may be preferable to use instead `INITIAL_TRIGGER_DWELL`, which triggers events only when the user stops for a defined duration within a geofence. This approach can help reduce "alert spam" resulting from large numbers notifications when a device briefly enters and exits geofences. Another strategy for getting best results from your geofences is to set a minimum radius of 100 meters. This helps account for the location accuracy of typical Wi-Fi networks, and also helps reduce device power consumption.

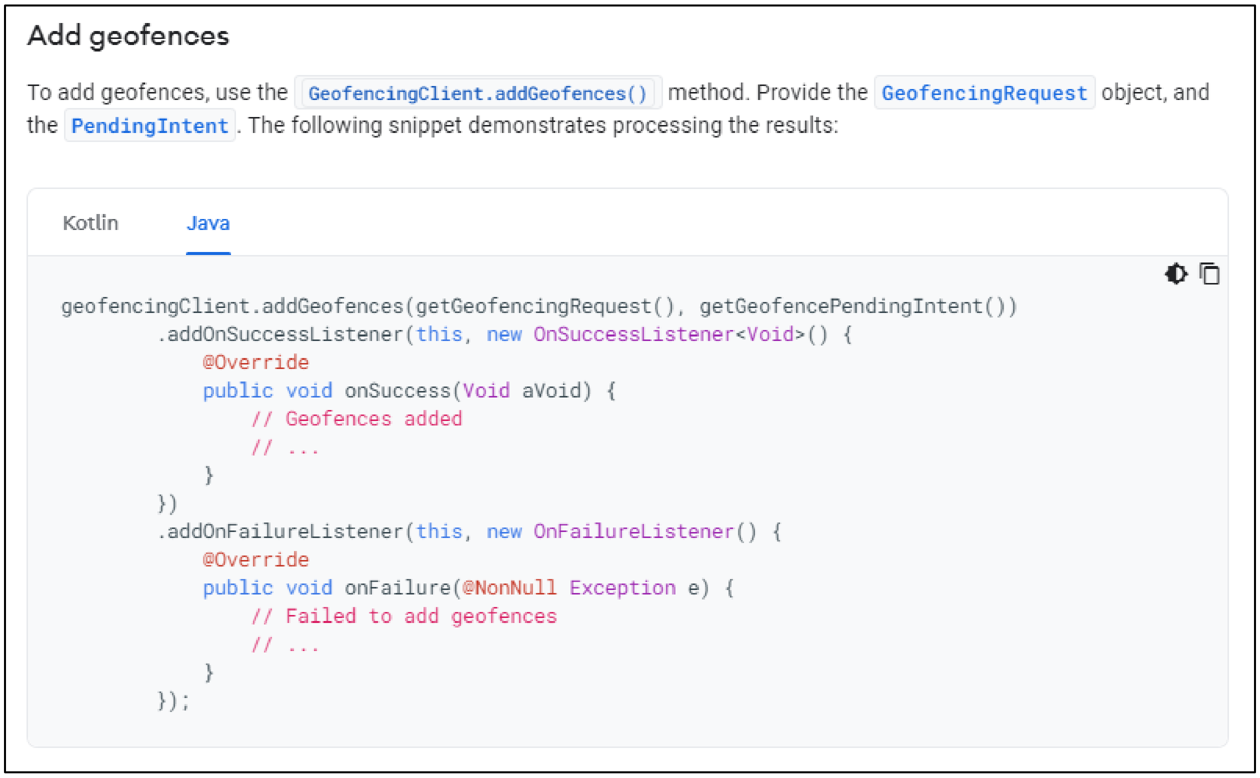


Define a broadcast receiver for geofence transitions

An `Intent` sent from Location Services can trigger various actions in your app, but you should *not* have it start an activity or fragment, because components should only become visible in response to a user action. In many cases, a `BroadcastReceiver` is a good way to handle a geofence transition. A `BroadcastReceiver` gets updates when an event occurs, such as a transition into or out of a geofence, and can start long-running background work.

The following snippet shows how to define a `PendingIntent` that starts a `BroadcastReceiver`:

```java
public class MainActivity extends AppCompatActivity {

    // ...

    private PendingIntent getGeofencePendingIntent() {
        // Reuse the PendingIntent if we already have it.
        if (geofencePendingIntent != null) {
            return geofencePendingIntent;
        }
        Intent intent = new Intent(this, GeofenceBroadcastReceiver.class);
        // We use FLAG_UPDATE_CURRENT so that we get the same pending intent back when
        // calling addGeofences() and removeGeofences().
        geofencePendingIntent = PendingIntent.getBroadcast(this, 0, intent, PendingIntent.
                FLAG_UPDATE_CURRENT);
        return geofencePendingIntent;
    }
```

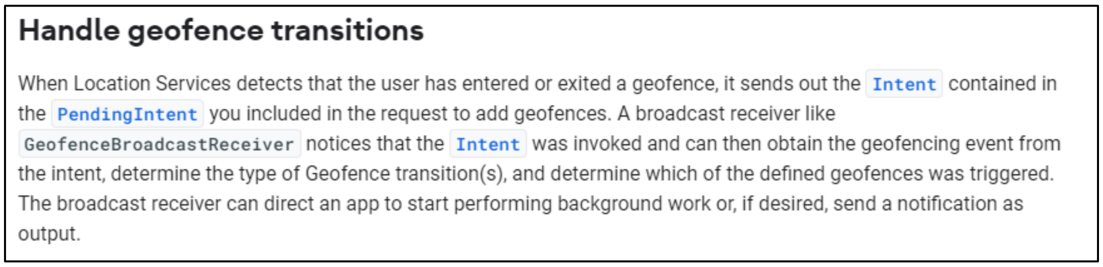


Add geofences

To add geofences, use the `GeofencingClient.addGeofences()` method. Provide the `GeofencingRequest` object, and the `PendingIntent`. The following snippet demonstrates processing the results:

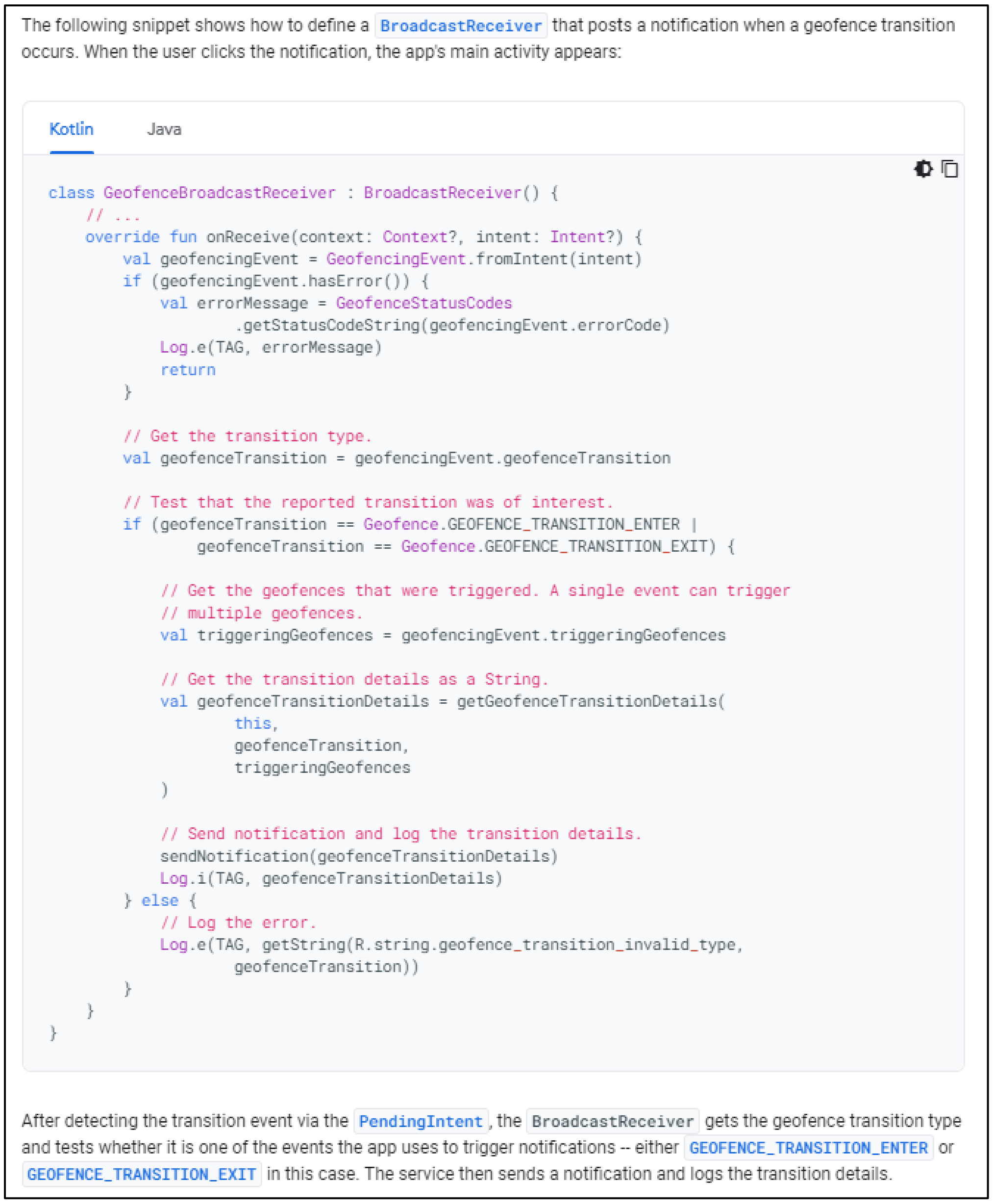
Kotlin Java

```
geofencingClient.addGeofences(getGeofencingRequest(), getGeofencePendingIntent())
        .addOnSuccessListener(this, new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                // Geofences added
                // ...
            }
        })
        .addOnFailureListener(this, new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                // Failed to add geofences
                // ...
            }
        });
```



Handle geofence transitions

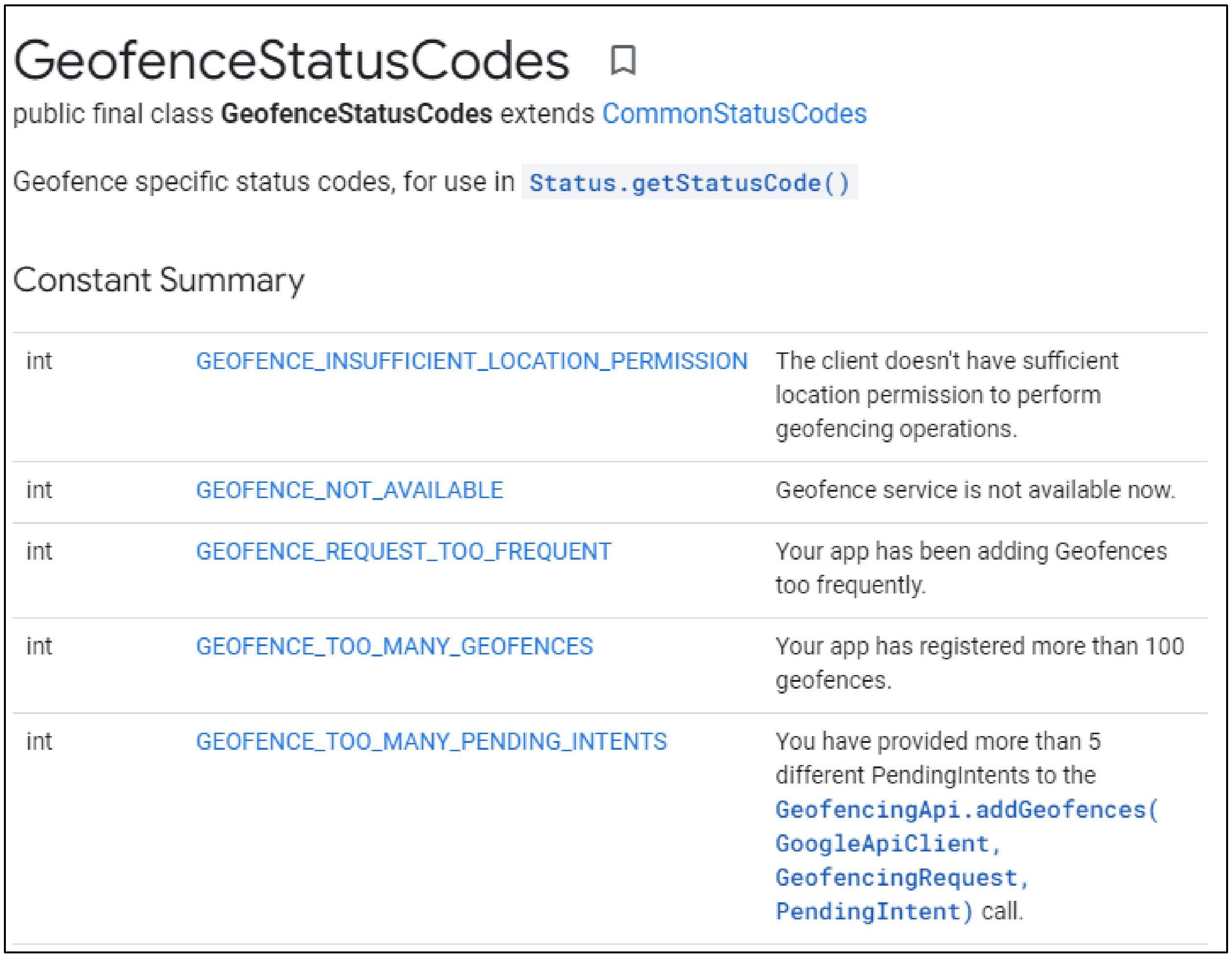
When Location Services detects that the user has entered or exited a geofence, it sends out the `Intent` contained in the `PendingIntent` you included in the request to add geofences. A broadcast receiver like `GeofenceBroadcastReceiver` notices that the `Intent` was invoked and can then obtain the geofencing event from the intent, determine the type of Geofence transition(s), and determine which of the defined geofences was triggered. The broadcast receiver can direct an app to start performing background work or, if desired, send a notification as output.


The following snippet shows how to define a `BroadcastReceiver` that posts a notification when a geofence transition occurs. When the user clicks the notification, the app's main activity appears:

```
class GeofenceBroadcastReceiver : BroadcastReceiver() {
    // ...
    override fun onReceive(context: Context?, intent: Intent?) {
        val geofencingEvent = GeofencingEvent.fromIntent(intent)
        if (geofencingEvent.hasError()) {
            val errorMessage = GeofenceStatusCodes
                    .getStatusCodeString(geofencingEvent.errorCode)
            Log.e(TAG, errorMessage)
            return
        }

        // Get the transition type.
        val geofenceTransition = geofencingEvent.geofenceTransition

        // Test that the reported transition was of interest.
        if (geofenceTransition == Geofence.GEOFENCE_TRANSITION_ENTER |
                geofenceTransition == Geofence.GEOFENCE_TRANSITION_EXIT) {

            // Get the geofences that were triggered. A single event can trigger
            // multiple geofences.
            val triggeringGeofences = geofencingEvent.triggeringGeofences

            // Get the transition details as a String.
            val geofenceTransitionDetails = getGeofenceTransitionDetails(
                    this,
                    geofenceTransition,
                    triggeringGeofences
            )

            // Send notification and log the transition details.
            sendNotification(geofenceTransitionDetails)
            Log.i(TAG, geofenceTransitionDetails)
        } else {
            // Log the error.
            Log.e(TAG, getString(R.string.geofence_transition_invalid_type,
                    geofenceTransition))
        }
    }
}
```

After detecting the transition event via the `PendingIntent`, the `BroadcastReceiver` gets the geofence transition type and tests whether it is one of the events the app uses to trigger notifications -- either `GEOFENCE_TRANSITION_ENTER` or `GEOFENCE_TRANSITION_EXIT` in this case. The service then sends a notification and logs the transition details.

https://developer.android.com/training/location/geofencing



GeofenceStatusCodes

public final class **GeofenceStatusCodes** extends CommonStatusCodes

Geofence specific status codes, for use in `Status.getStatusCode()`

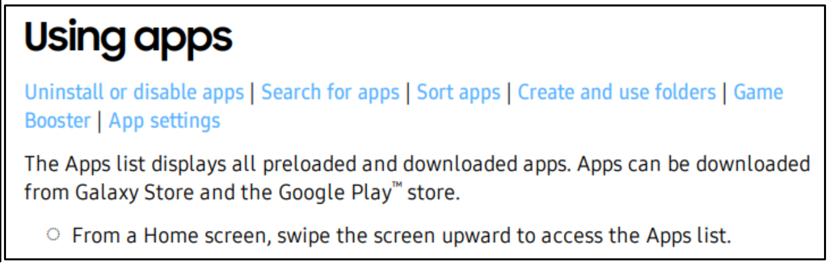
Constant Summary

int	GEOFENCE_INSUFFICIENT_LOCATION_PERMISSION	The client doesn't have sufficient location permission to perform geofencing operations.
int	GEOFENCE_NOT_AVAILABLE	Geofence service is not available now.
int	GEOFENCE_REQUEST_TOO_FREQUENT	Your app has been adding Geofences too frequently.
int	GEOFENCE_TOO_MANY_GEOFENCES	Your app has registered more than 100 geofences.
int	GEOFENCE_TOO_MANY_PENDING_INTENTS	You have provided more than 5 different PendingIntents to the `GeofencingApi.addGeofences( GoogleApiClient, GeofencingRequest, PendingIntent)` call.

https://developers.google.com/android/reference/com/google/android/gms/location/GeofenceStatusCodes

36. Samsung provides a system comprising a processor configured to execute a program of instructions, the program of instructions comprising: at least one instruction to store a record of the interest in the selected one of the plurality of geographic areas; at least one instruction to receive, from the sponsor, content to be delivered to registered application programs having target locations contained within the selected one of the plurality of geographic areas; at least one instruction to receive a request from a registered application program for content to be used within the registered application program; at least one instruction to select content associated with at least

one of the plurality of sponsors, to be delivered to the registered application program of a specific type, in accordance with a content delivery reservation associating the at least one sponsor with a geographic area and a registered application program type; and at least one instruction to provide the selected content to the registered application program, including as demonstrated in the exemplary text and images below:



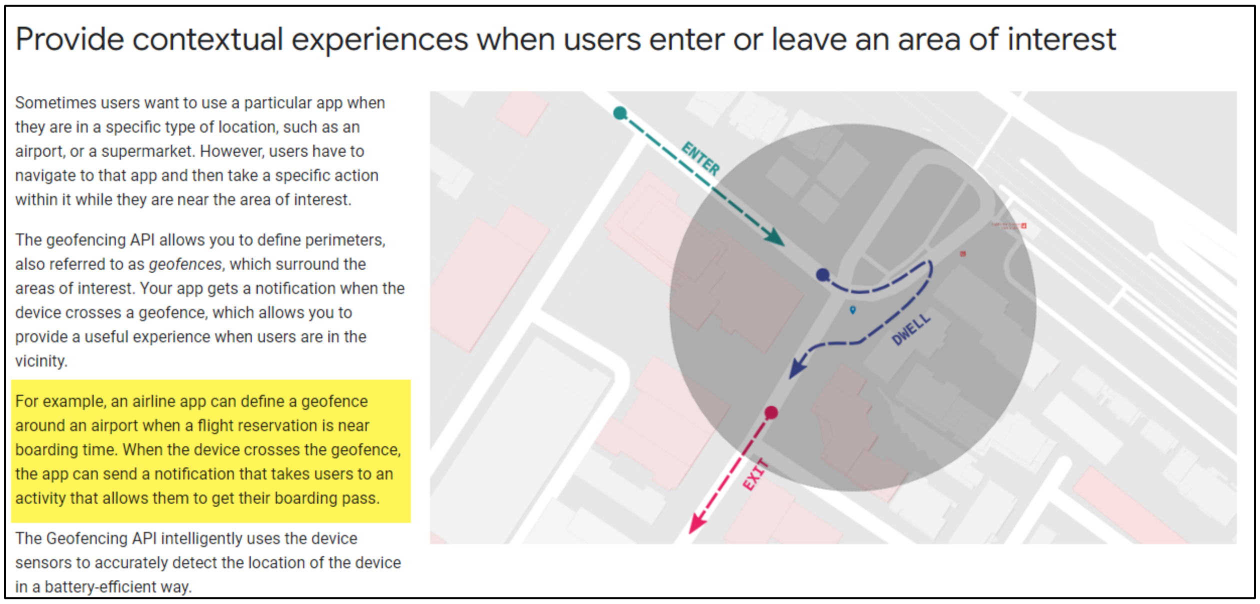
Using apps

Uninstall or disable apps | Search for apps | Sort apps | Create and use folders | Game Booster | App settings

The Apps list displays all preloaded and downloaded apps. Apps can be downloaded from Galaxy Store and the Google Play™ store.

- From a Home screen, swipe the screen upward to access the Apps list.

https://downloadcenter.samsung.com/content/UM/202103/20210304004310357/ATT_SM-G970U_G973U_G975U_EN_UM_R_11.0_022221_FINAL.pdf



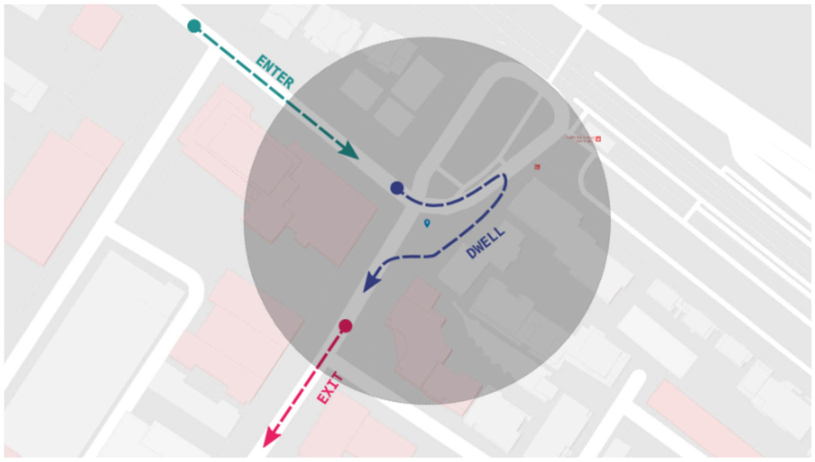
Provide contextual experiences when users enter or leave an area of interest

Sometimes users want to use a particular app when they are in a specific type of location, such as an airport, or a supermarket. However, users have to navigate to that app and then take a specific action within it while they are near the area of interest.

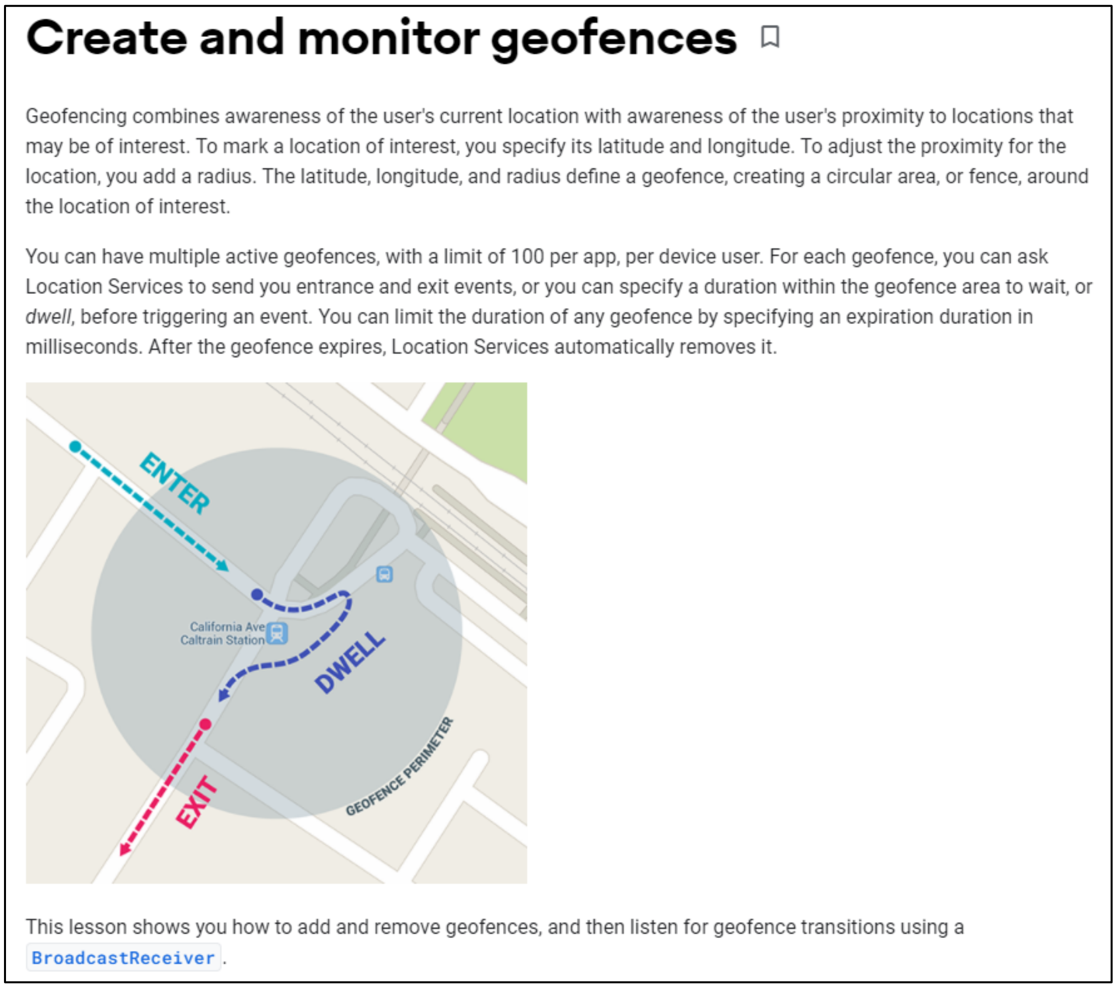
The geofencing API allows you to define perimeters, also referred to as *geofences*, which surround the areas of interest. Your app gets a notification when the device crosses a geofence, which allows you to provide a useful experience when users are in the vicinity.

For example, an airline app can define a geofence around an airport when a flight reservation is near boarding time. When the device crosses the geofence, the app can send a notification that takes users to an activity that allows them to get their boarding pass.

The Geofencing API intelligently uses the device sensors to accurately detect the location of the device in a battery-efficient way.



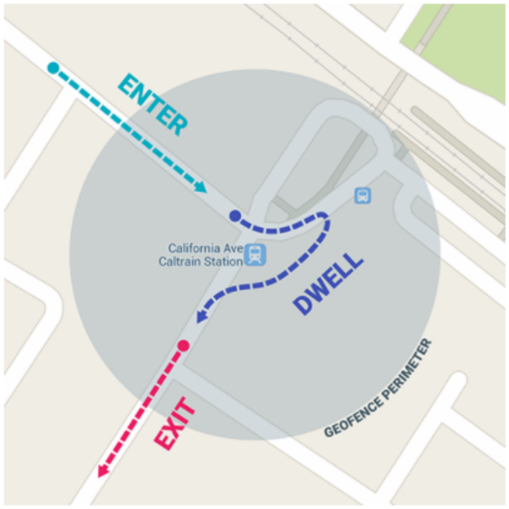
https://developers.google.com/location-context/geofencing#:~:text=The%20geofencing%20API%20allows%20you,users%20are%20in%20the%20vicinity



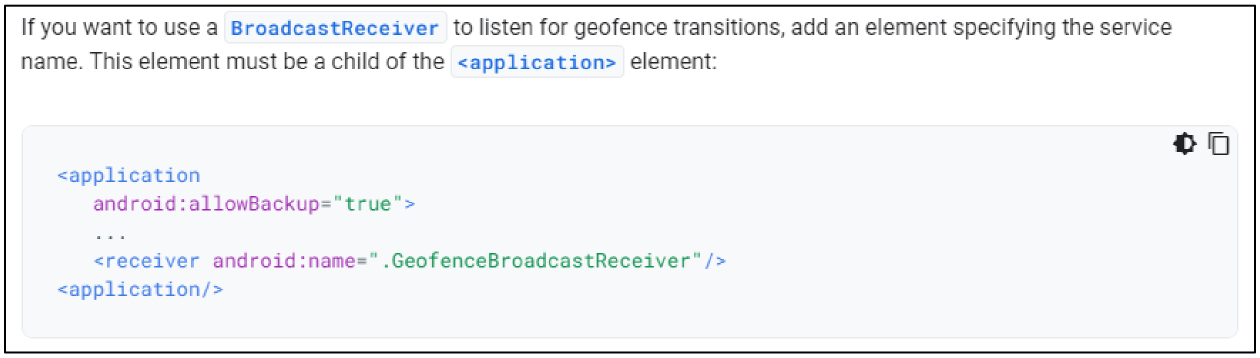
Create and monitor geofences

Geofencing combines awareness of the user's current location with awareness of the user's proximity to locations that may be of interest. To mark a location of interest, you specify its latitude and longitude. To adjust the proximity for the location, you add a radius. The latitude, longitude, and radius define a geofence, creating a circular area, or fence, around the location of interest.

You can have multiple active geofences, with a limit of 100 per app, per device user. For each geofence, you can ask Location Services to send you entrance and exit events, or you can specify a duration within the geofence area to wait, or *dwell*, before triggering an event. You can limit the duration of any geofence by specifying an expiration duration in milliseconds. After the geofence expires, Location Services automatically removes it.

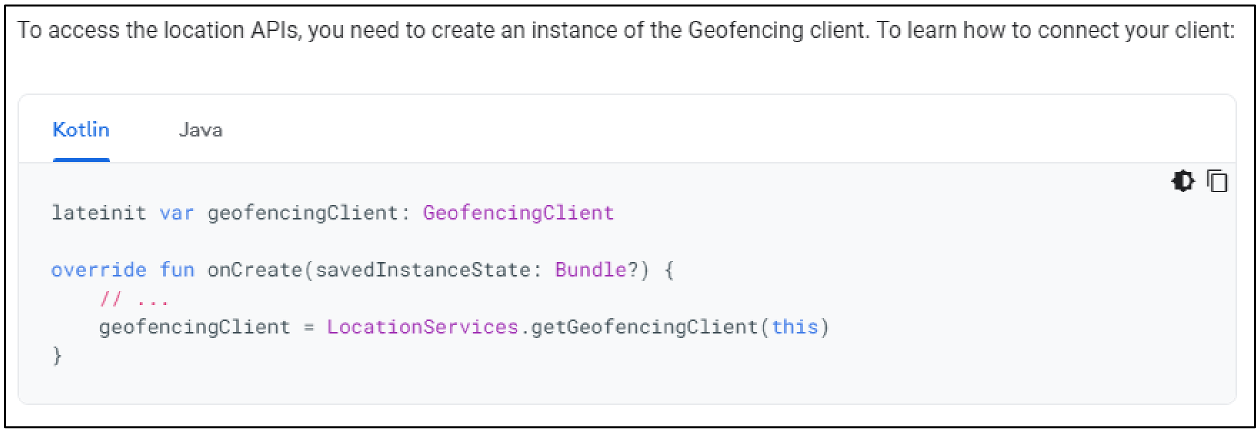


This lesson shows you how to add and remove geofences, and then listen for geofence transitions using a `BroadcastReceiver`.



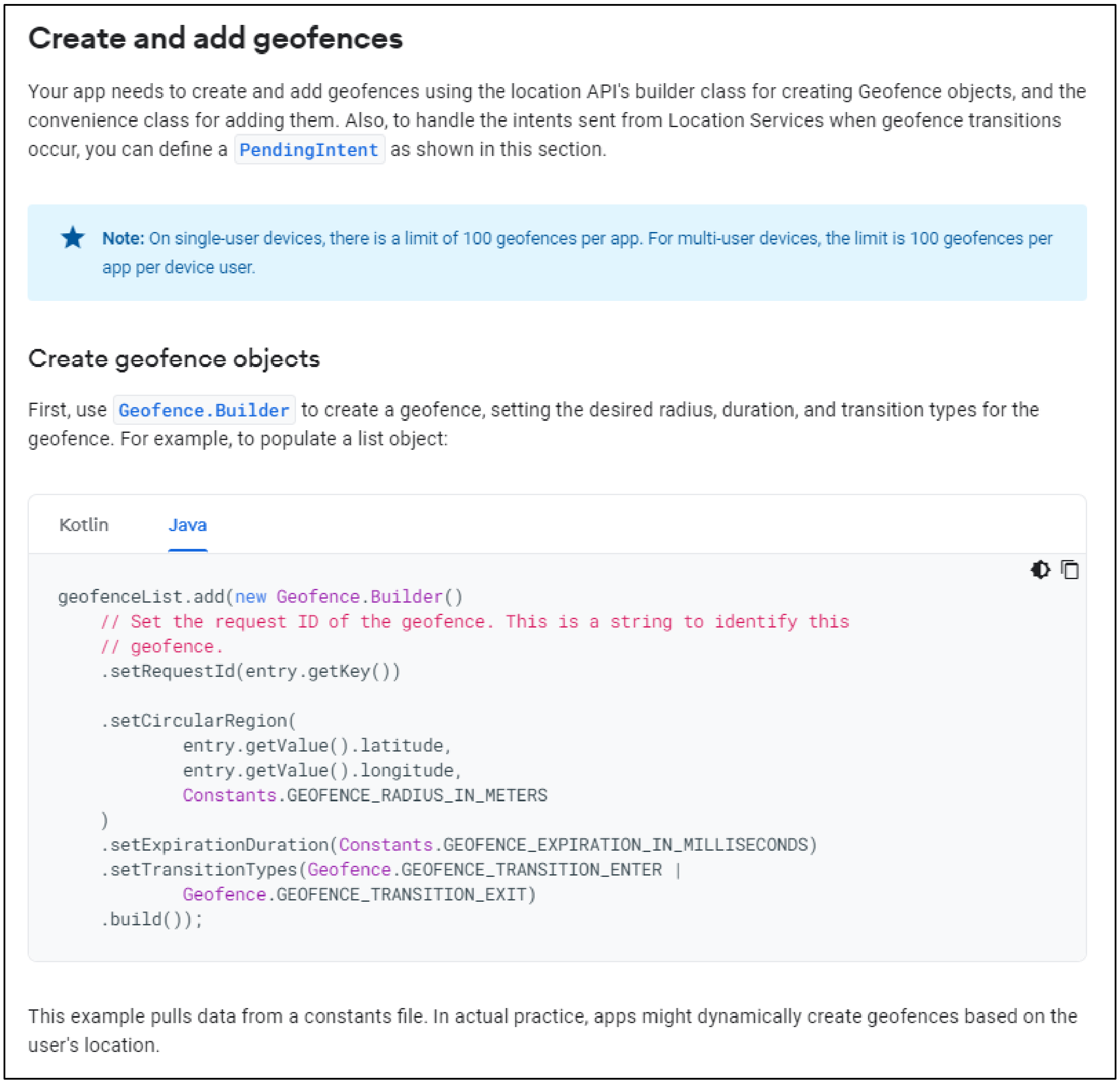
If you want to use a `BroadcastReceiver` to listen for geofence transitions, add an element specifying the service name. This element must be a child of the `<application>` element:

```
<application
   android:allowBackup="true">
   ...
   <receiver android:name=".GeofenceBroadcastReceiver"/>
<application/>
```



To access the location APIs, you need to create an instance of the Geofencing client. To learn how to connect your client:

```kotlin
lateinit var geofencingClient: GeofencingClient

override fun onCreate(savedInstanceState: Bundle?) {
    // ...
    geofencingClient = LocationServices.getGeofencingClient(this)
}
```



Create and add geofences

Your app needs to create and add geofences using the location API's builder class for creating Geofence objects, and the convenience class for adding them. Also, to handle the intents sent from Location Services when geofence transitions occur, you can define a `PendingIntent` as shown in this section.

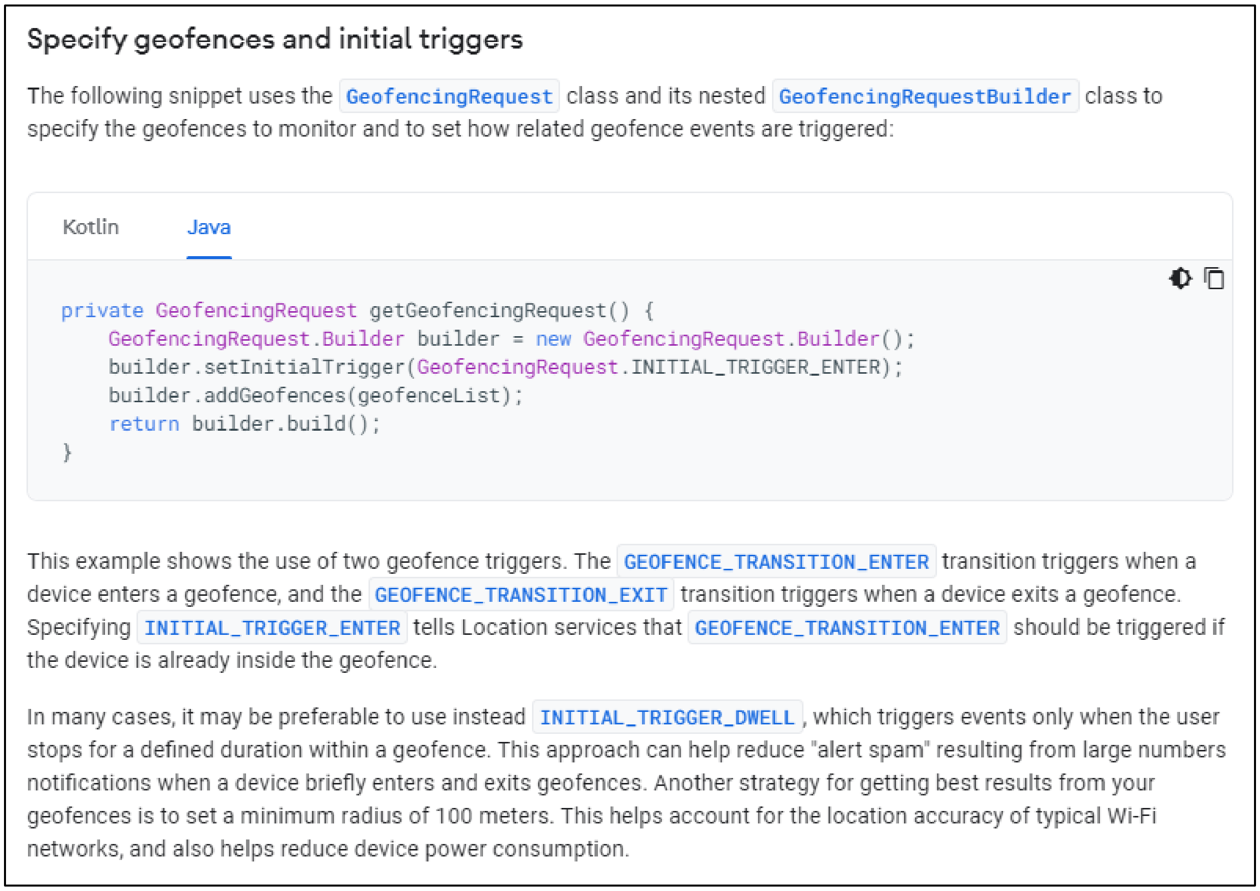
★ **Note:** On single-user devices, there is a limit of 100 geofences per app. For multi-user devices, the limit is 100 geofences per app per device user.

Create geofence objects

First, use `Geofence.Builder` to create a geofence, setting the desired radius, duration, and transition types for the geofence. For example, to populate a list object:

```
geofenceList.add(new Geofence.Builder()
    // Set the request ID of the geofence. This is a string to identify this
    // geofence.
    .setRequestId(entry.getKey())

    .setCircularRegion(
            entry.getValue().latitude,
            entry.getValue().longitude,
            Constants.GEOFENCE_RADIUS_IN_METERS
    )
    .setExpirationDuration(Constants.GEOFENCE_EXPIRATION_IN_MILLISECONDS)
    .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER |
            Geofence.GEOFENCE_TRANSITION_EXIT)
    .build());
```

This example pulls data from a constants file. In actual practice, apps might dynamically create geofences based on the user's location.



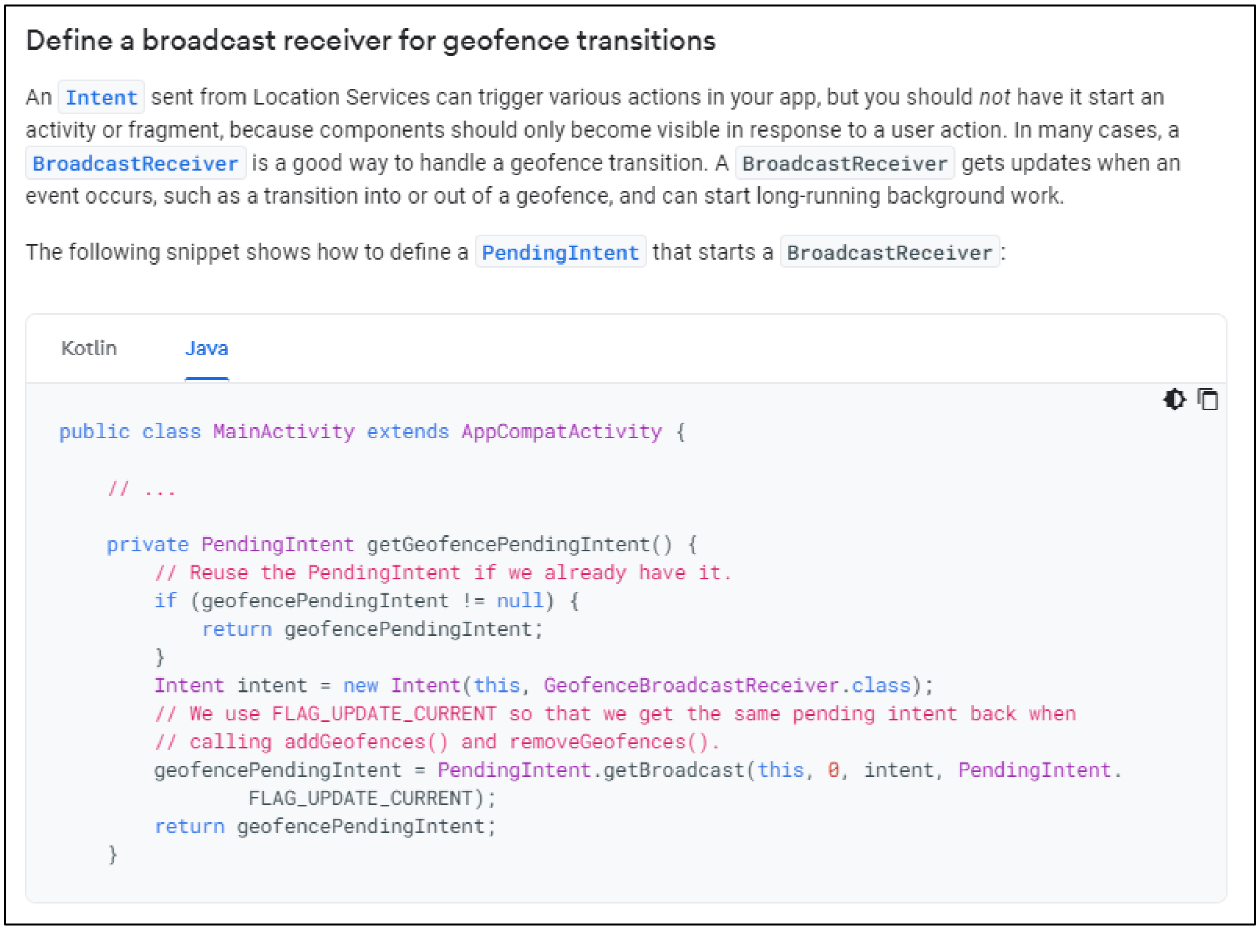
Specify geofences and initial triggers

The following snippet uses the `GeofencingRequest` class and its nested `GeofencingRequestBuilder` class to specify the geofences to monitor and to set how related geofence events are triggered:

```
private GeofencingRequest getGeofencingRequest() {
    GeofencingRequest.Builder builder = new GeofencingRequest.Builder();
    builder.setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER);
    builder.addGeofences(geofenceList);
    return builder.build();
}
```

This example shows the use of two geofence triggers. The `GEOFENCE_TRANSITION_ENTER` transition triggers when a device enters a geofence, and the `GEOFENCE_TRANSITION_EXIT` transition triggers when a device exits a geofence. Specifying `INITIAL_TRIGGER_ENTER` tells Location services that `GEOFENCE_TRANSITION_ENTER` should be triggered if the device is already inside the geofence.

In many cases, it may be preferable to use instead `INITIAL_TRIGGER_DWELL`, which triggers events only when the user stops for a defined duration within a geofence. This approach can help reduce "alert spam" resulting from large numbers notifications when a device briefly enters and exits geofences. Another strategy for getting best results from your geofences is to set a minimum radius of 100 meters. This helps account for the location accuracy of typical Wi-Fi networks, and also helps reduce device power consumption.



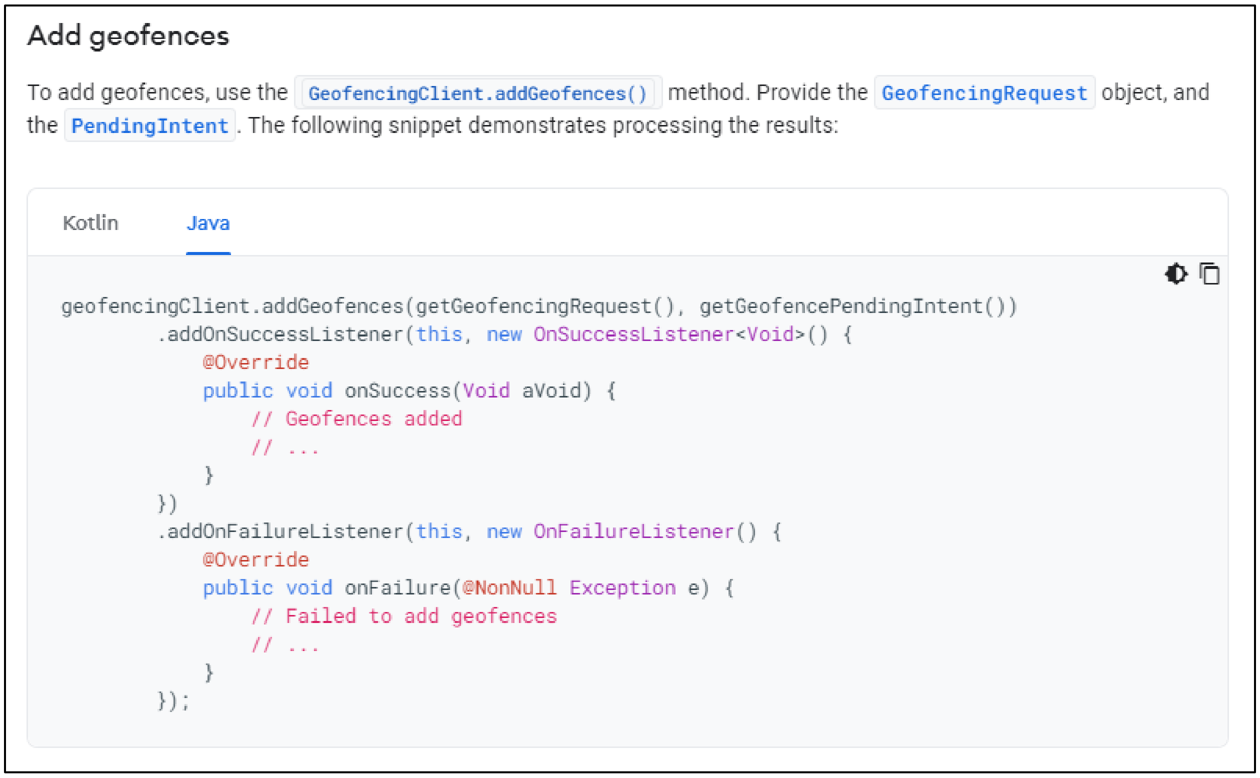
Define a broadcast receiver for geofence transitions

An `Intent` sent from Location Services can trigger various actions in your app, but you should *not* have it start an activity or fragment, because components should only become visible in response to a user action. In many cases, a `BroadcastReceiver` is a good way to handle a geofence transition. A `BroadcastReceiver` gets updates when an event occurs, such as a transition into or out of a geofence, and can start long-running background work.

The following snippet shows how to define a `PendingIntent` that starts a `BroadcastReceiver`:

Kotlin Java

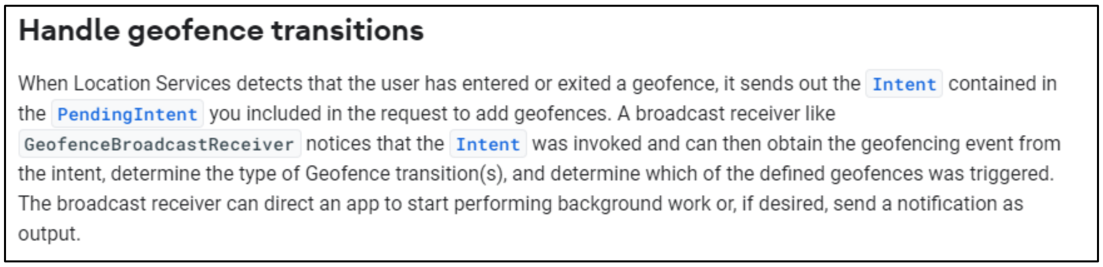
```
public class MainActivity extends AppCompatActivity {

    // ...

    private PendingIntent getGeofencePendingIntent() {
        // Reuse the PendingIntent if we already have it.
        if (geofencePendingIntent != null) {
            return geofencePendingIntent;
        }
        Intent intent = new Intent(this, GeofenceBroadcastReceiver.class);
        // We use FLAG_UPDATE_CURRENT so that we get the same pending intent back when
        // calling addGeofences() and removeGeofences().
        geofencePendingIntent = PendingIntent.getBroadcast(this, 0, intent, PendingIntent.
                FLAG_UPDATE_CURRENT);
        return geofencePendingIntent;
    }
```



Add geofences

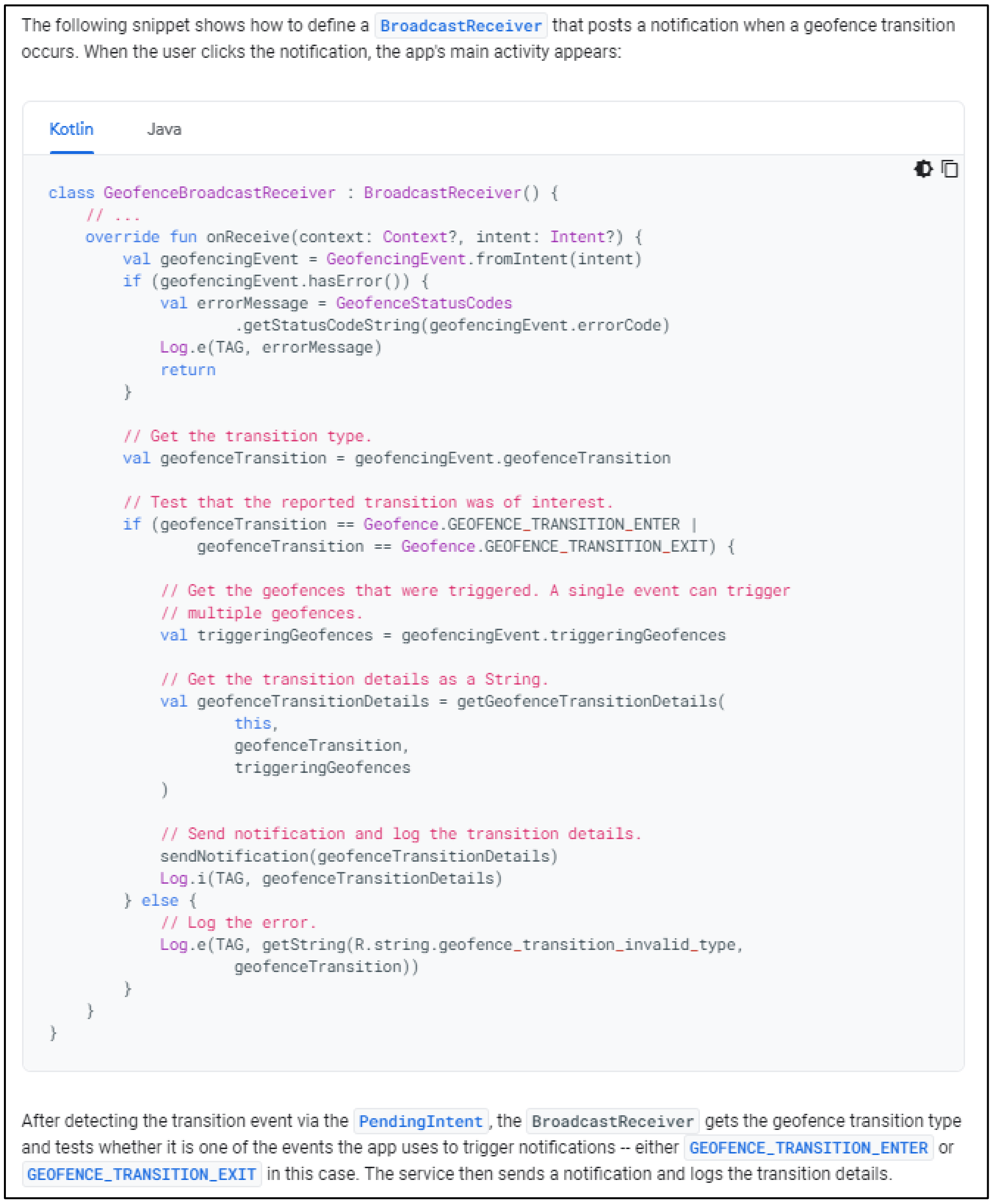
To add geofences, use the `GeofencingClient.addGeofences()` method. Provide the `GeofencingRequest` object, and the `PendingIntent`. The following snippet demonstrates processing the results:

```
geofencingClient.addGeofences(getGeofencingRequest(), getGeofencePendingIntent())
        .addOnSuccessListener(this, new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                // Geofences added
                // ...
            }
        })
        .addOnFailureListener(this, new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                // Failed to add geofences
                // ...
            }
        });
```



Handle geofence transitions

When Location Services detects that the user has entered or exited a geofence, it sends out the `Intent` contained in the `PendingIntent` you included in the request to add geofences. A broadcast receiver like `GeofenceBroadcastReceiver` notices that the `Intent` was invoked and can then obtain the geofencing event from the intent, determine the type of Geofence transition(s), and determine which of the defined geofences was triggered. The broadcast receiver can direct an app to start performing background work or, if desired, send a notification as output.



The following snippet shows how to define a `BroadcastReceiver` that posts a notification when a geofence transition occurs. When the user clicks the notification, the app's main activity appears:

Kotlin Java

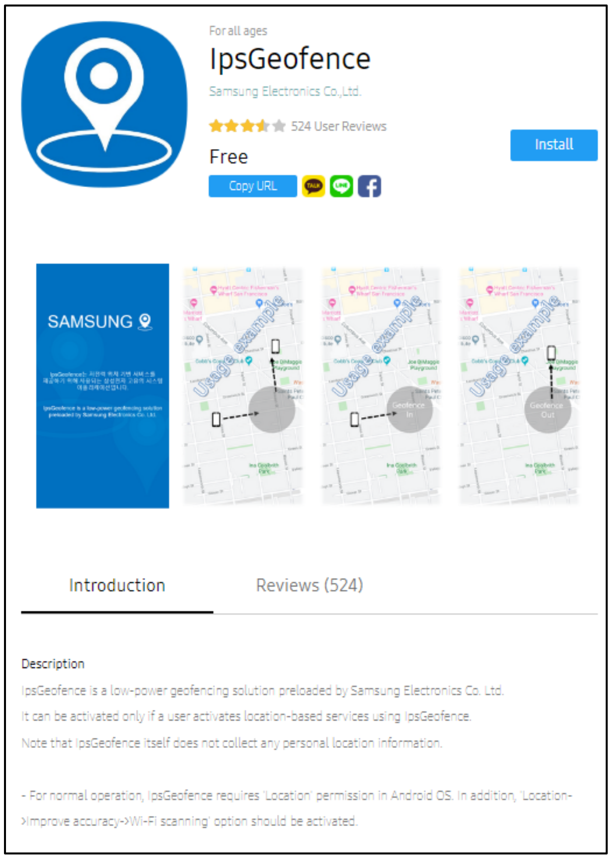
```kotlin
class GeofenceBroadcastReceiver : BroadcastReceiver() {
    // ...
    override fun onReceive(context: Context?, intent: Intent?) {
        val geofencingEvent = GeofencingEvent.fromIntent(intent)
        if (geofencingEvent.hasError()) {
            val errorMessage = GeofenceStatusCodes
                    .getStatusCodeString(geofencingEvent.errorCode)
            Log.e(TAG, errorMessage)
            return
        }

        // Get the transition type.
        val geofenceTransition = geofencingEvent.geofenceTransition

        // Test that the reported transition was of interest.
        if (geofenceTransition == Geofence.GEOFENCE_TRANSITION_ENTER |
                geofenceTransition == Geofence.GEOFENCE_TRANSITION_EXIT) {

            // Get the geofences that were triggered. A single event can trigger
            // multiple geofences.
            val triggeringGeofences = geofencingEvent.triggeringGeofences

            // Get the transition details as a String.
            val geofenceTransitionDetails = getGeofenceTransitionDetails(
                    this,
                    geofenceTransition,
                    triggeringGeofences
            )

            // Send notification and log the transition details.
            sendNotification(geofenceTransitionDetails)
            Log.i(TAG, geofenceTransitionDetails)
        } else {
            // Log the error.
            Log.e(TAG, getString(R.string.geofence_transition_invalid_type,
                    geofenceTransition))
        }
    }
}
```

After detecting the transition event via the `PendingIntent`, the `BroadcastReceiver` gets the geofence transition type and tests whether it is one of the events the app uses to trigger notifications -- either `GEOFENCE_TRANSITION_ENTER` or `GEOFENCE_TRANSITION_EXIT` in this case. The service then sends a notification and logs the transition details.

https://developer.android.com/training/location/geofencing

37. On information and belief, Samsung developed applications such as Smart Things, which uses Samsung's geofencing solution called IPSGeofence that appears to function similarly to the geofencing solution in Android described above. In doing so, Samsung infringes the '296 patent in an alternative way by providing a system comprising: a memory; a communications

interface operably coupled to the memory; and at least one processor operably coupled to the memory and the communications interface, the processor configured to execute a program of instructions, the program of instructions comprising: at least one instruction to register a plurality of application programs for use with a content delivery platform, wherein the plurality of registered application programs are each associated with at least one application program type; at least one instruction to establish a plurality of perimeters defining a plurality of geographic areas; at least one instruction to maintain at least one record indicating content delivery reservations associating each of a plurality of sponsors with specific registered application program types and at least one of the plurality of geographic areas; at least one instruction to receive, from a sponsor, a request to obtain an interest in a selected one of the plurality of geographic areas; at least one instruction to provide the sponsor a response to the request; at least one instruction to store a record of the interest in the selected one of the plurality of geographic areas; at least one instruction to receive, from the sponsor, content to be delivered to registered application programs having target locations contained within the selected one of the plurality of geographic areas; at least one instruction to receive a request from a registered application program for content to be used within the registered application program; at least one instruction to select content associated with at least one of the plurality of sponsors, to be delivered to the registered application program of a specific type, in accordance with a content delivery reservation associating the at least one sponsor with a geographic area and a registered application program type; and at least one instruction to provide the selected content to the registered application program. This is demonstrated in the exemplary text and images below:



Description

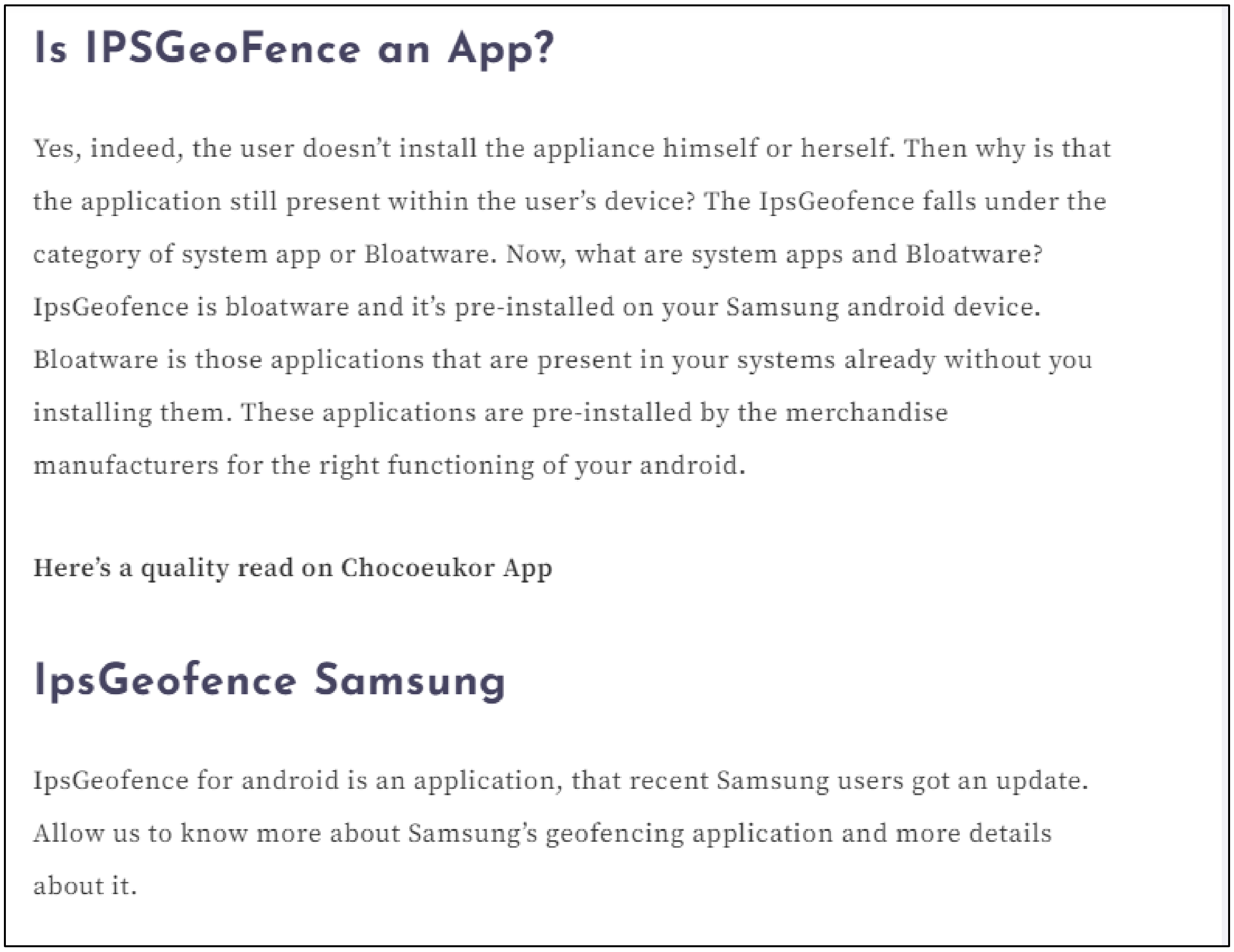
IpsGeofence is a low-power geofencing solution preloaded by Samsung Electronics Co. Ltd.

It can be activated only if a user activates location-based services using IpsGeofence.

Note that IpsGeofence itself does not collect any personal location information.

- For normal operation, IpsGeofence requires 'Location' permission in Android OS. In addition, 'Location->Improve accuracy->Wi-Fi scanning' option should be activated.

https://galaxystore.samsung.com/prepost/000004701101?appId=com.samsung.android.ipsgeofence



Is IPSGeoFence an App?

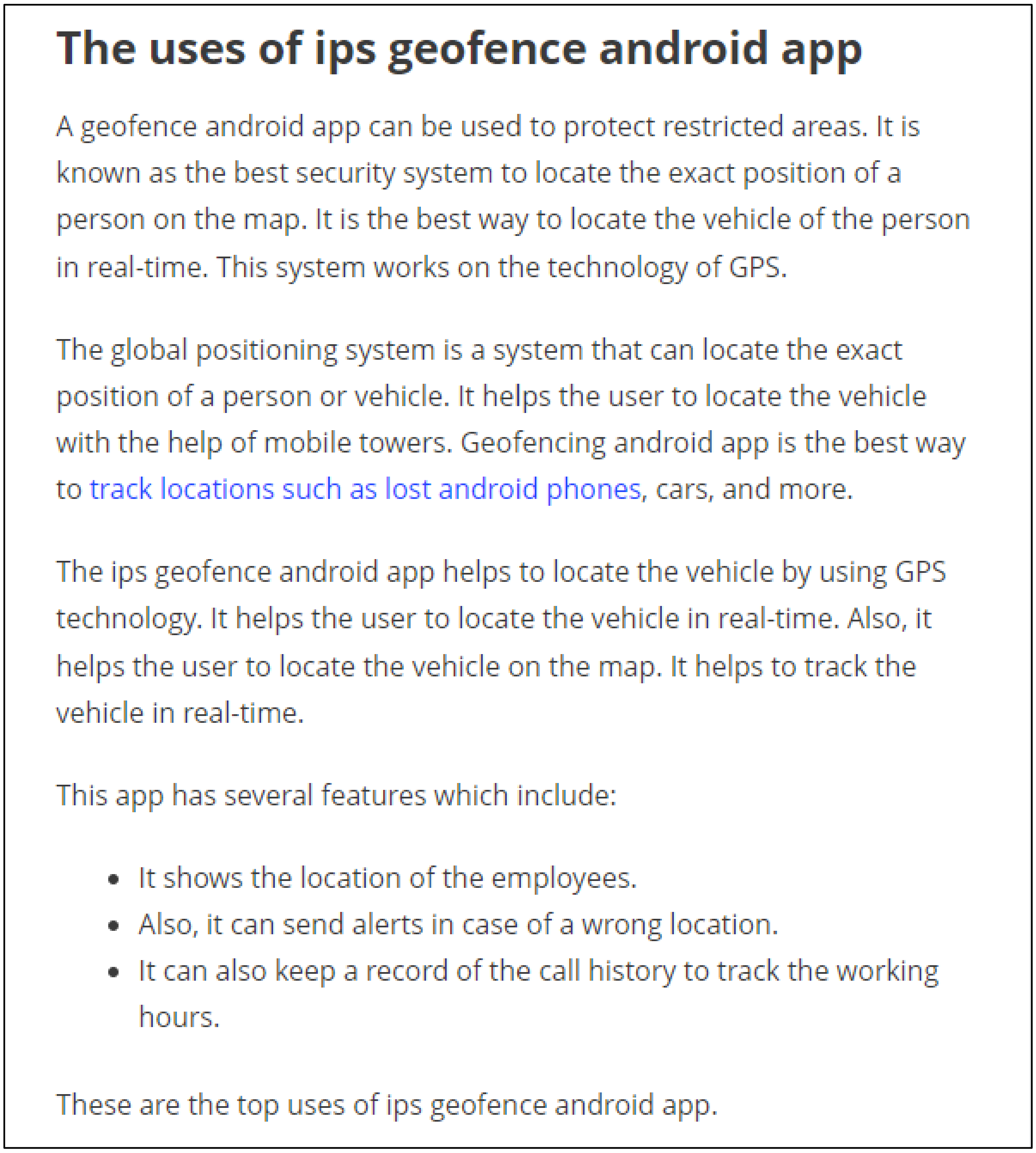
Yes, indeed, the user doesn't install the appliance himself or herself. Then why is that the application still present within the user's device? The IpsGeofence falls under the category of system app or Bloatware. Now, what are system apps and Bloatware? IpsGeofence is bloatware and it's pre-installed on your Samsung android device. Bloatware is those applications that are present in your systems already without you installing them. These applications are pre-installed by the merchandise manufacturers for the right functioning of your android.

Here's a quality read on Chocoeukor App

IpsGeofence Samsung

IpsGeofence for android is an application, that recent Samsung users got an update. Allow us to know more about Samsung's geofencing application and more details about it.

https://hackanons.com/2021/06/what-is-ipsgeofence-app.html



The uses of ips geofence android app

A geofence android app can be used to protect restricted areas. It is known as the best security system to locate the exact position of a person on the map. It is the best way to locate the vehicle of the person in real-time. This system works on the technology of GPS.

The global positioning system is a system that can locate the exact position of a person or vehicle. It helps the user to locate the vehicle with the help of mobile towers. Geofencing android app is the best way to track locations such as lost android phones, cars, and more.

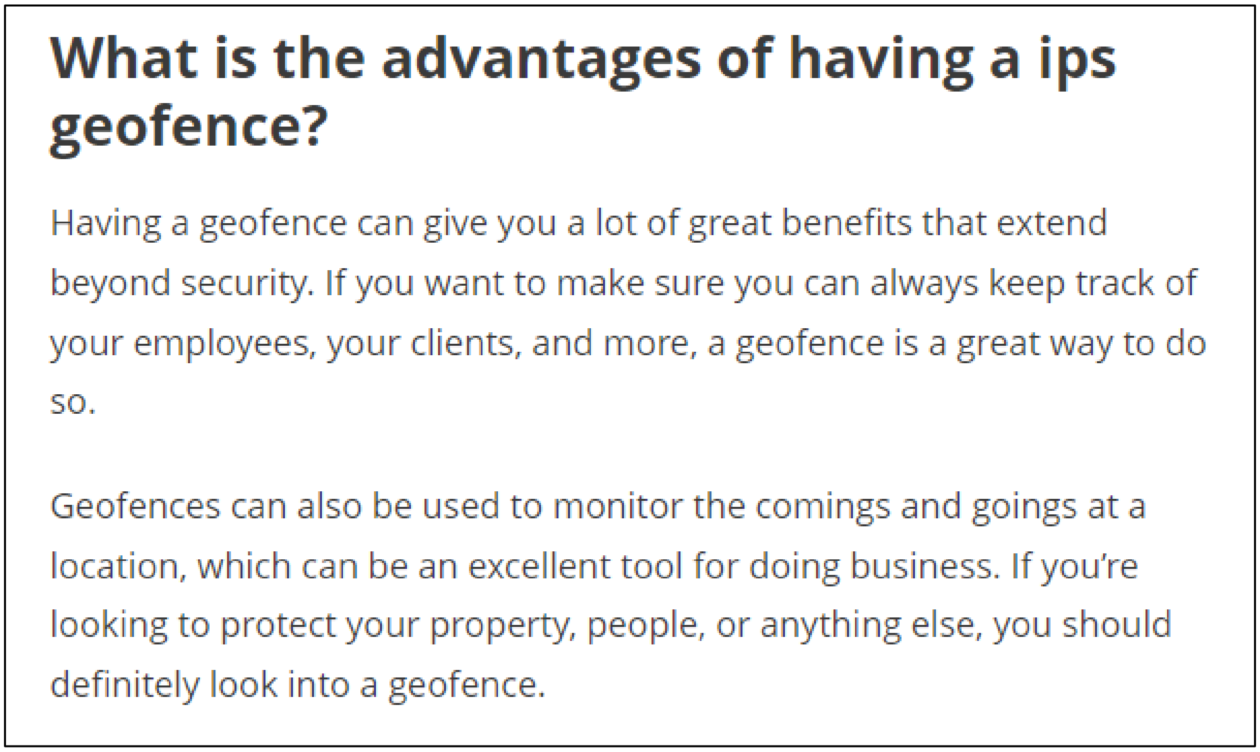
The ips geofence android app helps to locate the vehicle by using GPS technology. It helps the user to locate the vehicle in real-time. Also, it helps the user to locate the vehicle on the map. It helps to track the vehicle in real-time.

This app has several features which include:

- It shows the location of the employees.
- Also, it can send alerts in case of a wrong location.
- It can also keep a record of the call history to track the working hours.

These are the top uses of ips geofence android app.

https://grouphowto.com/ipsgeofence-app/

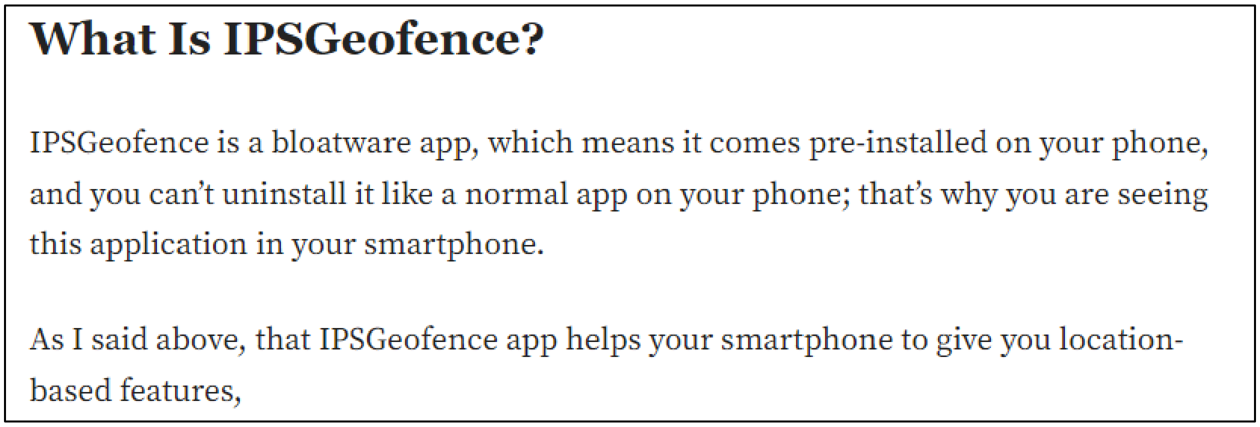


What is the advantages of having a ips geofence?

Having a geofence can give you a lot of great benefits that extend beyond security. If you want to make sure you can always keep track of your employees, your clients, and more, a geofence is a great way to do so.

Geofences can also be used to monitor the comings and goings at a location, which can be an excellent tool for doing business. If you're looking to protect your property, people, or anything else, you should definitely look into a geofence.

https://grouphowto.com/ipsgeofence-app/

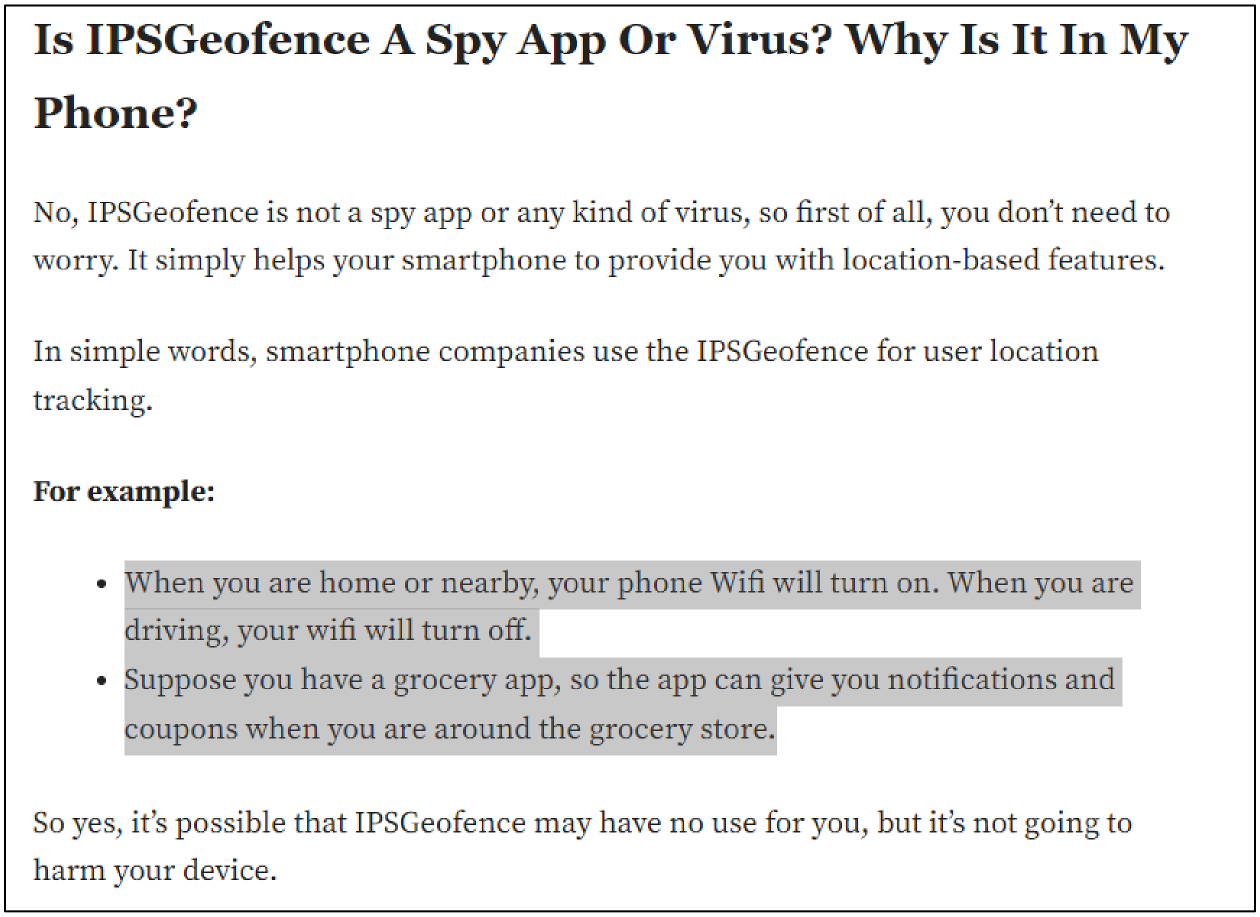


What Is IPSGeofence?

IPSGeofence is a bloatware app, which means it comes pre-installed on your phone, and you can't uninstall it like a normal app on your phone; that's why you are seeing this application in your smartphone.

As I said above, that IPSGeofence app helps your smartphone to give you location-based features,

https://quaries.com/what-is-ipsgeofence-and-how-to-uninstall-it/



Is IPSGeofence A Spy App Or Virus? Why Is It In My Phone?

No, IPSGeofence is not a spy app or any kind of virus, so first of all, you don't need to worry. It simply helps your smartphone to provide you with location-based features.

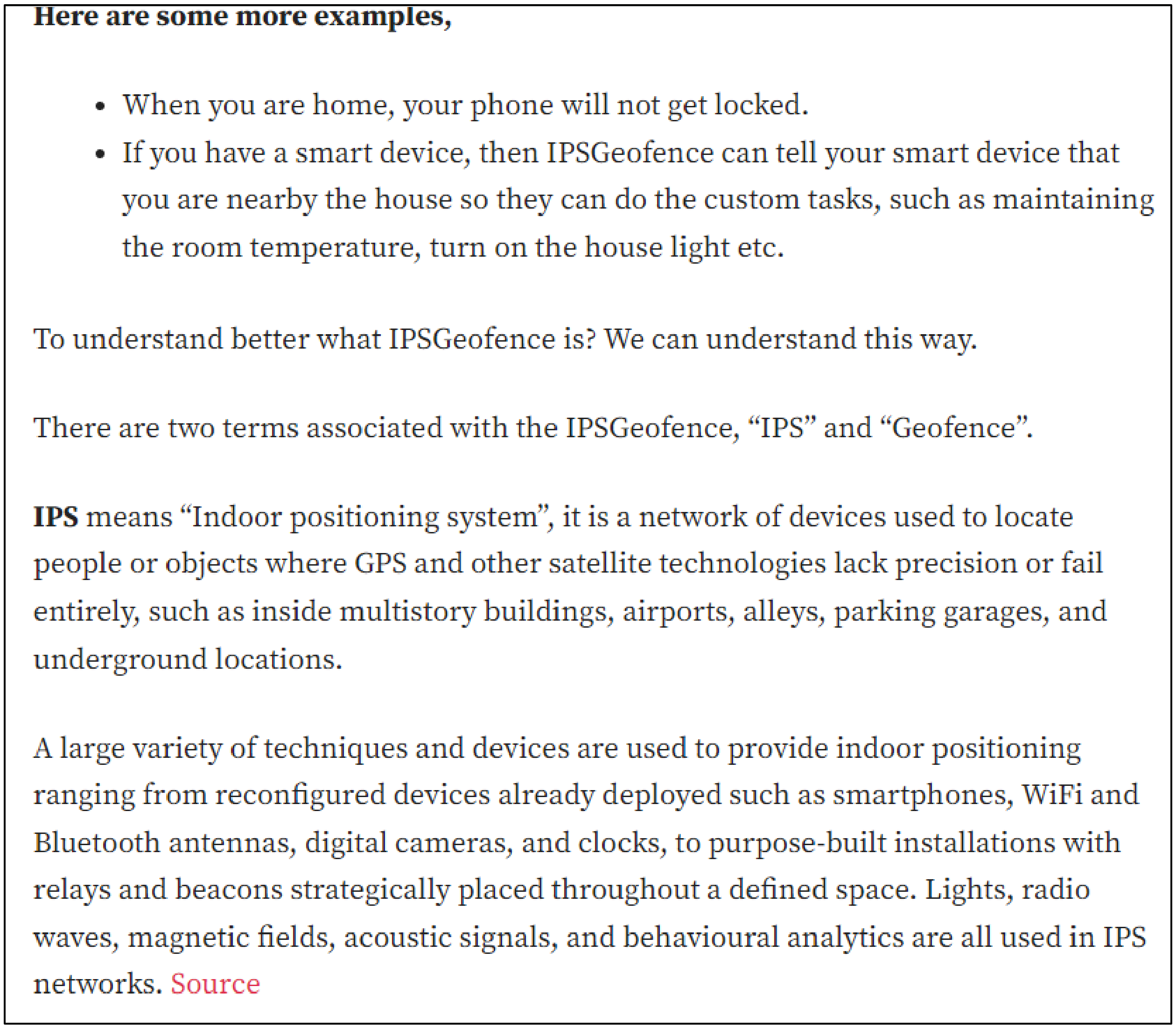
In simple words, smartphone companies use the IPSGeofence for user location tracking.

For example:

- When you are home or nearby, your phone Wifi will turn on. When you are driving, your wifi will turn off.
- Suppose you have a grocery app, so the app can give you notifications and coupons when you are around the grocery store.

So yes, it's possible that IPSGeofence may have no use for you, but it's not going to harm your device.

Id.



Here are some more examples,

- When you are home, your phone will not get locked.
- If you have a smart device, then IPSGeofence can tell your smart device that you are nearby the house so they can do the custom tasks, such as maintaining the room temperature, turn on the house light etc.

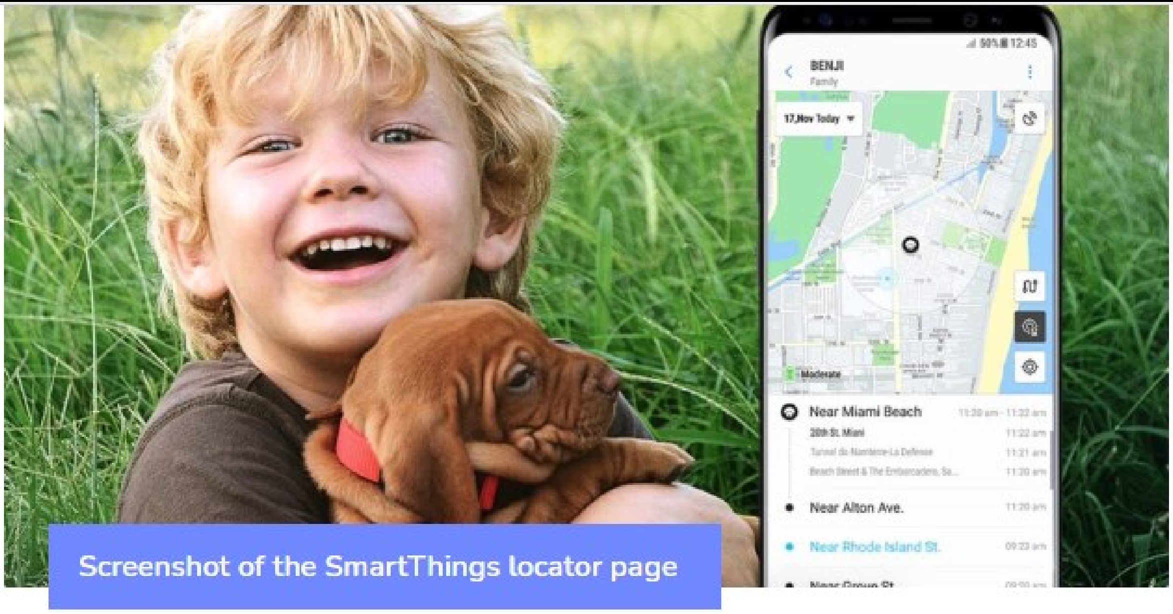
To understand better what IPSGeofence is? We can understand this way.

There are two terms associated with the IPSGeofence, "IPS" and "Geofence".

IPS means "Indoor positioning system", it is a network of devices used to locate people or objects where GPS and other satellite technologies lack precision or fail entirely, such as inside multistory buildings, airports, alleys, parking garages, and underground locations.

A large variety of techniques and devices are used to provide indoor positioning ranging from reconfigured devices already deployed such as smartphones, WiFi and Bluetooth antennas, digital cameras, and clocks, to purpose-built installations with relays and beacons strategically placed throughout a defined space. Lights, radio waves, magnetic fields, acoustic signals, and behavioural analytics are all used in IPS networks. Source

Id.



Screenshot of the SmartThings locator page

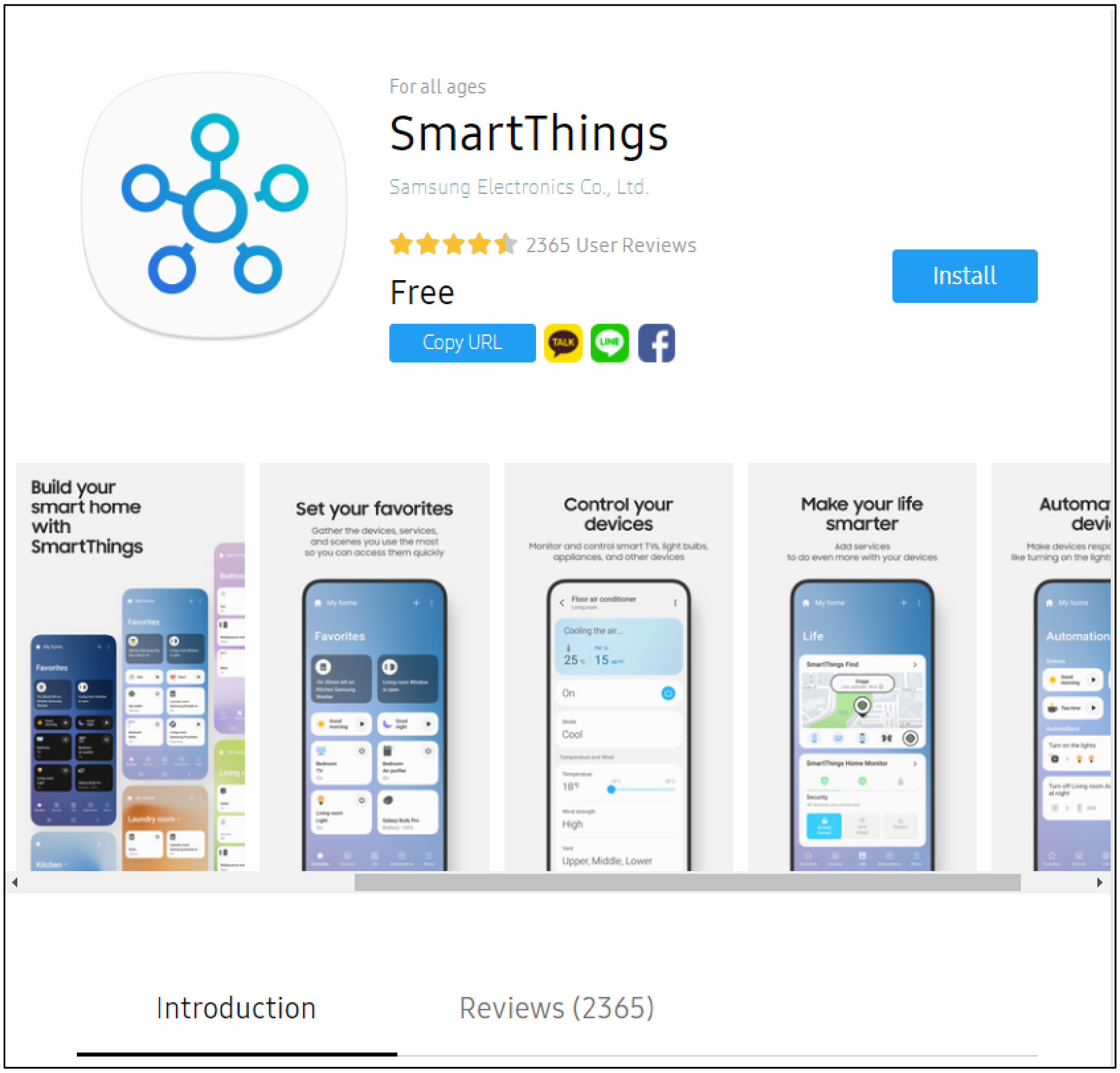
Source: Samsung

The acronym IPS, meanwhile, stands for *Indoor Positioning System* , which, as its name suggests, aims to allow more precise localization in closed places (house, shopping center, etc.). .). It can make it possible to find the tracker already mentioned more precisely in a smaller space.

Let's take the description sentence of IPSGeofence: " *It can only be activated if a user activates services that pass through IPSGeofence.* This app therefore seems to allow the proper functioning of a *tracking* system like the one used by SmartThings. As of now, Samsung hasn't confirmed that IPSGeofencing has anything to do with its SmartThings app and tracking features, although this is the most likely solution. Contacted, the company has not yet returned to us.

In any case, this is not a new or revolutionary solution: this kind of system has been working for a long time and can be programmed on different smartphones to generate specific small actions when your device enters an area (for example, turning on your bulbs connected when you enter the perimeter of your home or switch your smartphone to Wi-Fi automatically when you enter your home). With its AirTags , Apple is even developing an object that works in the same way as Samsung's roller.

https://www.numerama.com/tech/619559-ipsgeofence-a-quoi-sert-lapp-samsung-quil-est-impossible-de-desactiver.html



For all ages

SmartThings

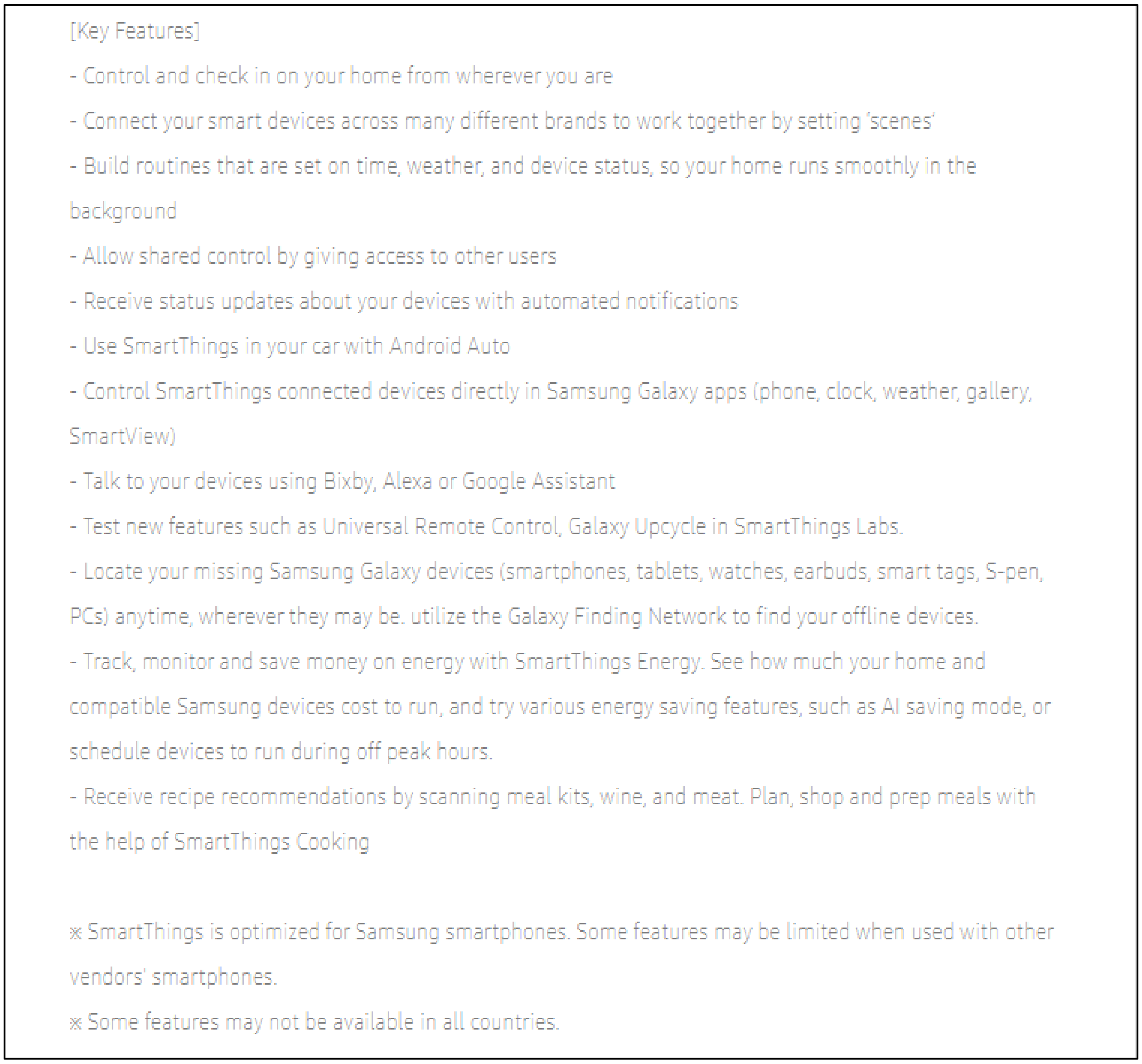
Samsung Electronics Co., Ltd.

2365 User Reviews

Free

Copy URL
Install
Build your smart home with SmartThings
Set your favorites
Gather the devices, services, and scenes you use the most so you can access them quickly
Control your devices
Monitor and control smart TVs, light bulbs, appliances, and other devices
Make your life smarter
Add services to do even more with your devices
Automa
Introduction
Reviews (2365)

https://galaxystore.samsung.com/prepost/000004262296?langCd=en

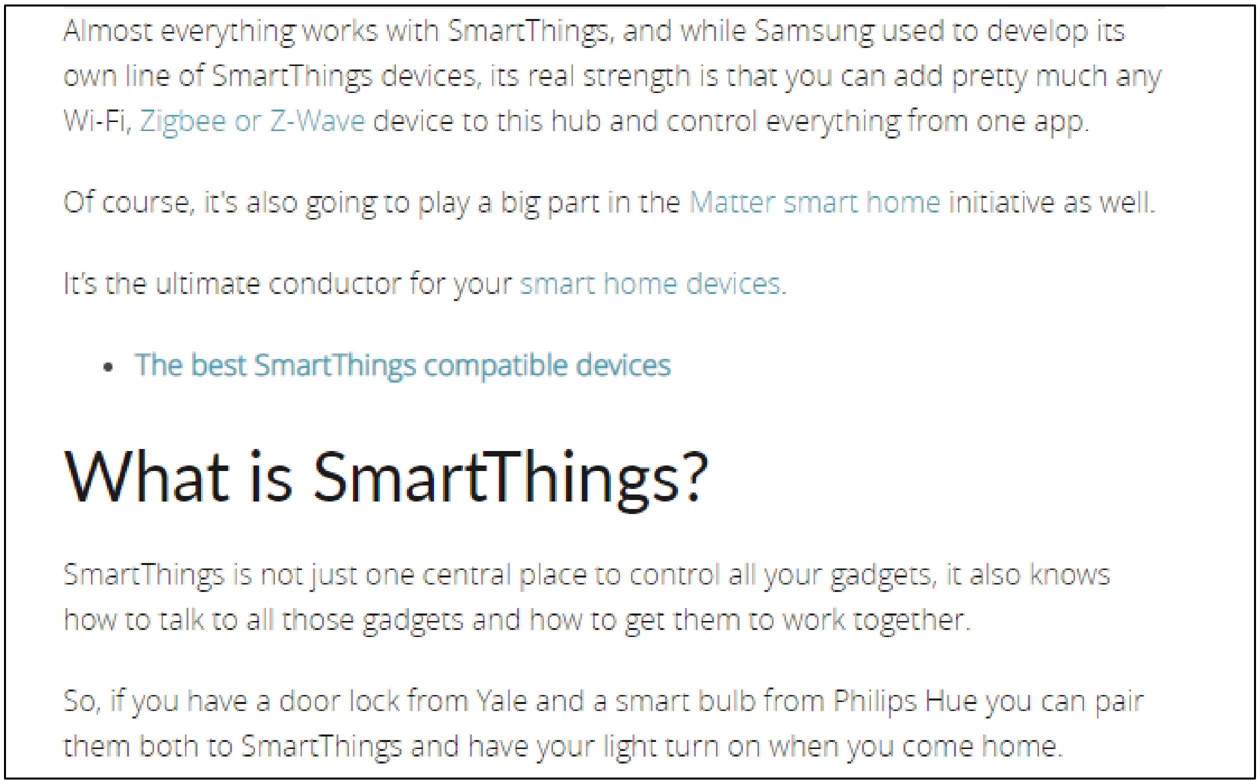

[Key Features]

- Control and check in on your home from wherever you are
- Connect your smart devices across many different brands to work together by setting 'scenes'
- Build routines that are set on time, weather, and device status, so your home runs smoothly in the background
- Allow shared control by giving access to other users
- Receive status updates about your devices with automated notifications
- Use SmartThings in your car with Android Auto
- Control SmartThings connected devices directly in Samsung Galaxy apps (phone, clock, weather, gallery, SmartView)
- Talk to your devices using Bixby, Alexa or Google Assistant
- Test new features such as Universal Remote Control, Galaxy Upcycle in SmartThings Labs.
- Locate your missing Samsung Galaxy devices (smartphones, tablets, watches, earbuds, smart tags, S-pen, PCs) anytime, wherever they may be. utilize the Galaxy Finding Network to find your offline devices.
- Track, monitor and save money on energy with SmartThings Energy. See how much your home and compatible Samsung devices cost to run, and try various energy saving features, such as AI saving mode, or schedule devices to run during off peak hours.
- Receive recipe recommendations by scanning meal kits, wine, and meat. Plan, shop and prep meals with the help of SmartThings Cooking

※ SmartThings is optimized for Samsung smartphones. Some features may be limited when used with other vendors' smartphones.

※ Some features may not be available in all countries.

Id.



Almost everything works with SmartThings, and while Samsung used to develop its own line of SmartThings devices, its real strength is that you can add pretty much any Wi-Fi, Zigbee or Z-Wave device to this hub and control everything from one app.

Of course, it's also going to play a big part in the Matter smart home initiative as well.

It's the ultimate conductor for your smart home devices.

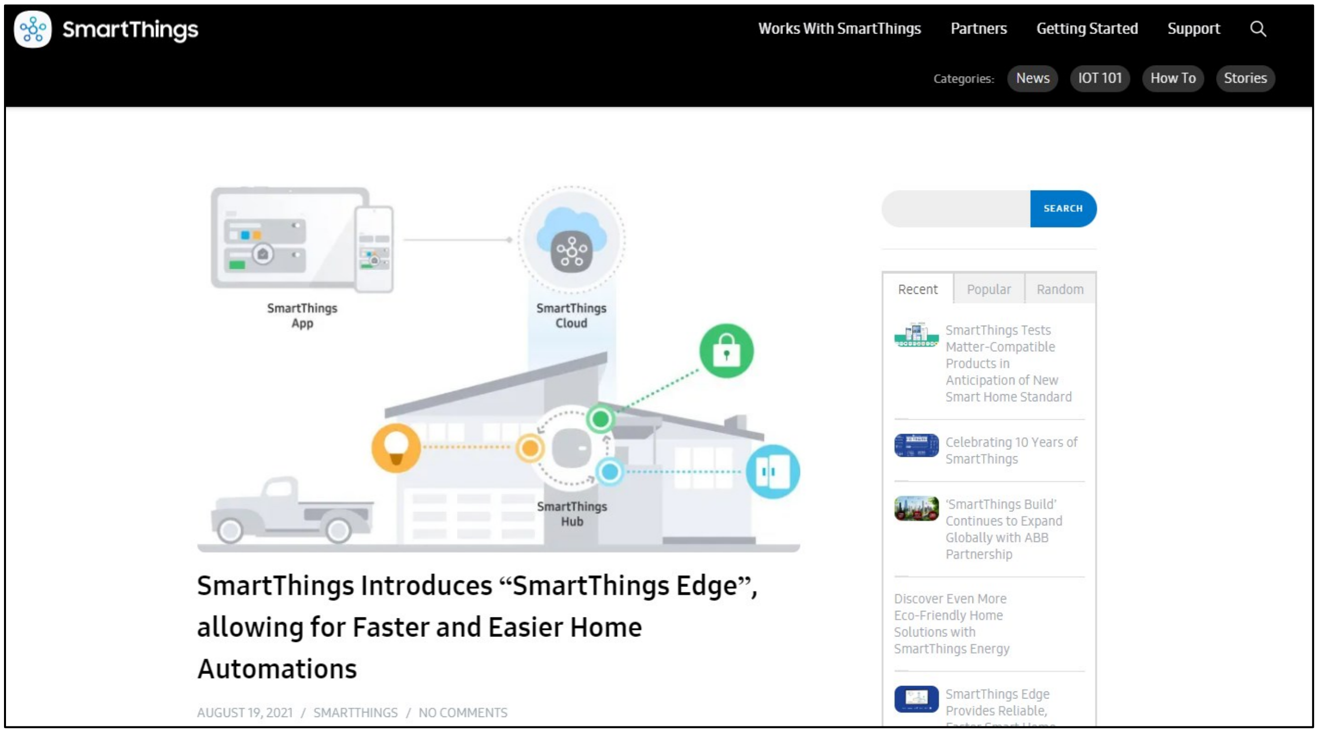
- The best SmartThings compatible devices

What is SmartThings?

SmartThings is not just one central place to control all your gadgets, it also knows how to talk to all those gadgets and how to get them to work together.

So, if you have a door lock from Yale and a smart bulb from Philips Hue you can pair them both to SmartThings and have your light turn on when you come home.

https://www.the-ambient.com/guides/samsung-smartthings-guide-smart-home-163



SmartThings Introduces "SmartThings Edge", allowing for Faster and Easier Home Automations

AUGUST 19, 2021 / SMARTTHINGS / NO COMMENTS

SmartThings Tests Matter-Compatible Products in Anticipation of New Smart Home Standard

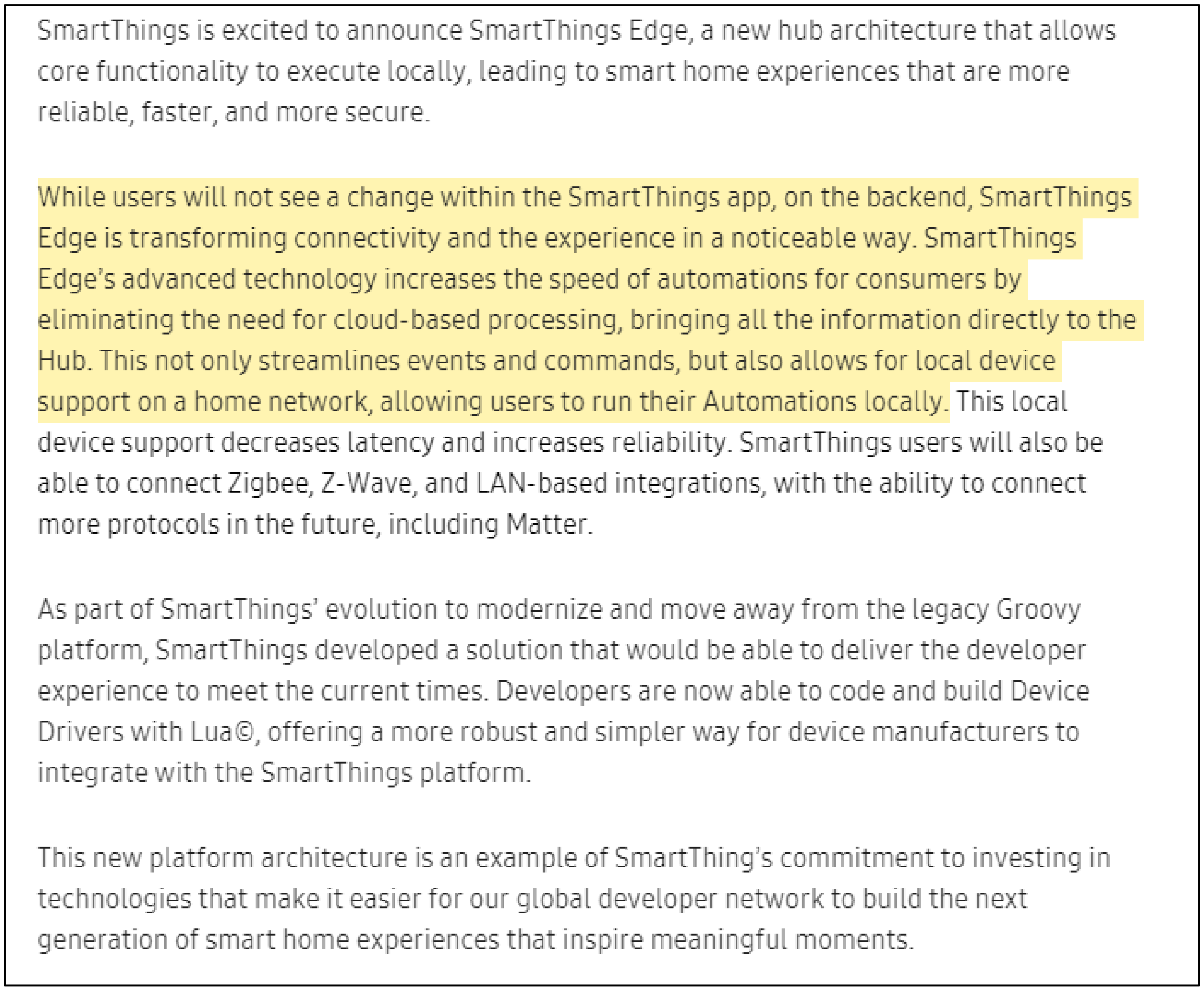
Celebrating 10 Years of SmartThings

'SmartThings Build' Continues to Expand Globally with ABB Partnership

Discover Even More Eco-Friendly Home Solutions with SmartThings Energy

SmartThings Edge Provides Reliable,

https://blog.smartthings.com/roundups/smartthings-introduces-smartthings-edgeallowing-for-faster-and-easier-home-automations/

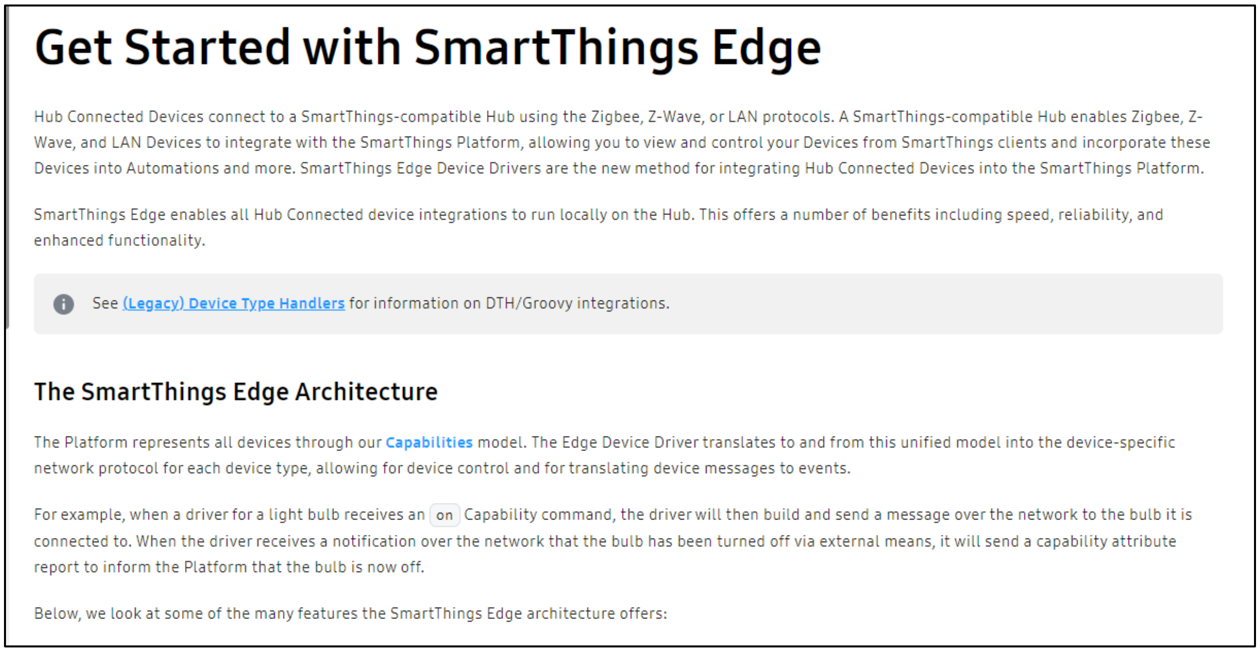

SmartThings is excited to announce SmartThings Edge, a new hub architecture that allows core functionality to execute locally, leading to smart home experiences that are more reliable, faster, and more secure.

While users will not see a change within the SmartThings app, on the backend, SmartThings Edge is transforming connectivity and the experience in a noticeable way. SmartThings Edge's advanced technology increases the speed of automations for consumers by eliminating the need for cloud-based processing, bringing all the information directly to the Hub. This not only streamlines events and commands, but also allows for local device support on a home network, allowing users to run their Automations locally. This local device support decreases latency and increases reliability. SmartThings users will also be able to connect Zigbee, Z-Wave, and LAN-based integrations, with the ability to connect more protocols in the future, including Matter.

As part of SmartThings' evolution to modernize and move away from the legacy Groovy platform, SmartThings developed a solution that would be able to deliver the developer experience to meet the current times. Developers are now able to code and build Device Drivers with Lua©, offering a more robust and simpler way for device manufacturers to integrate with the SmartThings platform.

This new platform architecture is an example of SmartThing's commitment to investing in technologies that make it easier for our global developer network to build the next generation of smart home experiences that inspire meaningful moments.

Id.



Get Started with SmartThings Edge

Hub Connected Devices connect to a SmartThings-compatible Hub using the Zigbee, Z-Wave, or LAN protocols. A SmartThings-compatible Hub enables Zigbee, Z-Wave, and LAN Devices to integrate with the SmartThings Platform, allowing you to view and control your Devices from SmartThings clients and incorporate these Devices into Automations and more. SmartThings Edge Device Drivers are the new method for integrating Hub Connected Devices into the SmartThings Platform.

SmartThings Edge enables all Hub Connected device integrations to run locally on the Hub. This offers a number of benefits including speed, reliability, and enhanced functionality.

See (Legacy) Device Type Handlers for information on DTH/Groovy integrations.

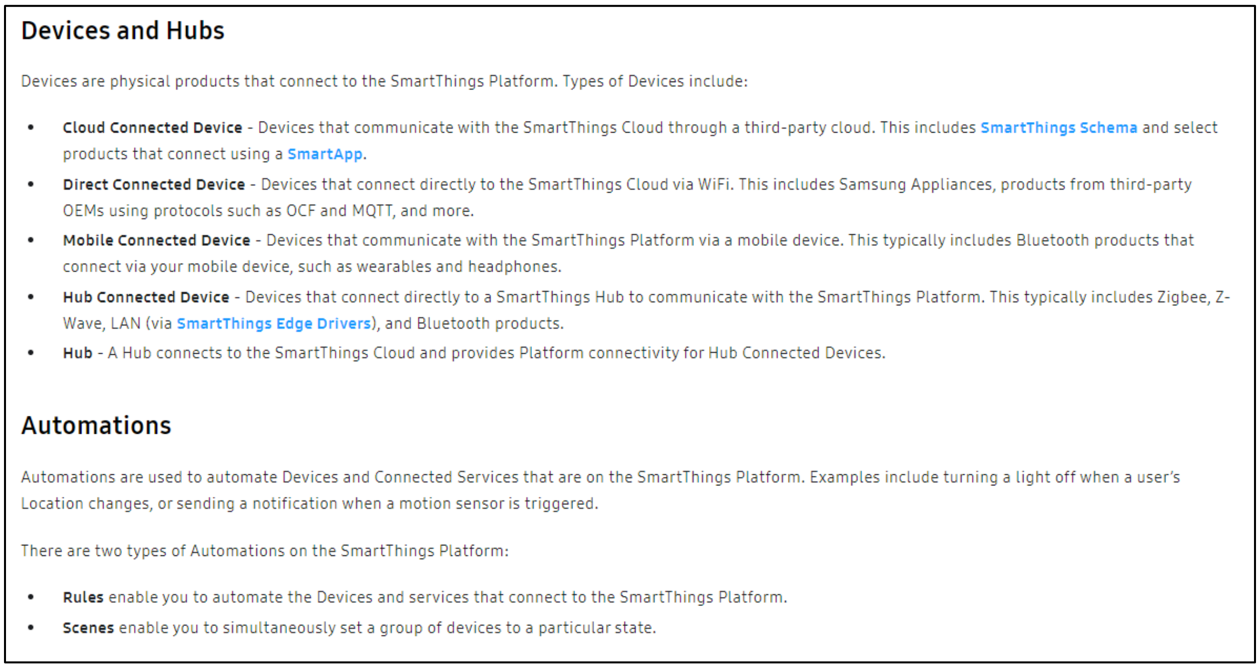
The SmartThings Edge Architecture

The Platform represents all devices through our **Capabilities** model. The Edge Device Driver translates to and from this unified model into the device-specific network protocol for each device type, allowing for device control and for translating device messages to events.

For example, when a driver for a light bulb receives an `on` Capability command, the driver will then build and send a message over the network to the bulb it is connected to. When the driver receives a notification over the network that the bulb has been turned off via external means, it will send a capability attribute report to inform the Platform that the bulb is now off.

Below, we look at some of the many features the SmartThings Edge architecture offers:

https://developer-preview.smartthings.com/docs/devices/hub-connected/get-started/



Devices and Hubs

Devices are physical products that connect to the SmartThings Platform. Types of Devices include:

- **Cloud Connected Device** - Devices that communicate with the SmartThings Cloud through a third-party cloud. This includes **SmartThings Schema** and select products that connect using a **SmartApp**.
- **Direct Connected Device** - Devices that connect directly to the SmartThings Cloud via WiFi. This includes Samsung Appliances, products from third-party OEMs using protocols such as OCF and MQTT, and more.
- **Mobile Connected Device** - Devices that communicate with the SmartThings Platform via a mobile device. This typically includes Bluetooth products that connect via your mobile device, such as wearables and headphones.
- **Hub Connected Device** - Devices that connect directly to a SmartThings Hub to communicate with the SmartThings Platform. This typically includes Zigbee, Z-Wave, LAN (via **SmartThings Edge Drivers**), and Bluetooth products.
- **Hub** - A Hub connects to the SmartThings Cloud and provides Platform connectivity for Hub Connected Devices.

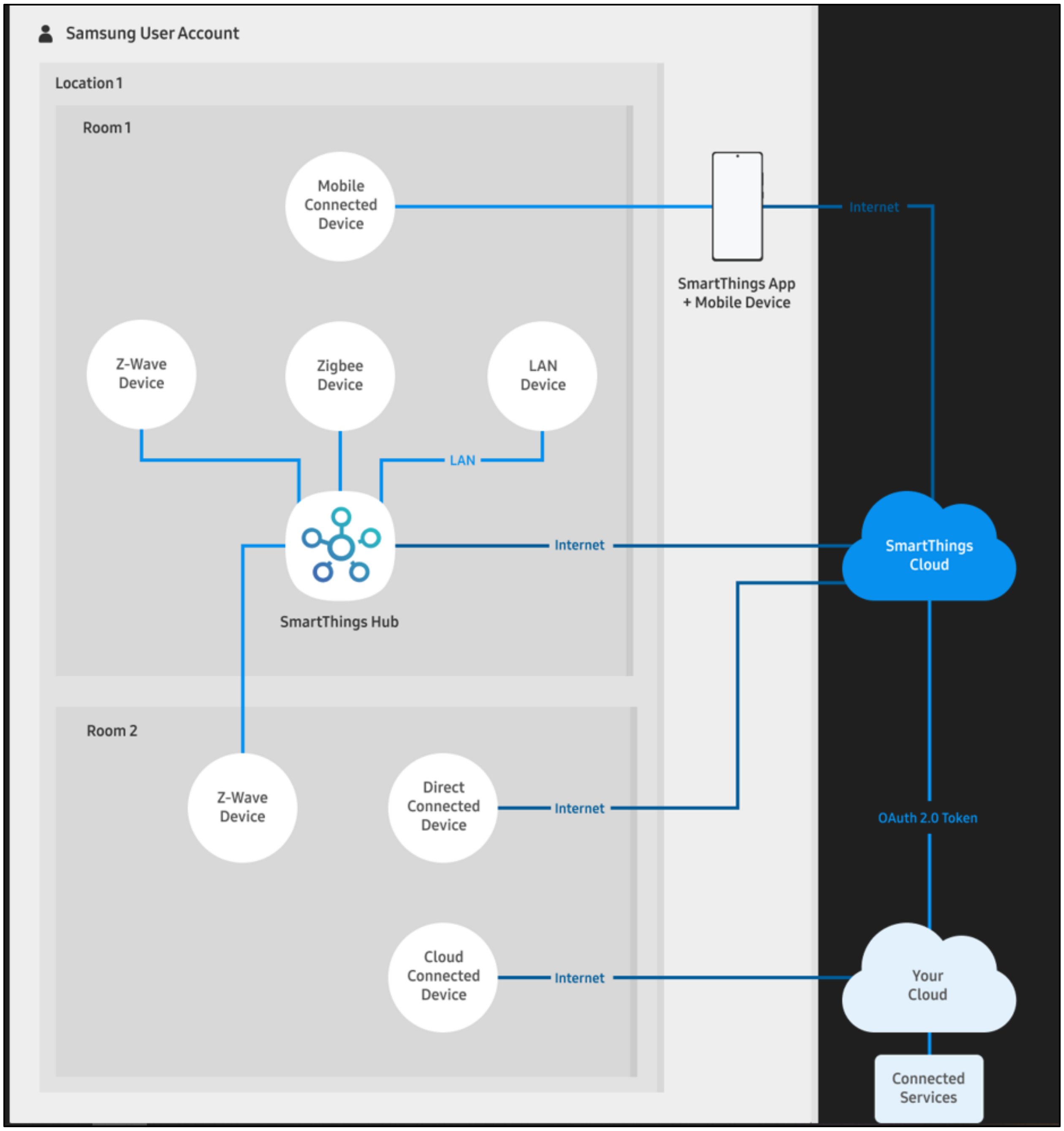
Automations

Automations are used to automate Devices and Connected Services that are on the SmartThings Platform. Examples include turning a light off when a user's Location changes, or sending a notification when a motion sensor is triggered.

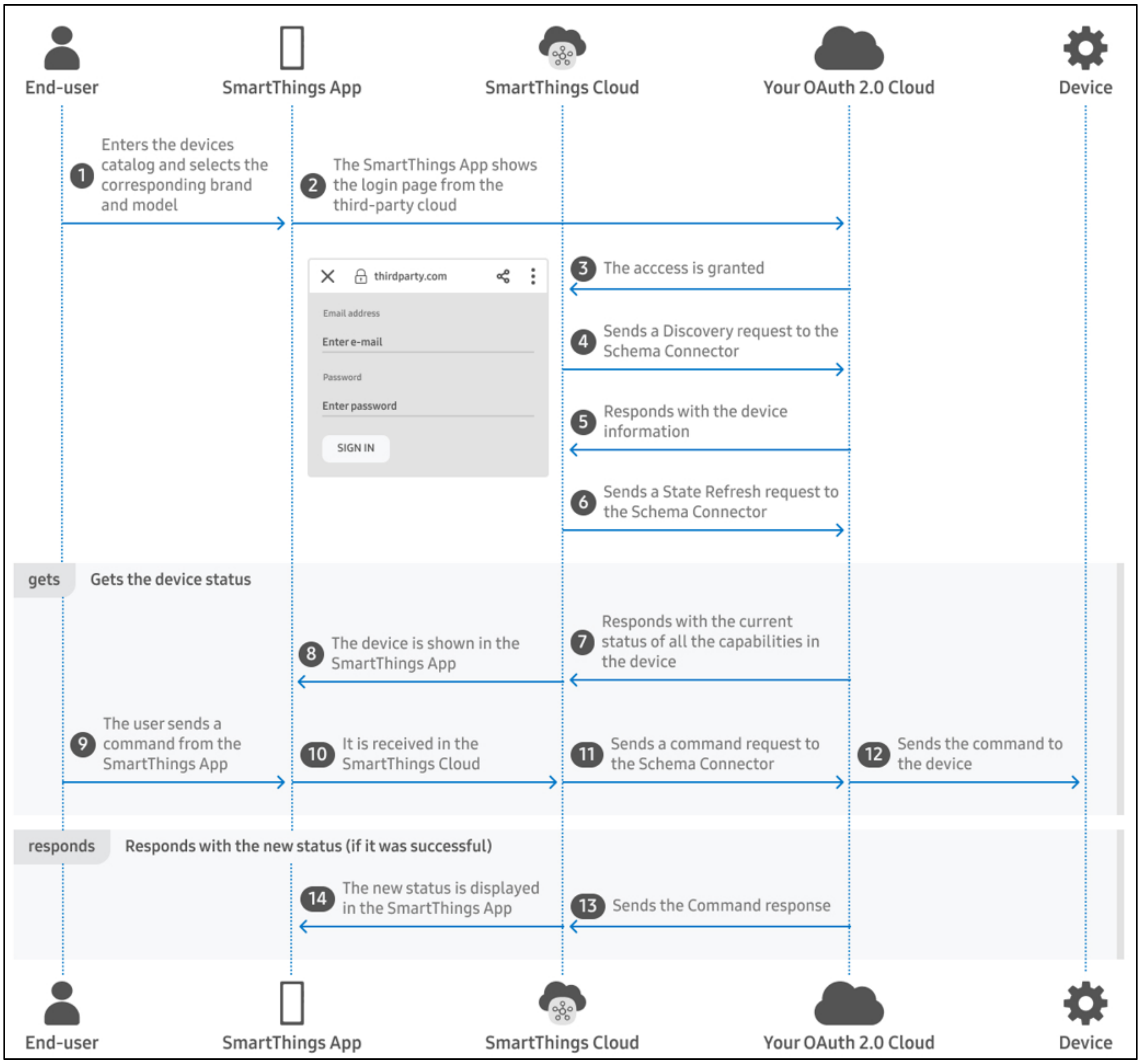
There are two types of Automations on the SmartThings Platform:

- **Rules** enable you to automate the Devices and services that connect to the SmartThings Platform.
- **Scenes** enable you to simultaneously set a group of devices to a particular state.

Id.



https://developer-preview.smartthings.com/docs/getting-started/architecture-of-smartthings


End-user
SmartThings App
SmartThings Cloud
Your OAuth 2.0 Cloud
Device
1 Enters the devices catalog and selects the corresponding brand and model
2 The SmartThings App shows the login page from the third-party cloud
thirdparty.com
Email address
Enter e-mail
Password
Enter password
SIGN IN
3 The acccess is granted
4 Sends a Discovery request to the Schema Connector
5 Responds with the device information
6 Sends a State Refresh request to the Schema Connector
gets
Gets the device status
7 Responds with the current status of all the capabilities in the device
8 The device is shown in the SmartThings App
9 The user sends a command from the SmartThings App
10 It is received in the SmartThings Cloud
11 Sends a command request to the Schema Connector
12 Sends the command to the device
responds
Responds with the new status (if it was successful)
13 Sends the Command response
14 The new status is displayed in the SmartThings App
End-user
SmartThings App
SmartThings Cloud
Your OAuth 2.0 Cloud
Device

https://developer-preview.smartthings.com/docs/devices/cloud-connected/get-started

38. By engaging in the conduct described herein, Samsung has injured Plaintiffs and is thus liable for infringement of the ’296 Patent, pursuant to 35 U.S.C. § 271. Samsung has committed these acts of infringement without license or authorization.

39. As a result of Samsung's infringement of the '296 Patent, Plaintiffs have suffered monetary damages and are entitled to a monetary judgment in an amount adequate to compensate for Samsung's past infringement, together with interests and costs. In addition, Samsung's infringement is causing irreparable harm and monetary damage to Plaintiffs and will continue to do so unless and until Samsung is enjoined by the Court.

COUNT II: CLAIM FOR PATENT INFRINGEMENT OF THE '247 PATENT

40. Plaintiffs repeat and reallege the allegations in paragraphs 1-39 as if fully set forth herein.

41. Samsung has infringed, contributed to the infringement of, and/or induced infringement of the '247 Patent by making, using, selling, offering for sale, or importing into the United States, or by intending that others make, use, import into, offer for sale, or sell in the United States, products and/or methods covered by one or more claims of the '247 Patent including, but not limited to, mobile devices and components thereof.

42. Upon information and belief, Samsung's mobile devices, including smartphones, tablets, and watches, that infringe one or more claims of the '247 Patent include at least the following products, as well as products with reasonably similar functionality:

- Galaxy S series, Galaxy Note series, Galaxy Z Flip series, Galaxy Fold series, and Galaxy A series smartphones;
- Galaxy Tab S series, Galaxy Tab A series, and Galaxy Book S series tablets; and

- Galaxy Watch 3 series, Galaxy Watch 3 Titanium series, Galaxy Watch Active2 series, Galaxy Watch Active series, Galaxy Watch series, and Galaxy Fit2 series watches.

Upon information and belief, these products are offered in several varieties, including different options for size, display, processor, storage, and battery. The products listed are exemplary, and Plaintiffs will be able to provide a more comprehensive list after discovery. Upon information and belief, Samsung also infringes one or more claims of the '247 Patent through its use of geofencing or location information in its advertising or content delivery efforts or its own applications it develops.

43. For example, upon information and belief, Samsung's mobile devices infringe at least claim 8 of the '247 Patent. Samsung makes, uses, sells, offers for sale, imports, exports, supplies, or distributes within the United States its mobile devices and thus directly infringes the '247 Patent.

44. Samsung indirectly infringes the '247 Patent as provided by 35 U.S.C. § 271(b) by inducing infringement by others, such as manufacturers, resellers, and end-user customers in this District and throughout the United States. For example, direct infringement is the result of activities performed by manufacturers, resellers, or operators/carriers of Samsung's mobile devices, who, upon information and belief, perform each step of the claimed invention as directed by Samsung. Samsung received actual notice of the '247 Patent at least as early as the filing of this Complaint.

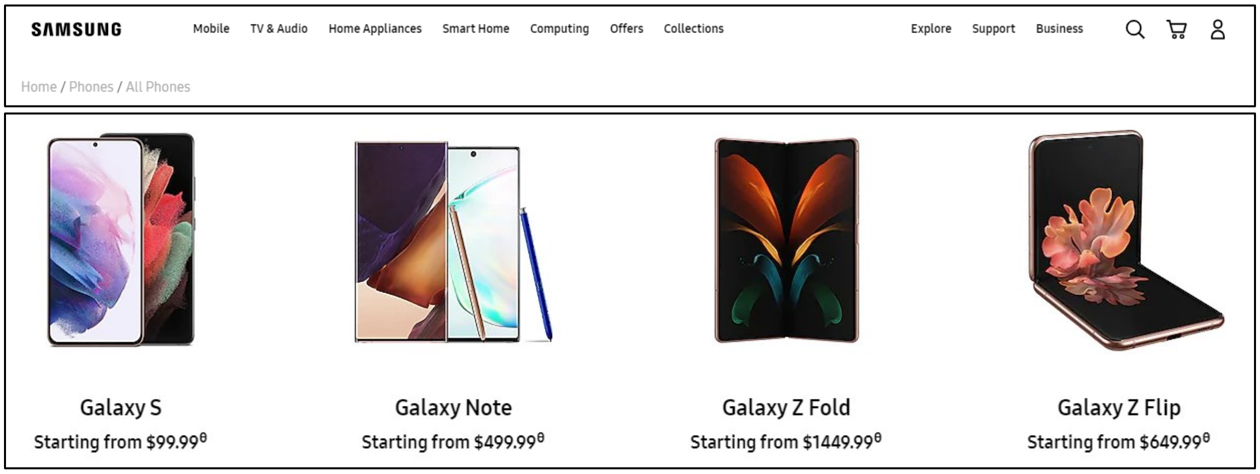
45. Upon information and belief, Samsung's affirmative acts of selling Samsung's mobile devices, causing Samsung's mobile devices to be manufactured, and providing directions,

instructions, schematics, diagrams, or designs to its manufacturers, resellers, or operators/carriers to make or use Samsung's mobile devices in a manner that directly infringes the '247 Patent induce infringement by others, such as manufacturers, resellers, or operators/carriers in this District and throughout the United States. Upon information and belief, through its manufacture and sales of Samsung's mobile devices, Samsung performed the acts that constitute induced infringement with knowledge or willful blindness that the induced acts would constitute infringement.

46. Samsung also indirectly infringes the '247 Patent by contributing to infringement by others, such as manufacturers, resellers, manufacturers, and operators/carriers, in accordance with 35 U.S.C. § 271(c) in this District and throughout the United States. Upon information and belief, direct infringement is the result of activities performed by manufacturers, resellers, or operators/carriers of Samsung's mobile devices.

47. Upon information and belief, Samsung's affirmative acts of selling Samsung's mobile devices and causing Samsung's mobile devices to be manufactured and sold contribute to Samsung's manufacturers, resellers, and operators/carriers making or using Samsung's mobile devices in a normal and customary way that infringes the '247 Patent. Upon information and belief, Samsung's mobile devices constitute the material part of Plaintiffs' patented invention, have no substantial non-infringing uses, and are known by Samsung to be especially made or especially adapted for use to infringe the '247 Patent.

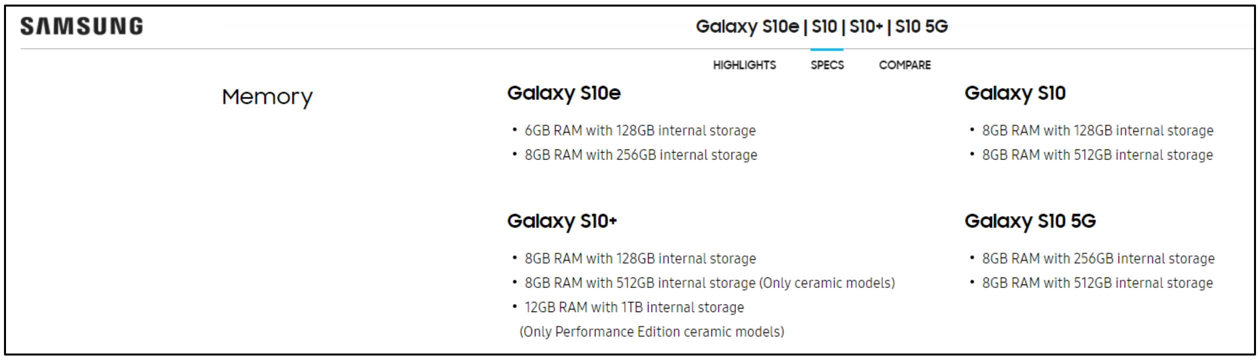
48. Samsung provides a system comprising: a memory; a communications interface operably coupled to the memory; and at least one processor operably coupled to the memory and the communications interface, the processor configured to execute a program of instructions:


SAMSUNG Mobile TV & Audio Home Appliances Smart Home Computing Offers Collections Explore Support Business

Home / Phones / All Phones

Galaxy S	Galaxy Note	Galaxy Z Fold	Galaxy Z Flip
Starting from $99.99[θ]	Starting from $499.99[θ]	Starting from $1449.99[θ]	Starting from $649.99[θ]

https://www.samsung.com/us/


SAMSUNG Galaxy S10e | S10 | S10+ | S10 5G

HIGHLIGHTS SPECS COMPARE

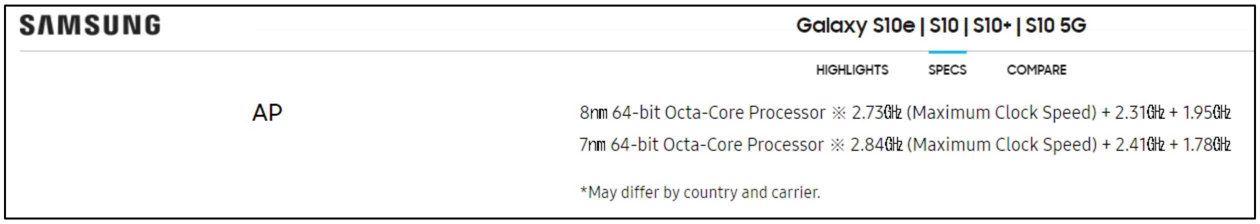
Memory

Galaxy S10e
- 6GB RAM with 128GB internal storage
- 8GB RAM with 256GB internal storage

Galaxy S10
- 8GB RAM with 128GB internal storage
- 8GB RAM with 512GB internal storage

Galaxy S10+
- 8GB RAM with 128GB internal storage
- 8GB RAM with 512GB internal storage (Only ceramic models)
- 12GB RAM with 1TB internal storage (Only Performance Edition ceramic models)

Galaxy S10 5G
- 8GB RAM with 256GB internal storage
- 8GB RAM with 512GB internal storage


SAMSUNG Galaxy S10e | S10 | S10+ | S10 5G

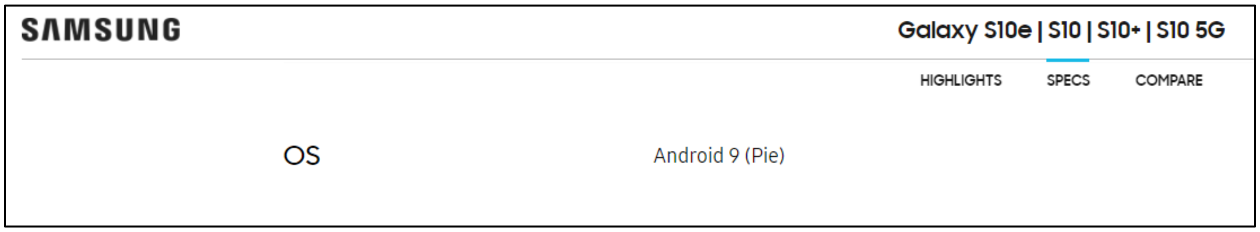
HIGHLIGHTS SPECS COMPARE

AP

8㎚ 64-bit Octa-Core Processor ※ 2.73㎓ (Maximum Clock Speed) + 2.31㎓ + 1.95㎓

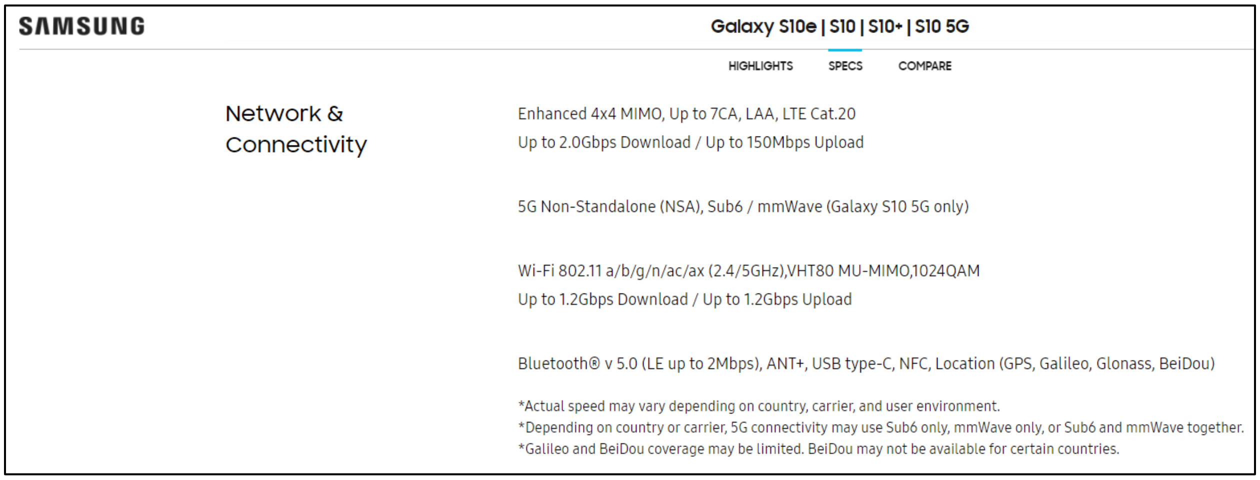
7㎚ 64-bit Octa-Core Processor ※ 2.84㎓ (Maximum Clock Speed) + 2.41㎓ + 1.78㎓

*May differ by country and carrier.


SAMSUNG Galaxy S10e | S10 | S10+ | S10 5G

HIGHLIGHTS SPECS COMPARE

OS Android 9 (Pie)



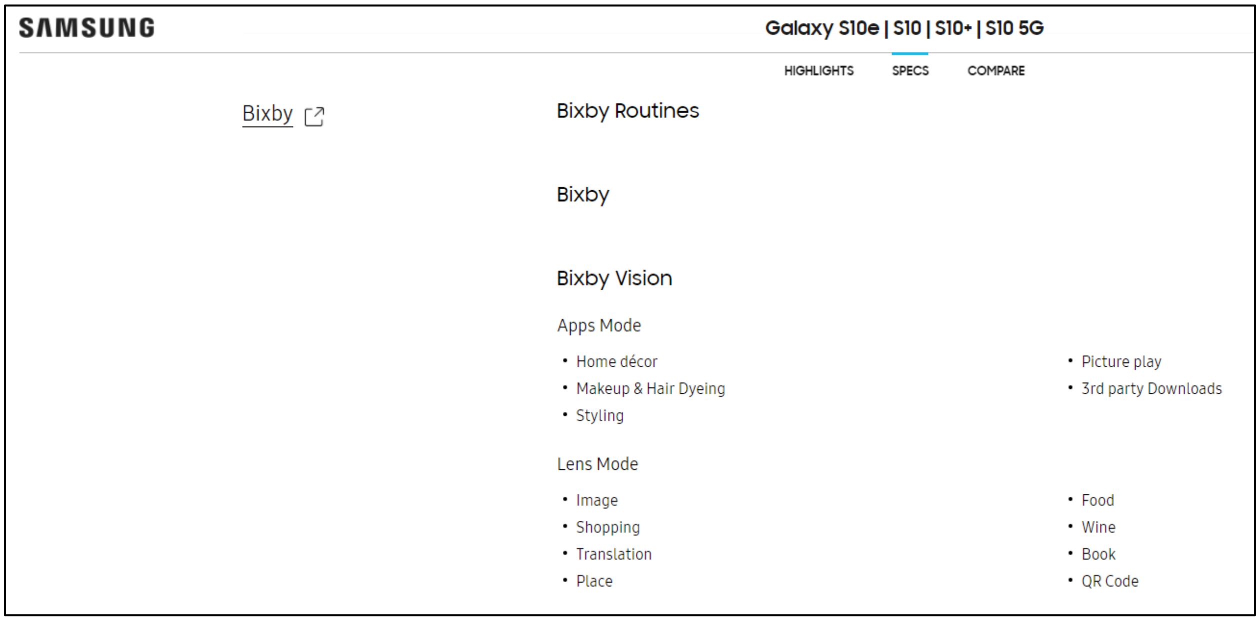
SAMSUNG

Galaxy S10e | S10 | S10+ | S10 5G

HIGHLIGHTS SPECS COMPARE

Network & Connectivity	Enhanced 4x4 MIMO, Up to 7CA, LAA, LTE Cat.20 Up to 2.0Gbps Download / Up to 150Mbps Upload
	5G Non-Standalone (NSA), Sub6 / mmWave (Galaxy S10 5G only)
	Wi-Fi 802.11 a/b/g/n/ac/ax (2.4/5GHz),VHT80 MU-MIMO,1024QAM Up to 1.2Gbps Download / Up to 1.2Gbps Upload
	Bluetooth® v 5.0 (LE up to 2Mbps), ANT+, USB type-C, NFC, Location (GPS, Galileo, Glonass, BeiDou)

*Actual speed may vary depending on country, carrier, and user environment.
*Depending on country or carrier, 5G connectivity may use Sub6 only, mmWave only, or Sub6 and mmWave together.
*Galileo and BeiDou coverage may be limited. BeiDou may not be available for certain countries.



SAMSUNG

Galaxy S10e | S10 | S10+ | S10 5G

HIGHLIGHTS SPECS COMPARE

Bixby

Bixby Routines

Bixby

Bixby Vision

Apps Mode

- Home décor
- Makeup & Hair Dyeing
- Styling
- Picture play
- 3rd party Downloads

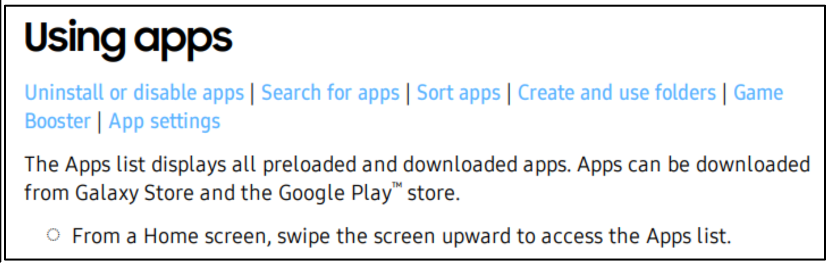
Lens Mode

- Image
- Shopping
- Translation
- Place
- Food
- Wine
- Book
- QR Code

https://www.samsung.com/global/galaxy/galaxy-s10/specs/

49. Samsung provides a processor configured to execute a program of instructions, the program of instructions comprises: at least one instruction for registering a plurality of application programs for use with a content delivery platform; at least one instruction for establishing a plurality of perimeters defining a plurality of geographic areas; at least one instruction for receiving, from at least a particular one of the registered application programs, request to obtain an interest in a designated geographic area; at least one instruction for determining if the interest

in the designated geographic area is to be provided to at least the particular one of the registered application programs, including as demonstrated in the exemplary text and images below:



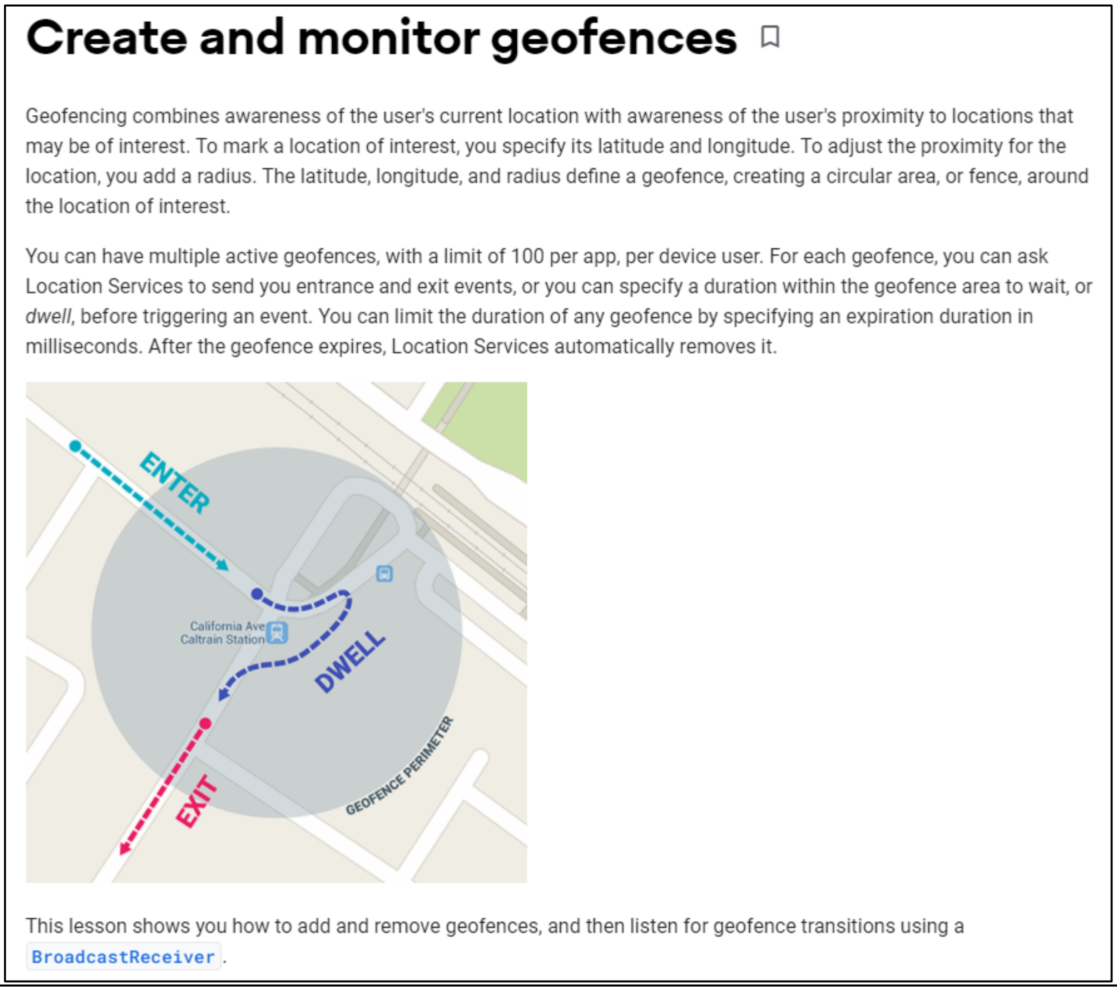
Using apps

Uninstall or disable apps | Search for apps | Sort apps | Create and use folders | Game Booster | App settings

The Apps list displays all preloaded and downloaded apps. Apps can be downloaded from Galaxy Store and the Google Play™ store.

- From a Home screen, swipe the screen upward to access the Apps list.

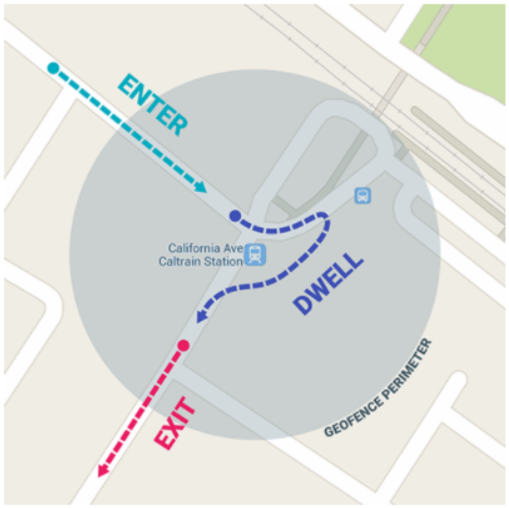
https://downloadcenter.samsung.com/content/UM/202103/20210304004310357/ATT_SM-G970U_G973U_G975U_EN_UM_R_11.0_022221_FINAL.pdf



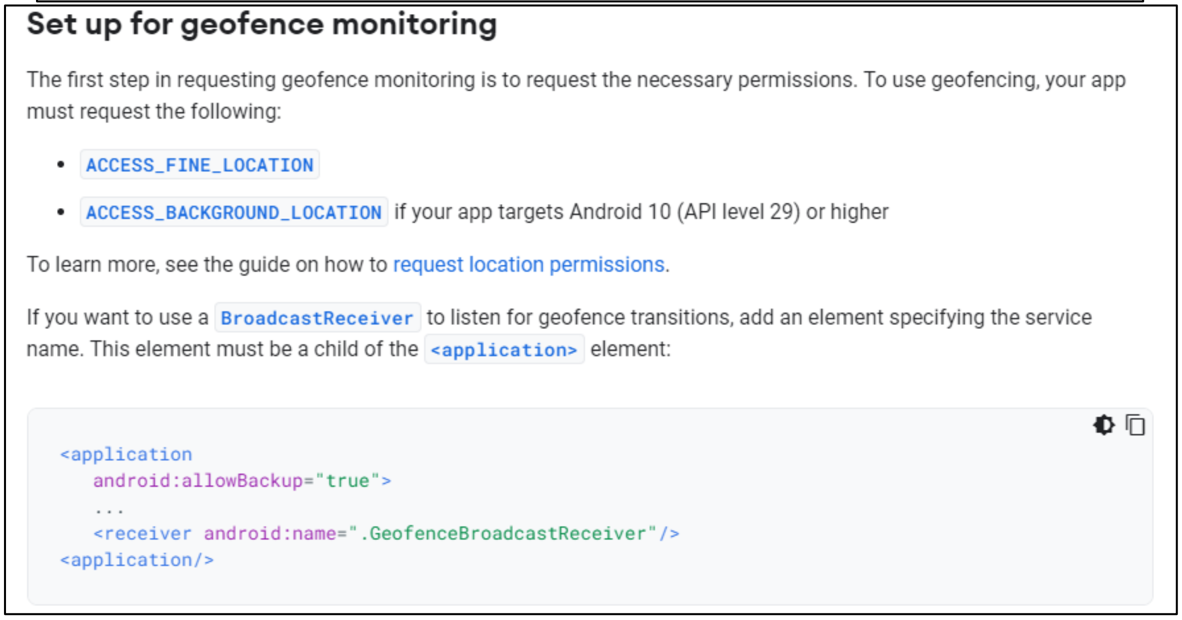
Create and monitor geofences

Geofencing combines awareness of the user's current location with awareness of the user's proximity to locations that may be of interest. To mark a location of interest, you specify its latitude and longitude. To adjust the proximity for the location, you add a radius. The latitude, longitude, and radius define a geofence, creating a circular area, or fence, around the location of interest.

You can have multiple active geofences, with a limit of 100 per app, per device user. For each geofence, you can ask Location Services to send you entrance and exit events, or you can specify a duration within the geofence area to wait, or *dwell*, before triggering an event. You can limit the duration of any geofence by specifying an expiration duration in milliseconds. After the geofence expires, Location Services automatically removes it.



This lesson shows you how to add and remove geofences, and then listen for geofence transitions using a `BroadcastReceiver`.



Set up for geofence monitoring

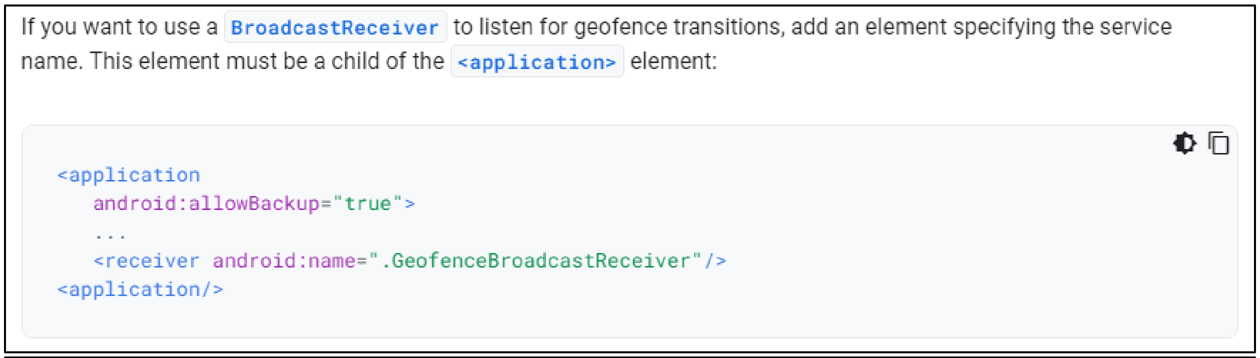
The first step in requesting geofence monitoring is to request the necessary permissions. To use geofencing, your app must request the following:

- `ACCESS_FINE_LOCATION`
- `ACCESS_BACKGROUND_LOCATION` if your app targets Android 10 (API level 29) or higher

To learn more, see the guide on how to request location permissions.

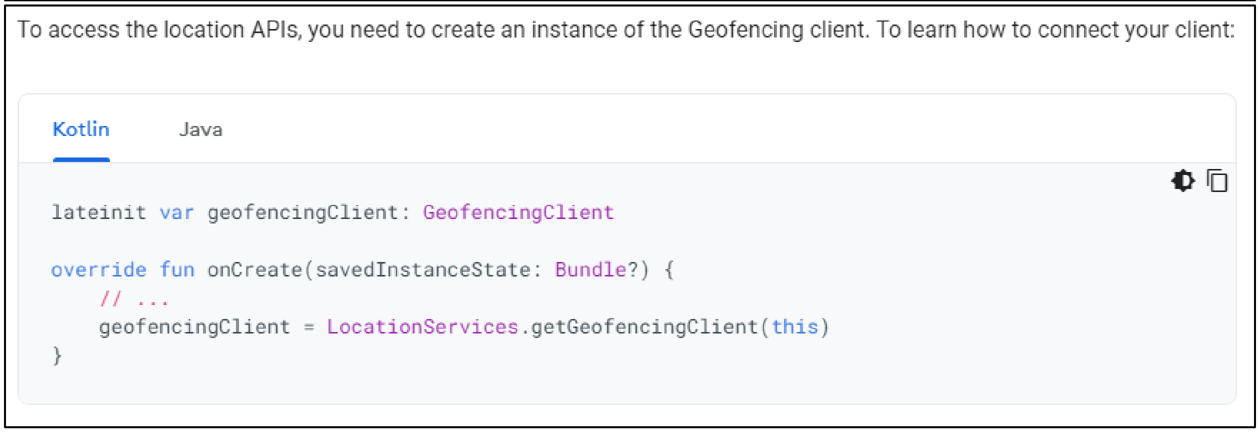
If you want to use a `BroadcastReceiver` to listen for geofence transitions, add an element specifying the service name. This element must be a child of the `<application>` element:

```
<application
   android:allowBackup="true">
   ...
   <receiver android:name=".GeofenceBroadcastReceiver"/>
<application/>
```



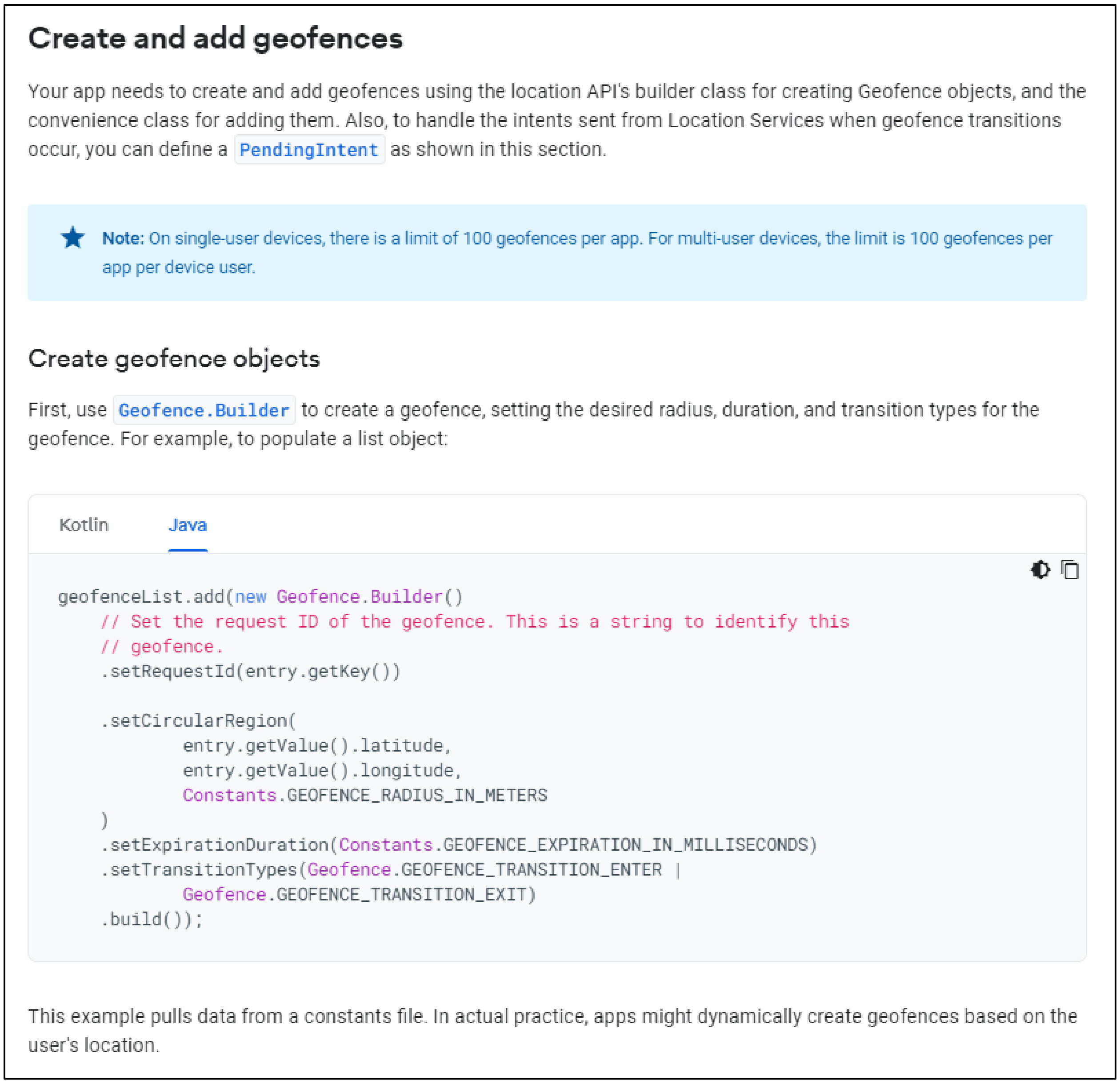
If you want to use a `BroadcastReceiver` to listen for geofence transitions, add an element specifying the service name. This element must be a child of the `<application>` element:

```
<application
   android:allowBackup="true">
   ...
   <receiver android:name=".GeofenceBroadcastReceiver"/>
<application/>
```



To access the location APIs, you need to create an instance of the Geofencing client. To learn how to connect your client:

```
lateinit var geofencingClient: GeofencingClient

override fun onCreate(savedInstanceState: Bundle?) {
    // ...
    geofencingClient = LocationServices.getGeofencingClient(this)
}
```



Create and add geofences

Your app needs to create and add geofences using the location API's builder class for creating Geofence objects, and the convenience class for adding them. Also, to handle the intents sent from Location Services when geofence transitions occur, you can define a `PendingIntent` as shown in this section.

★ **Note:** On single-user devices, there is a limit of 100 geofences per app. For multi-user devices, the limit is 100 geofences per app per device user.

Create geofence objects

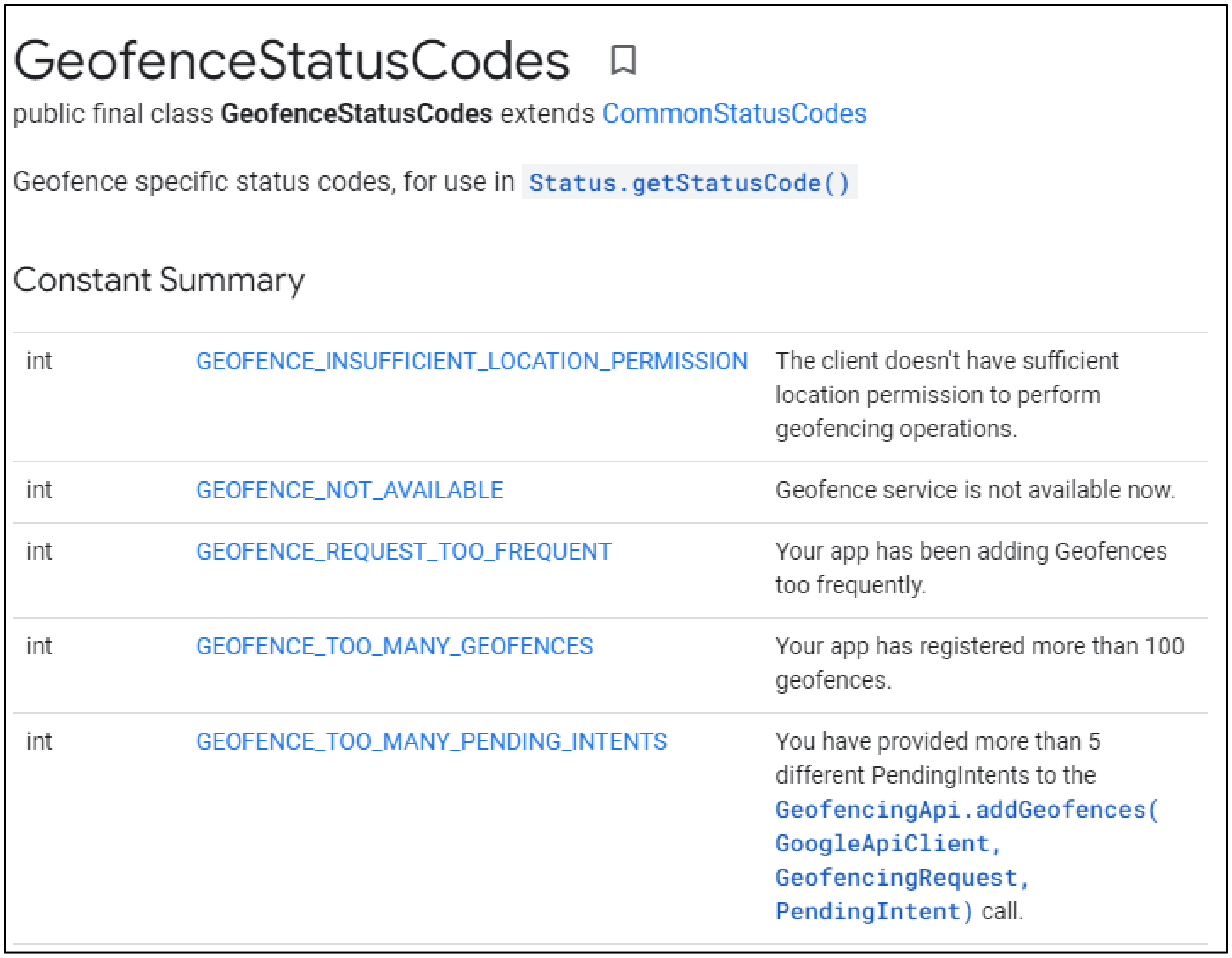
First, use `Geofence.Builder` to create a geofence, setting the desired radius, duration, and transition types for the geofence. For example, to populate a list object:

Kotlin Java

```
geofenceList.add(new Geofence.Builder()
    // Set the request ID of the geofence. This is a string to identify this
    // geofence.
    .setRequestId(entry.getKey())

    .setCircularRegion(
            entry.getValue().latitude,
            entry.getValue().longitude,
            Constants.GEOFENCE_RADIUS_IN_METERS
    )
    .setExpirationDuration(Constants.GEOFENCE_EXPIRATION_IN_MILLISECONDS)
    .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER |
            Geofence.GEOFENCE_TRANSITION_EXIT)
    .build());
```

This example pulls data from a constants file. In actual practice, apps might dynamically create geofences based on the user's location.

https://developer.android.com/training/location/geofencing


GeofenceStatusCodes

public final class **GeofenceStatusCodes** extends CommonStatusCodes

Geofence specific status codes, for use in `Status.getStatusCode()`

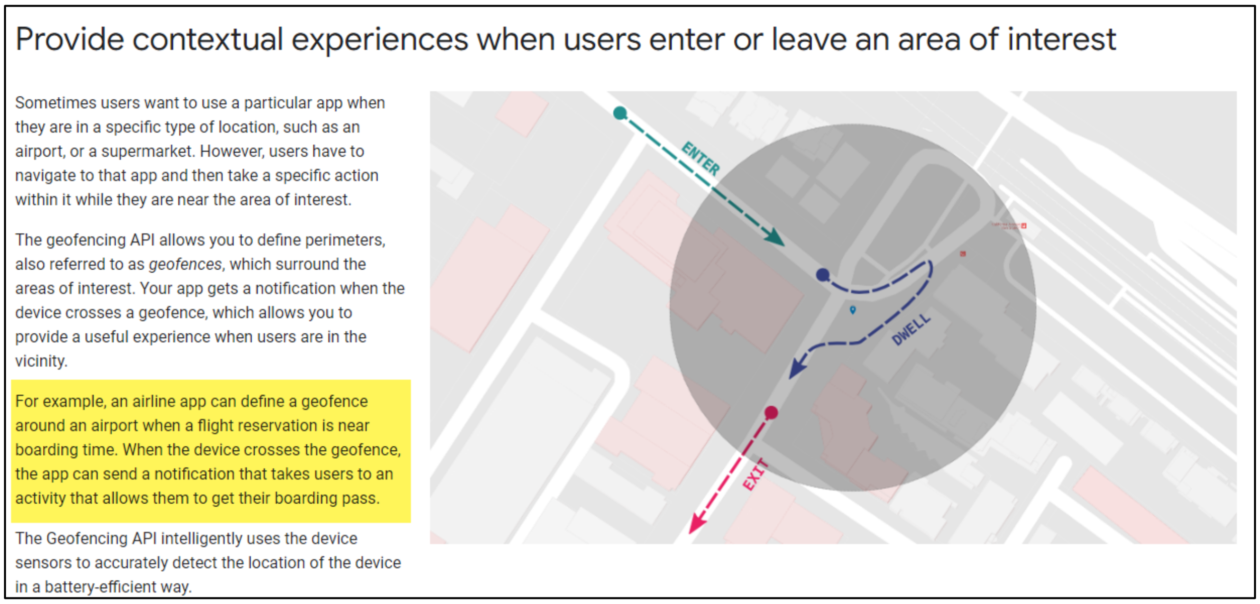
Constant Summary

int	GEOFENCE_INSUFFICIENT_LOCATION_PERMISSION	The client doesn't have sufficient location permission to perform geofencing operations.
int	GEOFENCE_NOT_AVAILABLE	Geofence service is not available now.
int	GEOFENCE_REQUEST_TOO_FREQUENT	Your app has been adding Geofences too frequently.
int	GEOFENCE_TOO_MANY_GEOFENCES	Your app has registered more than 100 geofences.
int	GEOFENCE_TOO_MANY_PENDING_INTENTS	You have provided more than 5 different PendingIntents to the `GeofencingApi.addGeofences( GoogleApiClient, GeofencingRequest, PendingIntent)` call.

https://developers.google.com/android/reference/com/google/android/gms/location/GeofenceStatusCodes

50. Samsung provides a processor configured to execute a program of instructions, the program of instructions comprising: at least one instruction for reserving content delivery to at least the particular one of the registered application programs to being from one or more sponsors after it is determined that an object of interest has entered the designated geographic area in response to determining that the interest in the designated geographic area is to be provided to at least the particular one of the registered application programs; at least one instruction for receiving, from the one or more sponsors, content to be delivered to at least one of the registered application

programs in response to said reserving; and at least one instruction for providing at least a portion of the content received from the one or more sponsors to at least the particular one of the registered application programs after it is determined that the object of interest has entered the designated geographic area, including as demonstrated in the exemplary text and images below:



Provide contextual experiences when users enter or leave an area of interest

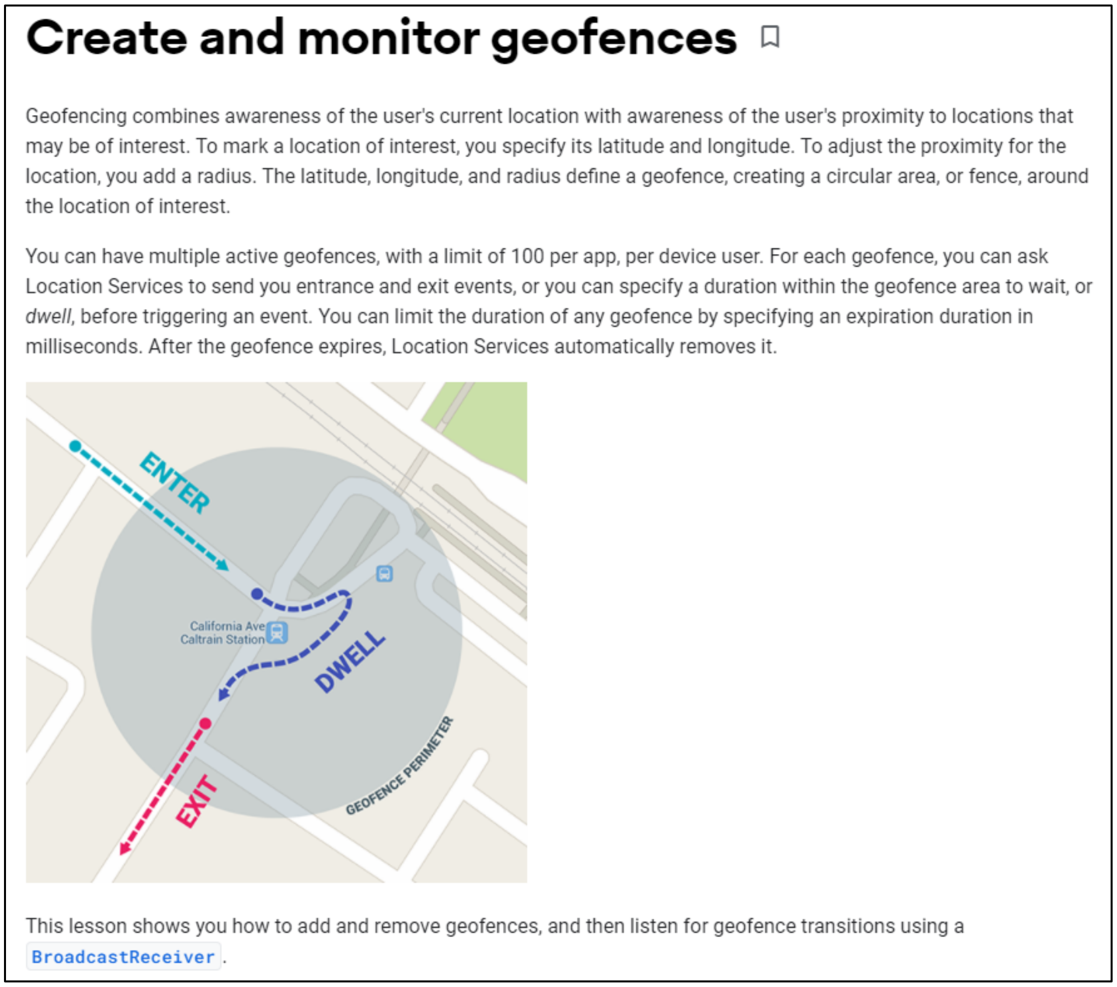
Sometimes users want to use a particular app when they are in a specific type of location, such as an airport, or a supermarket. However, users have to navigate to that app and then take a specific action within it while they are near the area of interest.

The geofencing API allows you to define perimeters, also referred to as *geofences*, which surround the areas of interest. Your app gets a notification when the device crosses a geofence, which allows you to provide a useful experience when users are in the vicinity.

For example, an airline app can define a geofence around an airport when a flight reservation is near boarding time. When the device crosses the geofence, the app can send a notification that takes users to an activity that allows them to get their boarding pass.

The Geofencing API intelligently uses the device sensors to accurately detect the location of the device in a battery-efficient way.

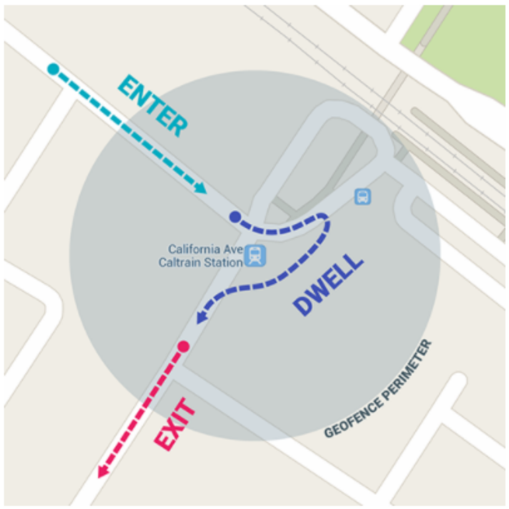
https://developers.google.com/location-context/geofencing#:~:text=The%20geofencing%20API%20allows%20you,users%20are%20in%20the%20vicinity



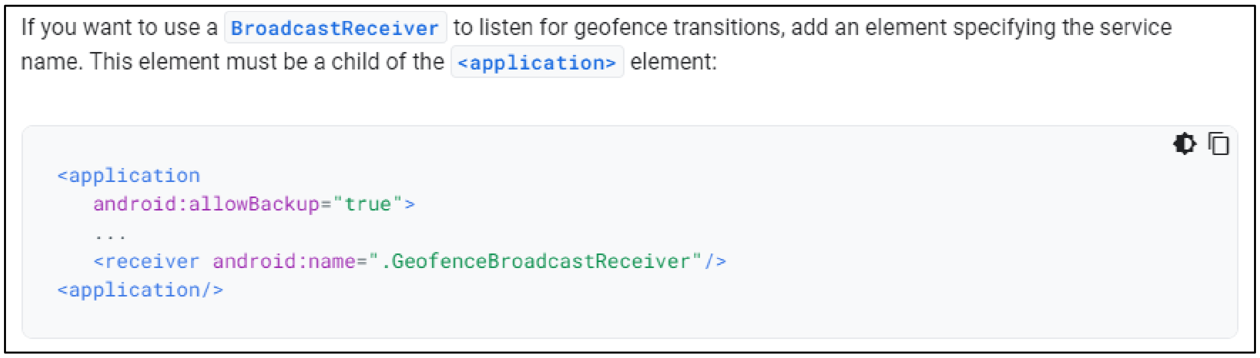
Create and monitor geofences

Geofencing combines awareness of the user's current location with awareness of the user's proximity to locations that may be of interest. To mark a location of interest, you specify its latitude and longitude. To adjust the proximity for the location, you add a radius. The latitude, longitude, and radius define a geofence, creating a circular area, or fence, around the location of interest.

You can have multiple active geofences, with a limit of 100 per app, per device user. For each geofence, you can ask Location Services to send you entrance and exit events, or you can specify a duration within the geofence area to wait, or *dwell*, before triggering an event. You can limit the duration of any geofence by specifying an expiration duration in milliseconds. After the geofence expires, Location Services automatically removes it.

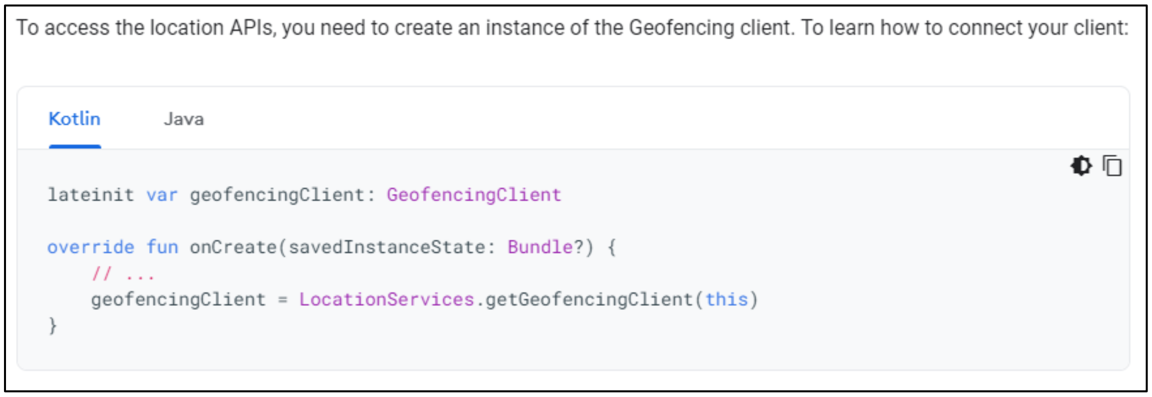


This lesson shows you how to add and remove geofences, and then listen for geofence transitions using a `BroadcastReceiver`.



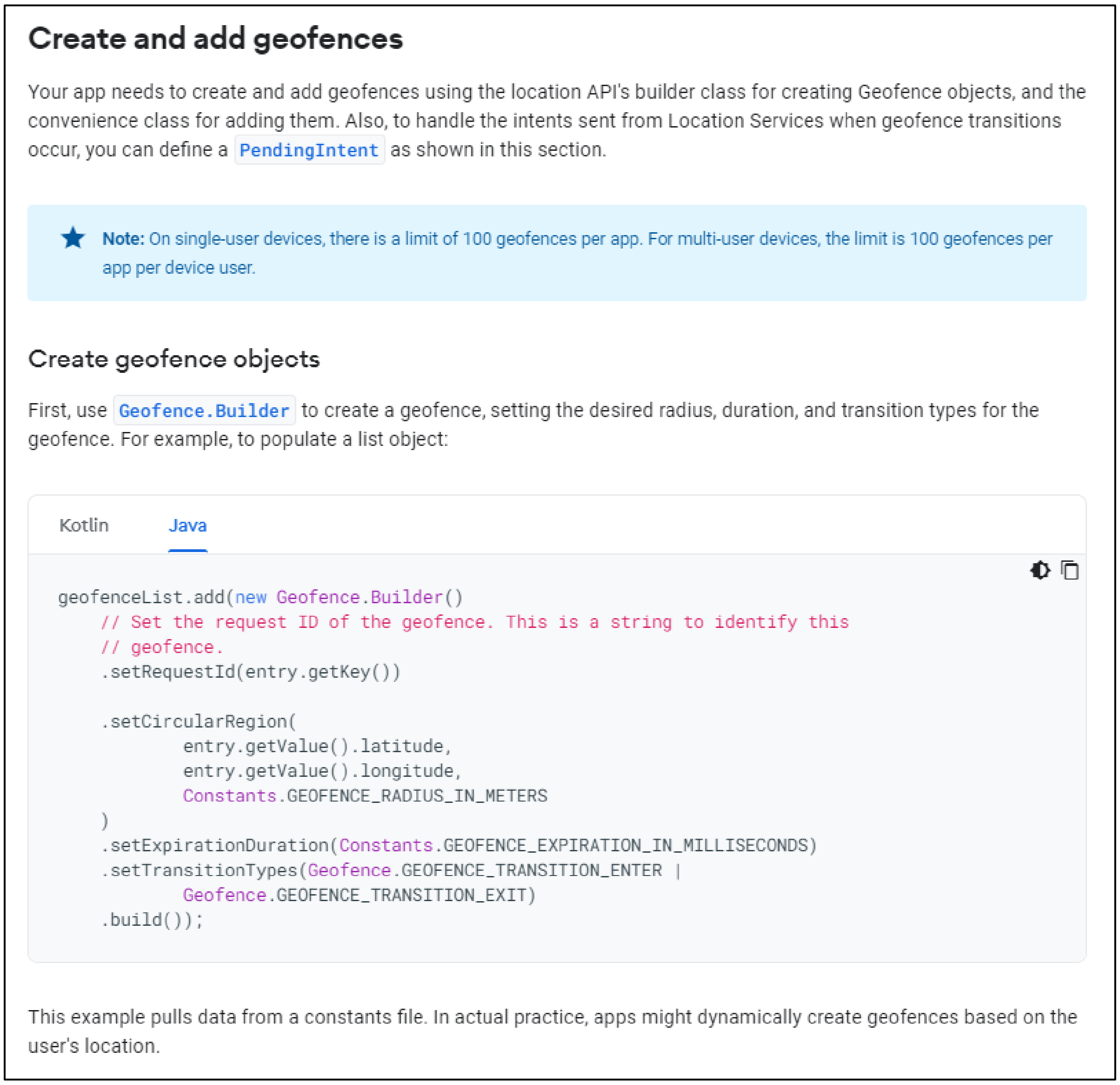
If you want to use a `BroadcastReceiver` to listen for geofence transitions, add an element specifying the service name. This element must be a child of the `<application>` element:

```
<application
   android:allowBackup="true">
   ...
   <receiver android:name=".GeofenceBroadcastReceiver"/>
<application/>
```



To access the location APIs, you need to create an instance of the Geofencing client. To learn how to connect your client:

```
lateinit var geofencingClient: GeofencingClient

override fun onCreate(savedInstanceState: Bundle?) {
    // ...
    geofencingClient = LocationServices.getGeofencingClient(this)
}
```



Create and add geofences

Your app needs to create and add geofences using the location API's builder class for creating Geofence objects, and the convenience class for adding them. Also, to handle the intents sent from Location Services when geofence transitions occur, you can define a `PendingIntent` as shown in this section.

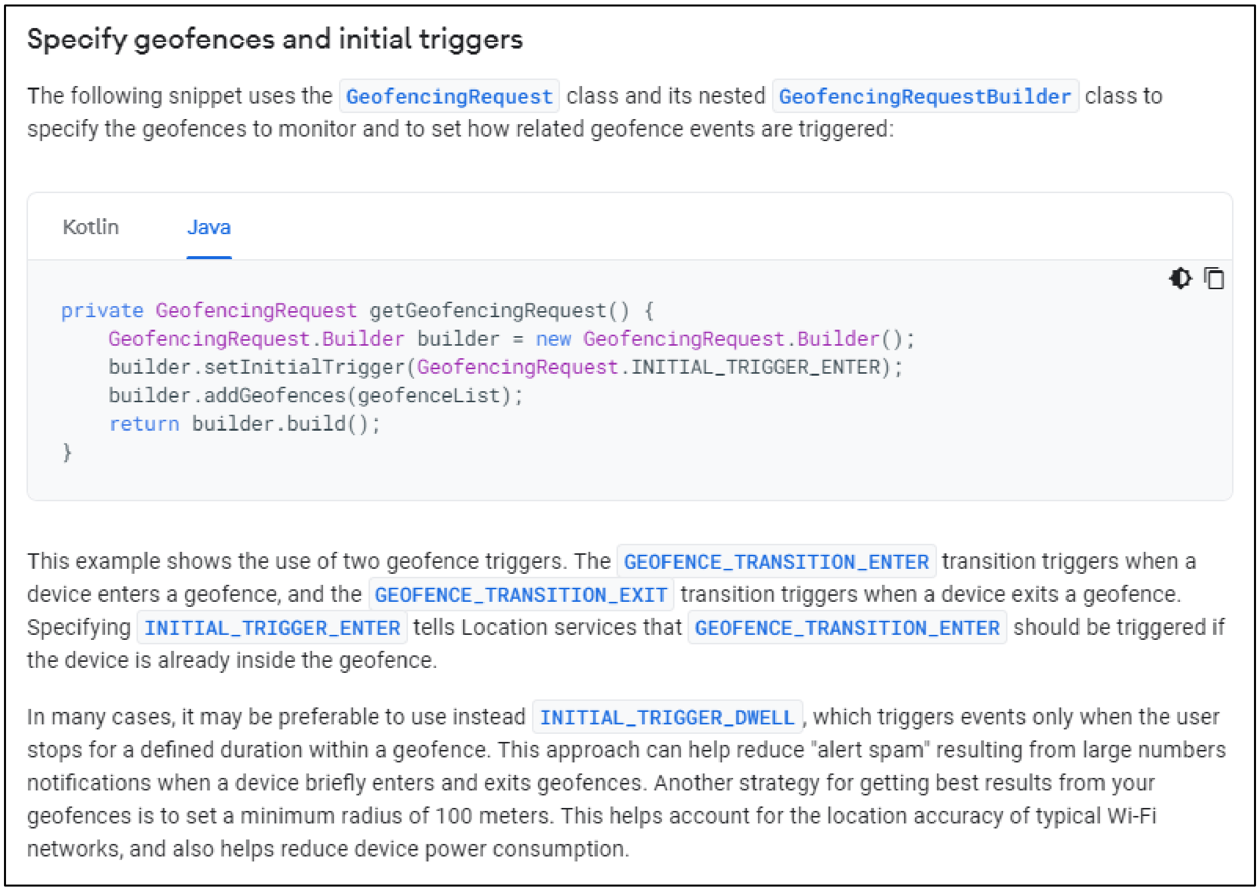
Note: On single-user devices, there is a limit of 100 geofences per app. For multi-user devices, the limit is 100 geofences per app per device user.

Create geofence objects

First, use `Geofence.Builder` to create a geofence, setting the desired radius, duration, and transition types for the geofence. For example, to populate a list object:

```
geofenceList.add(new Geofence.Builder()
    // Set the request ID of the geofence. This is a string to identify this
    // geofence.
    .setRequestId(entry.getKey())

    .setCircularRegion(
            entry.getValue().latitude,
            entry.getValue().longitude,
            Constants.GEOFENCE_RADIUS_IN_METERS
    )
    .setExpirationDuration(Constants.GEOFENCE_EXPIRATION_IN_MILLISECONDS)
    .setTransitionTypes(Geofence.GEOFENCE_TRANSITION_ENTER |
            Geofence.GEOFENCE_TRANSITION_EXIT)
    .build());
```

This example pulls data from a constants file. In actual practice, apps might dynamically create geofences based on the user's location.



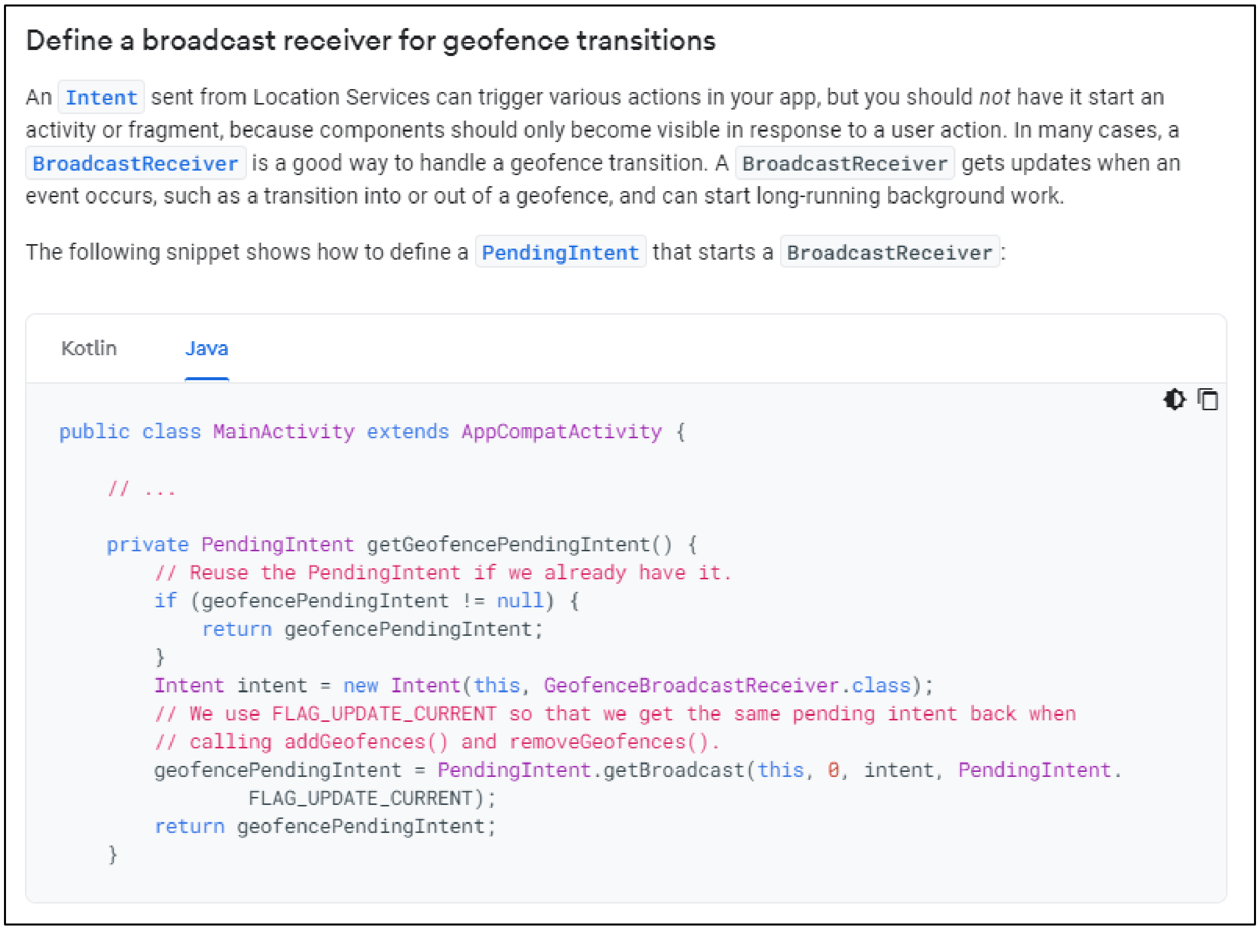
Specify geofences and initial triggers

The following snippet uses the `GeofencingRequest` class and its nested `GeofencingRequestBuilder` class to specify the geofences to monitor and to set how related geofence events are triggered:

```
private GeofencingRequest getGeofencingRequest() {
    GeofencingRequest.Builder builder = new GeofencingRequest.Builder();
    builder.setInitialTrigger(GeofencingRequest.INITIAL_TRIGGER_ENTER);
    builder.addGeofences(geofenceList);
    return builder.build();
}
```

This example shows the use of two geofence triggers. The `GEOFENCE_TRANSITION_ENTER` transition triggers when a device enters a geofence, and the `GEOFENCE_TRANSITION_EXIT` transition triggers when a device exits a geofence. Specifying `INITIAL_TRIGGER_ENTER` tells Location services that `GEOFENCE_TRANSITION_ENTER` should be triggered if the device is already inside the geofence.

In many cases, it may be preferable to use instead `INITIAL_TRIGGER_DWELL`, which triggers events only when the user stops for a defined duration within a geofence. This approach can help reduce "alert spam" resulting from large numbers notifications when a device briefly enters and exits geofences. Another strategy for getting best results from your geofences is to set a minimum radius of 100 meters. This helps account for the location accuracy of typical Wi-Fi networks, and also helps reduce device power consumption.

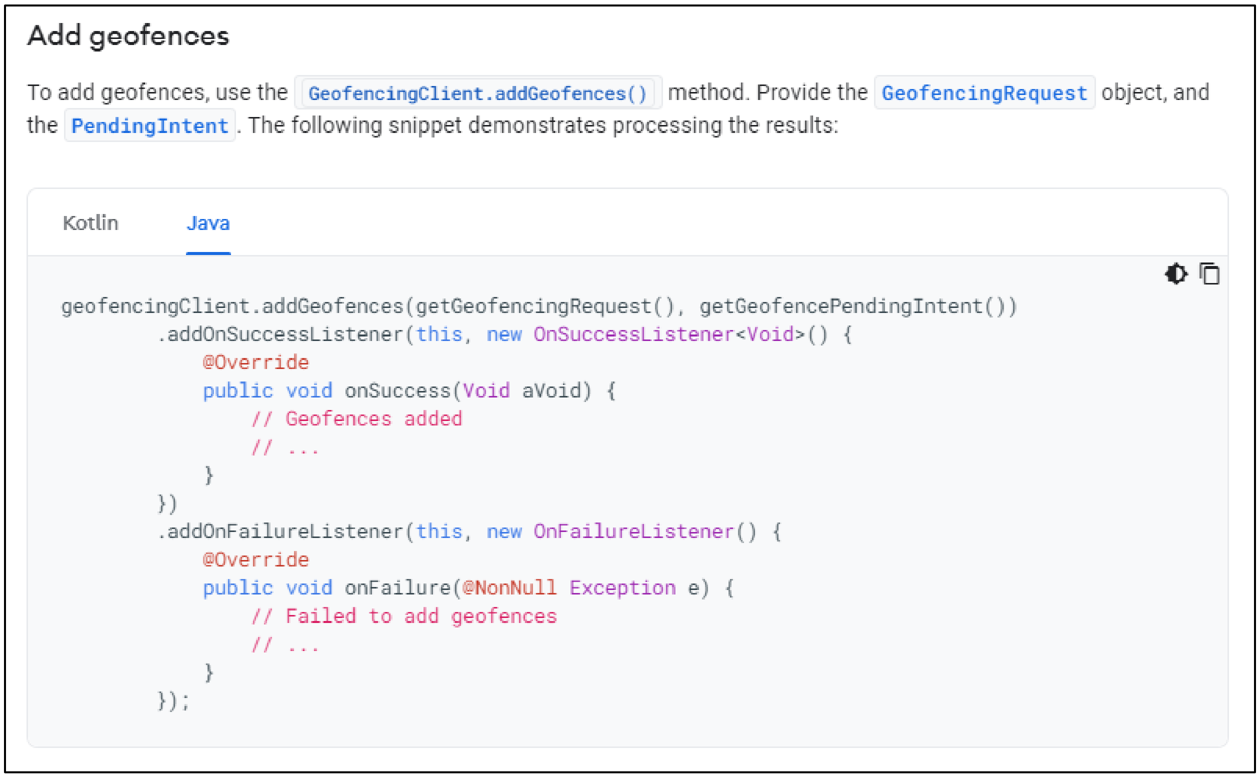


Define a broadcast receiver for geofence transitions

An `Intent` sent from Location Services can trigger various actions in your app, but you should *not* have it start an activity or fragment, because components should only become visible in response to a user action. In many cases, a `BroadcastReceiver` is a good way to handle a geofence transition. A `BroadcastReceiver` gets updates when an event occurs, such as a transition into or out of a geofence, and can start long-running background work.

The following snippet shows how to define a `PendingIntent` that starts a `BroadcastReceiver`:

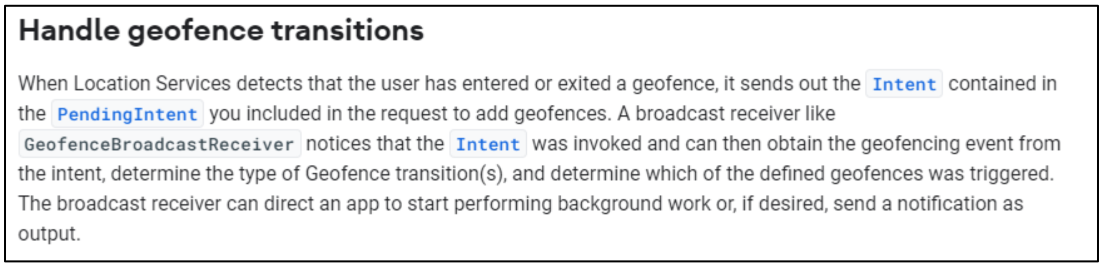
```java
public class MainActivity extends AppCompatActivity {

    // ...

    private PendingIntent getGeofencePendingIntent() {
        // Reuse the PendingIntent if we already have it.
        if (geofencePendingIntent != null) {
            return geofencePendingIntent;
        }
        Intent intent = new Intent(this, GeofenceBroadcastReceiver.class);
        // We use FLAG_UPDATE_CURRENT so that we get the same pending intent back when
        // calling addGeofences() and removeGeofences().
        geofencePendingIntent = PendingIntent.getBroadcast(this, 0, intent, PendingIntent.
                FLAG_UPDATE_CURRENT);
        return geofencePendingIntent;
    }
```



Add geofences

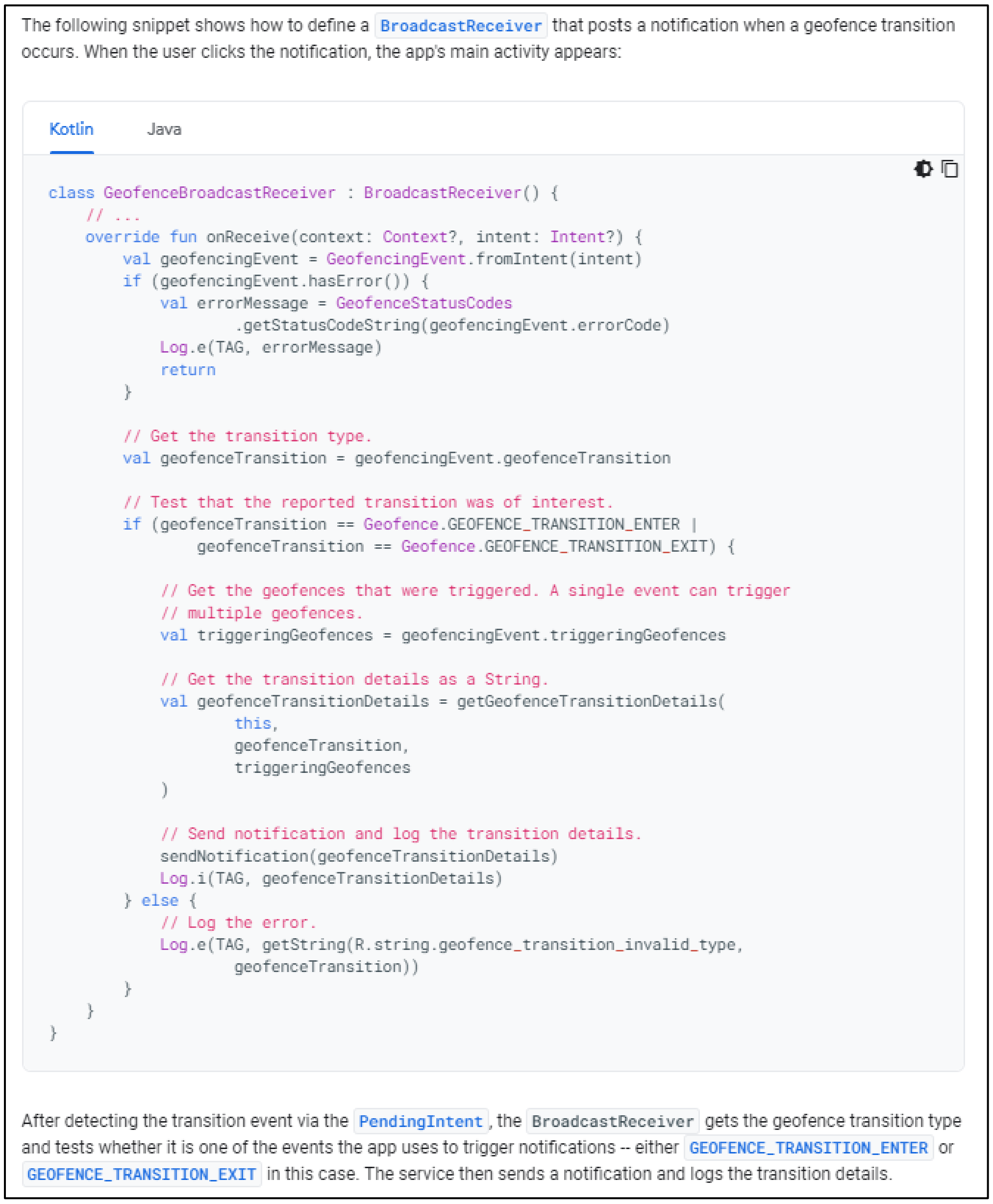
To add geofences, use the `GeofencingClient.addGeofences()` method. Provide the `GeofencingRequest` object, and the `PendingIntent`. The following snippet demonstrates processing the results:

```
geofencingClient.addGeofences(getGeofencingRequest(), getGeofencePendingIntent())
        .addOnSuccessListener(this, new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                // Geofences added
                // ...
            }
        })
        .addOnFailureListener(this, new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                // Failed to add geofences
                // ...
            }
        });
```



Handle geofence transitions

When Location Services detects that the user has entered or exited a geofence, it sends out the `Intent` contained in the `PendingIntent` you included in the request to add geofences. A broadcast receiver like `GeofenceBroadcastReceiver` notices that the `Intent` was invoked and can then obtain the geofencing event from the intent, determine the type of Geofence transition(s), and determine which of the defined geofences was triggered. The broadcast receiver can direct an app to start performing background work or, if desired, send a notification as output.



The following snippet shows how to define a BroadcastReceiver that posts a notification when a geofence transition occurs. When the user clicks the notification, the app's main activity appears:

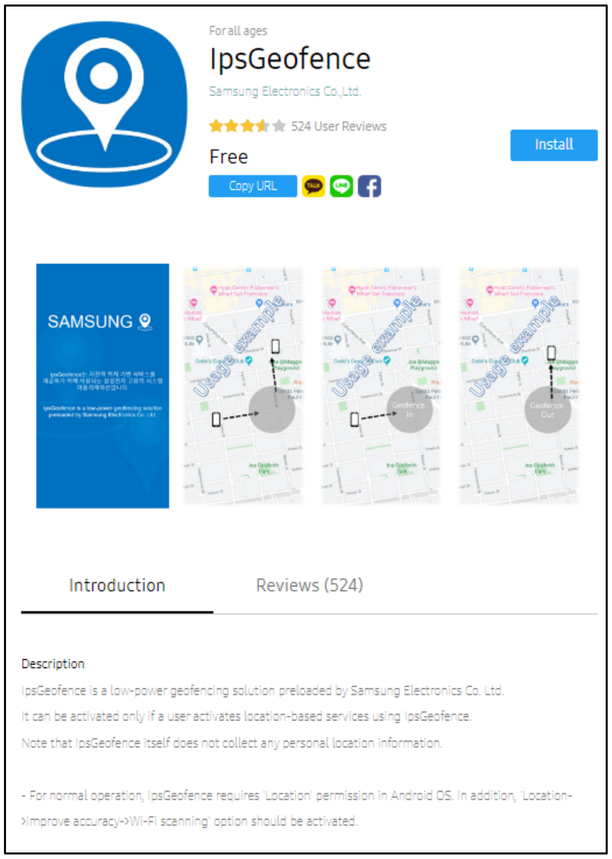
```
class GeofenceBroadcastReceiver : BroadcastReceiver() {
    // ...
    override fun onReceive(context: Context?, intent: Intent?) {
        val geofencingEvent = GeofencingEvent.fromIntent(intent)
        if (geofencingEvent.hasError()) {
            val errorMessage = GeofenceStatusCodes
                    .getStatusCodeString(geofencingEvent.errorCode)
            Log.e(TAG, errorMessage)
            return
        }

        // Get the transition type.
        val geofenceTransition = geofencingEvent.geofenceTransition

        // Test that the reported transition was of interest.
        if (geofenceTransition == Geofence.GEOFENCE_TRANSITION_ENTER |
                geofenceTransition == Geofence.GEOFENCE_TRANSITION_EXIT) {

            // Get the geofences that were triggered. A single event can trigger
            // multiple geofences.
            val triggeringGeofences = geofencingEvent.triggeringGeofences

            // Get the transition details as a String.
            val geofenceTransitionDetails = getGeofenceTransitionDetails(
                    this,
                    geofenceTransition,
                    triggeringGeofences
            )

            // Send notification and log the transition details.
            sendNotification(geofenceTransitionDetails)
            Log.i(TAG, geofenceTransitionDetails)
        } else {
            // Log the error.
            Log.e(TAG, getString(R.string.geofence_transition_invalid_type,
                    geofenceTransition))
        }
    }
}
```

After detecting the transition event via the PendingIntent, the BroadcastReceiver gets the geofence transition type and tests whether it is one of the events the app uses to trigger notifications -- either GEOFENCE_TRANSITION_ENTER or GEOFENCE_TRANSITION_EXIT in this case. The service then sends a notification and logs the transition details.

https://developer.android.com/training/location/geofencing

51. On information and belief, Samsung developed applications such as Smart Things, which uses Samsung's geofencing solution called IPSGeofence that appears to function similarly to the geofencing solution in Android described above. In doing so, Samsung infringes the '247 patent in an alternative way by providing a system comprising: a memory; a communications

interface operable coupled to the memory; and at least one processor operably coupled to the memory and the communications interface, the processor configured to execute a program of instructions, the program of instructions comprising: at least one instruction for registering a plurality of application programs for use with a content delivery platform; at least one instruction for establishing a plurality of perimeters defining a plurality of geographic areas; at least one instruction for receiving, from at least a particular one of the registered application programs, request to obtain an interest in a designated geographic area; at least one instruction for determining if the interest in the designated geographic area is to be provided to at least the particular one of the registered application programs; at least one instruction for reserving content delivery to at least the particular one of the registered application programs to being from one or more sponsors after it is determined that an object of interest has entered the designated geographic area in response to determining that the interest in the designated geographic area is to be provided to at least the particular one of the registered application programs; at least one instruction for receiving, from the one or more sponsors, content to be delivered to at least one of the registered application programs in response to said reserving; and at least one instruction for providing at least a portion of the content received from the one or more sponsors to at least the particular one of the registered application programs after it is determined that the object of interest has entered the designated geographic area. This is demonstrated in the exemplary text and images below:



IpsGeofence

Samsung Electronics Co.,Ltd.

524 User Reviews

Free

Introduction

Reviews (524)

Description

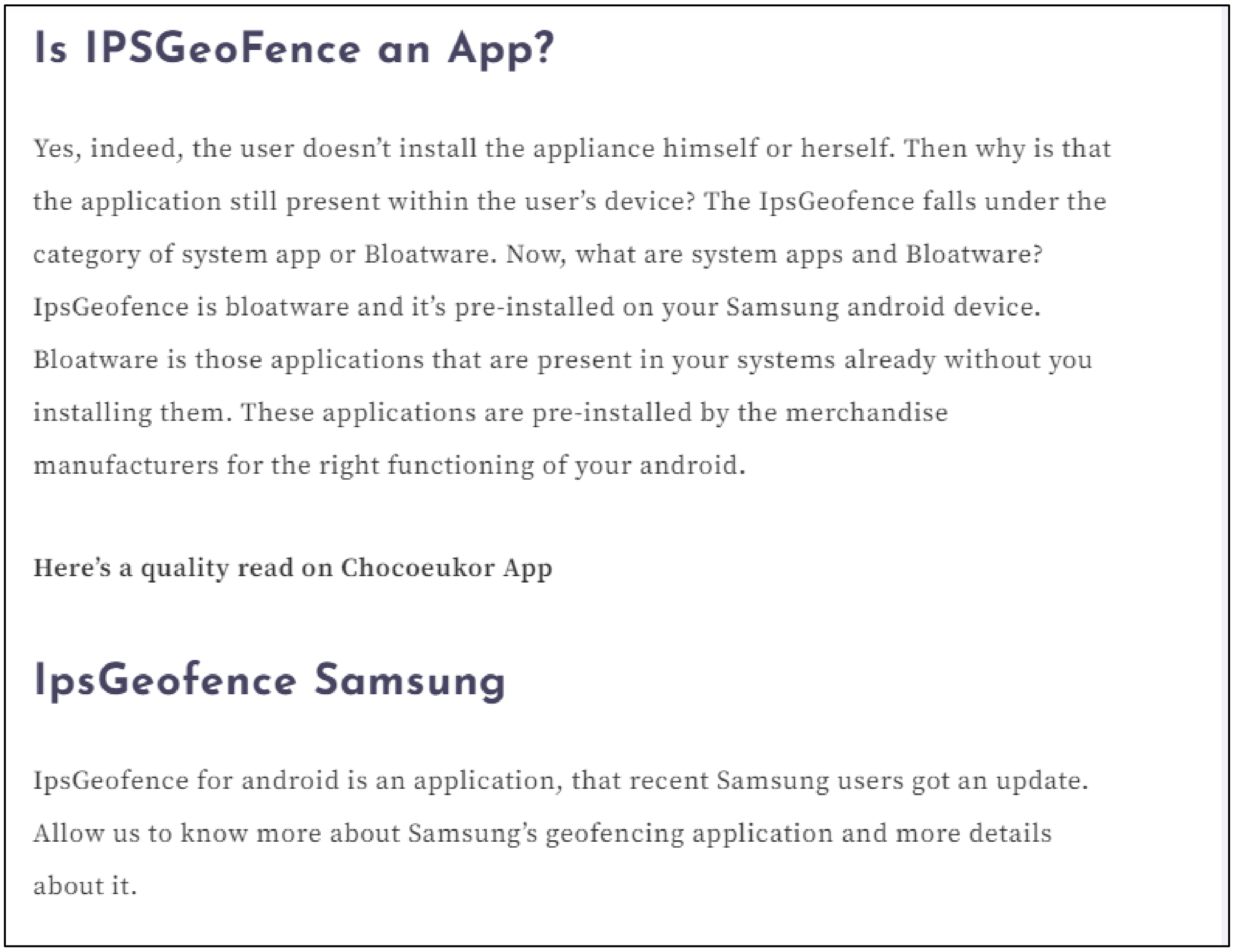
IpsGeofence is a low-power geofencing solution preloaded by Samsung Electronics Co. Ltd.

It can be activated only if a user activates location-based services using IpsGeofence.

Note that IpsGeofence itself does not collect any personal location information.

- For normal operation, IpsGeofence requires 'Location' permission in Android OS. In addition, 'Location->Improve accuracy->Wi-Fi scanning' option should be activated.

https://galaxystore.samsung.com/prepost/000004701101?appId=com.samsung.android.ipsgeofence



Is IPSGeoFence an App?

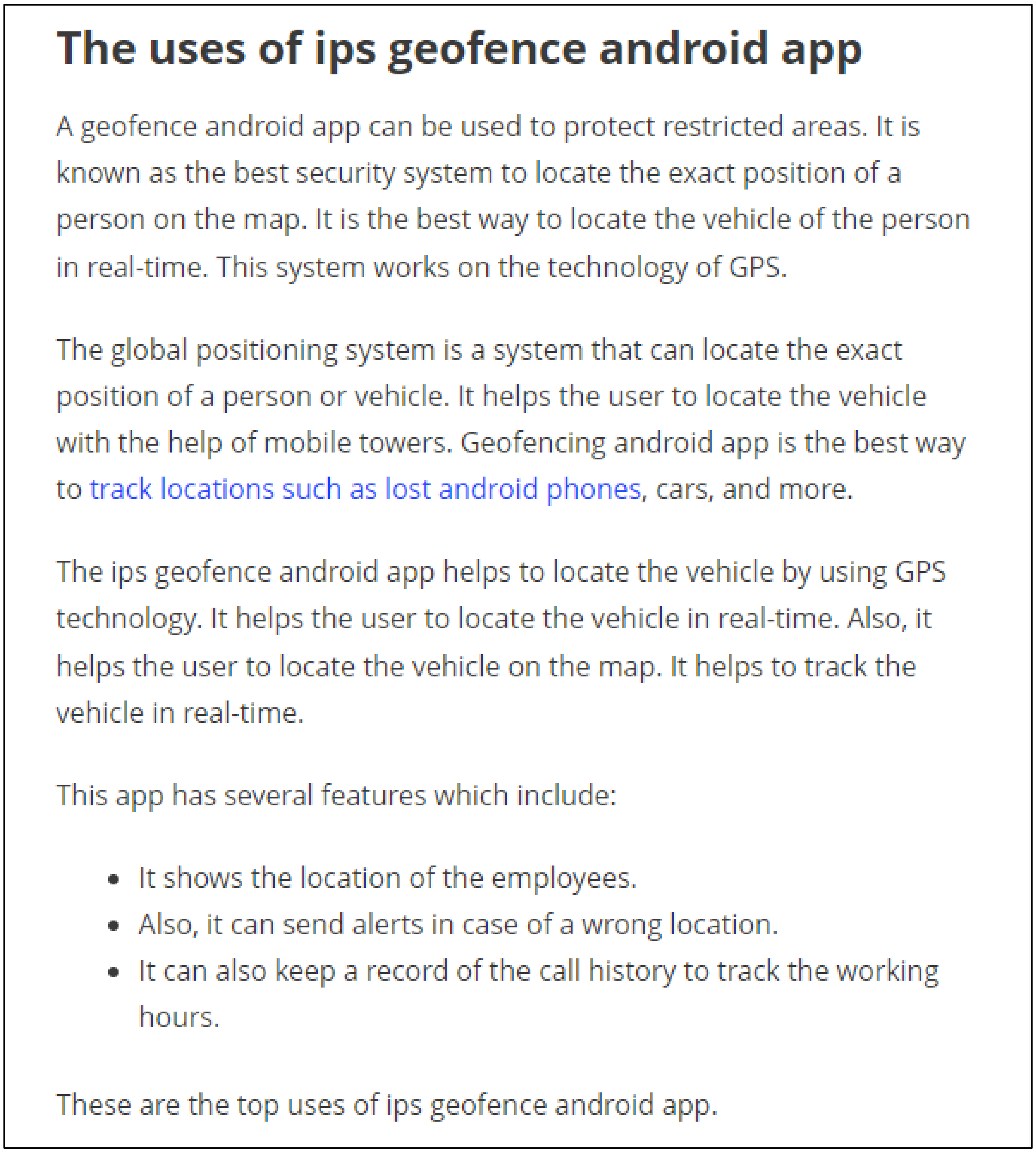
Yes, indeed, the user doesn't install the appliance himself or herself. Then why is that the application still present within the user's device? The IpsGeofence falls under the category of system app or Bloatware. Now, what are system apps and Bloatware? IpsGeofence is bloatware and it's pre-installed on your Samsung android device. Bloatware is those applications that are present in your systems already without you installing them. These applications are pre-installed by the merchandise manufacturers for the right functioning of your android.

Here's a quality read on Chocoeukor App

IpsGeofence Samsung

IpsGeofence for android is an application, that recent Samsung users got an update. Allow us to know more about Samsung's geofencing application and more details about it.

https://hackanons.com/2021/06/what-is-ipsgeofence-app.html



The uses of ips geofence android app

A geofence android app can be used to protect restricted areas. It is known as the best security system to locate the exact position of a person on the map. It is the best way to locate the vehicle of the person in real-time. This system works on the technology of GPS.

The global positioning system is a system that can locate the exact position of a person or vehicle. It helps the user to locate the vehicle with the help of mobile towers. Geofencing android app is the best way to track locations such as lost android phones, cars, and more.

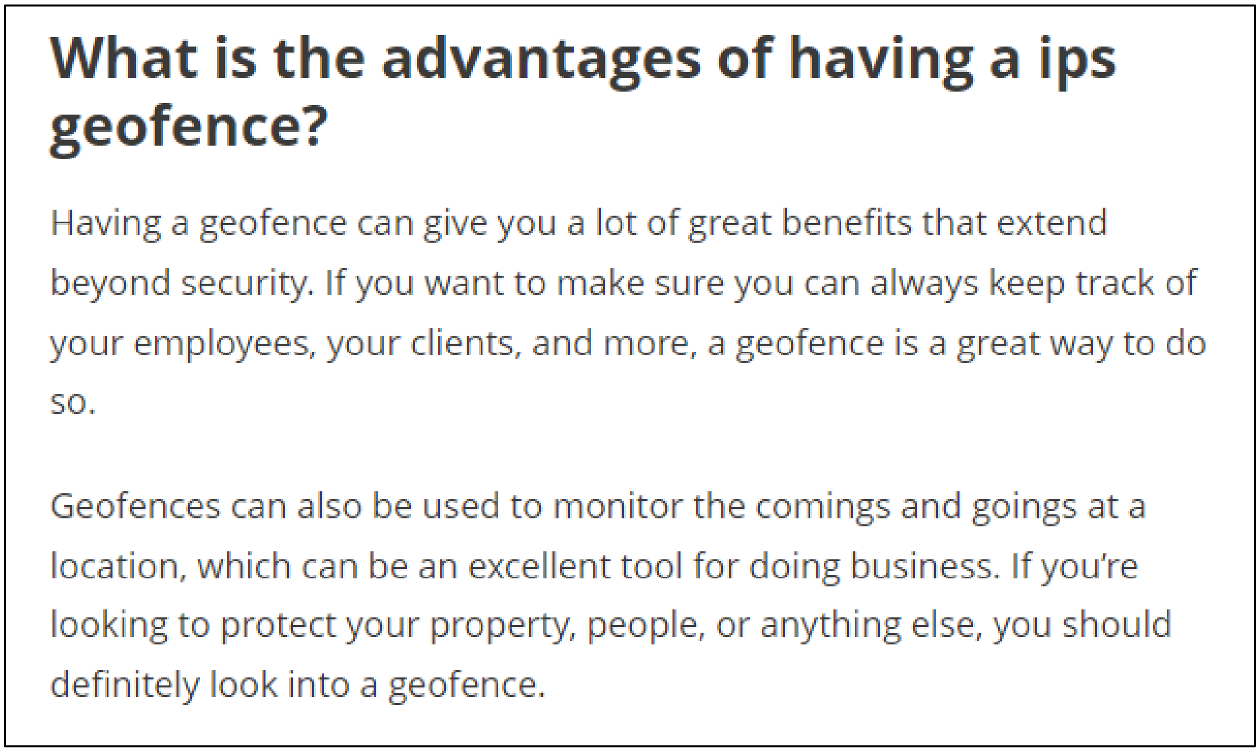
The ips geofence android app helps to locate the vehicle by using GPS technology. It helps the user to locate the vehicle in real-time. Also, it helps the user to locate the vehicle on the map. It helps to track the vehicle in real-time.

This app has several features which include:

- It shows the location of the employees.
- Also, it can send alerts in case of a wrong location.
- It can also keep a record of the call history to track the working hours.

These are the top uses of ips geofence android app.

https://grouphowto.com/ipsgeofence-app/

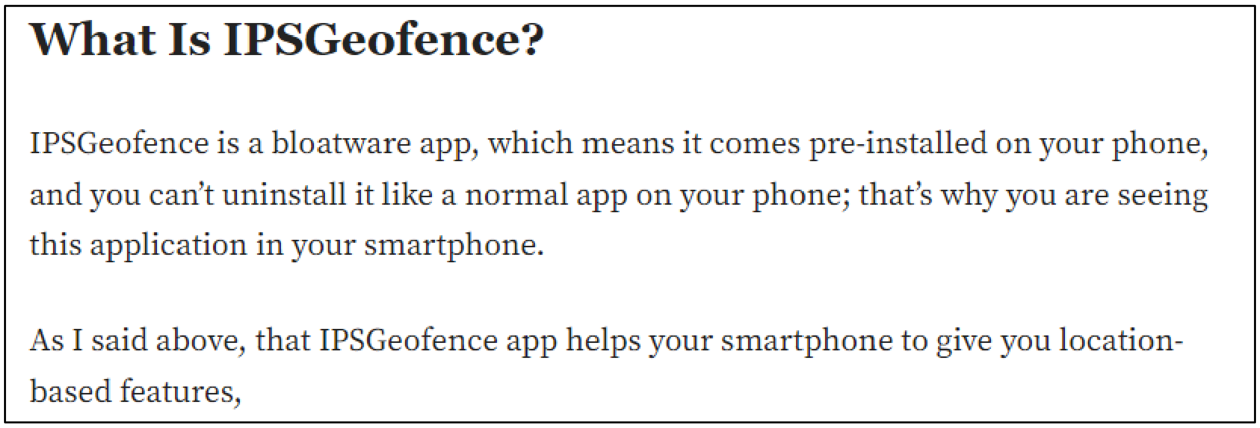


What is the advantages of having a ips geofence?

Having a geofence can give you a lot of great benefits that extend beyond security. If you want to make sure you can always keep track of your employees, your clients, and more, a geofence is a great way to do so.

Geofences can also be used to monitor the comings and goings at a location, which can be an excellent tool for doing business. If you're looking to protect your property, people, or anything else, you should definitely look into a geofence.

https://grouphowto.com/ipsgeofence-app/

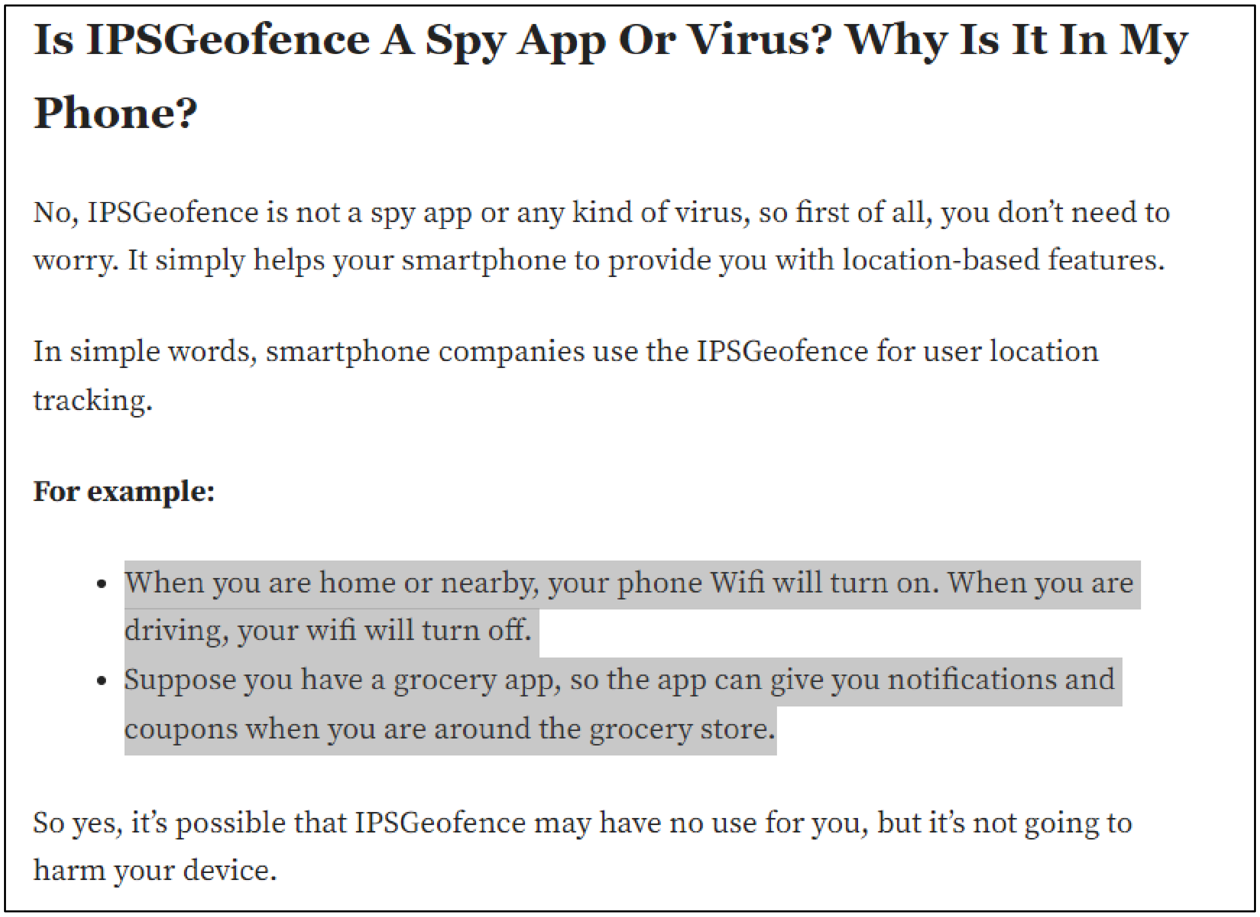


What Is IPSGeofence?

IPSGeofence is a bloatware app, which means it comes pre-installed on your phone, and you can't uninstall it like a normal app on your phone; that's why you are seeing this application in your smartphone.

As I said above, that IPSGeofence app helps your smartphone to give you location-based features,

https://quaries.com/what-is-ipsgeofence-and-how-to-uninstall-it/



Is IPSGeofence A Spy App Or Virus? Why Is It In My Phone?

No, IPSGeofence is not a spy app or any kind of virus, so first of all, you don't need to worry. It simply helps your smartphone to provide you with location-based features.

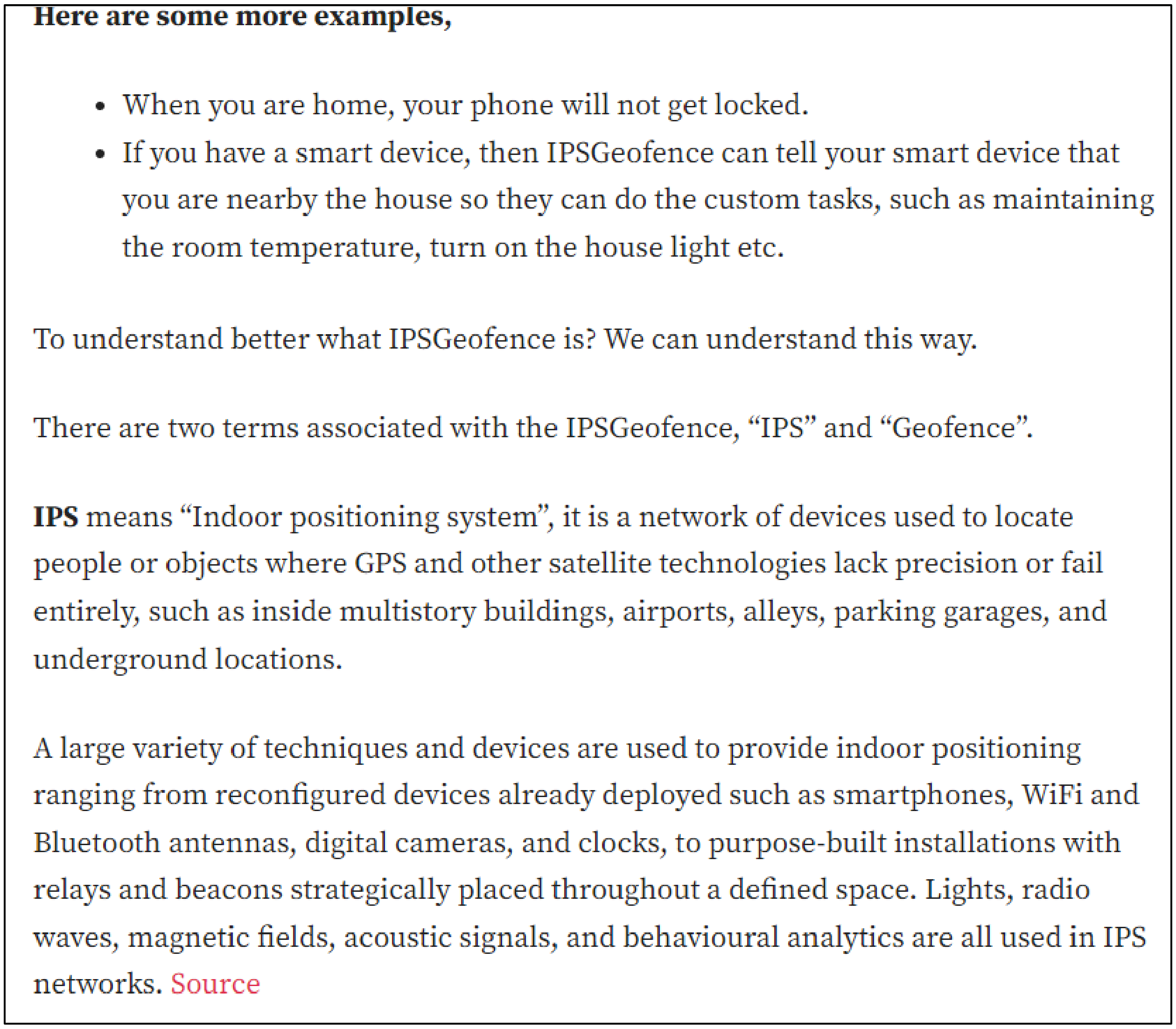
In simple words, smartphone companies use the IPSGeofence for user location tracking.

For example:

- When you are home or nearby, your phone Wifi will turn on. When you are driving, your wifi will turn off.
- Suppose you have a grocery app, so the app can give you notifications and coupons when you are around the grocery store.

So yes, it's possible that IPSGeofence may have no use for you, but it's not going to harm your device.

Id.



Here are some more examples,

- When you are home, your phone will not get locked.
- If you have a smart device, then IPSGeofence can tell your smart device that you are nearby the house so they can do the custom tasks, such as maintaining the room temperature, turn on the house light etc.

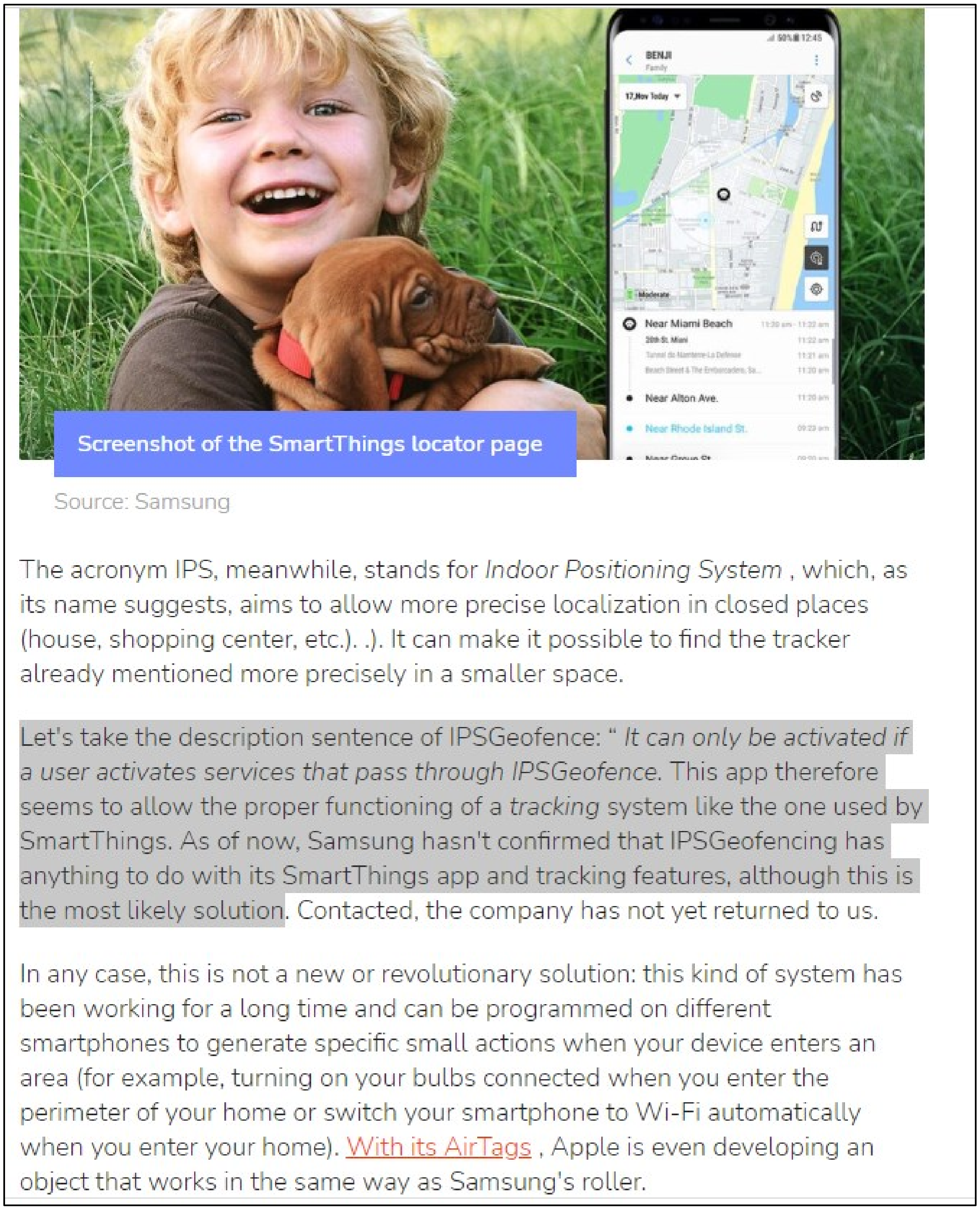
To understand better what IPSGeofence is? We can understand this way.

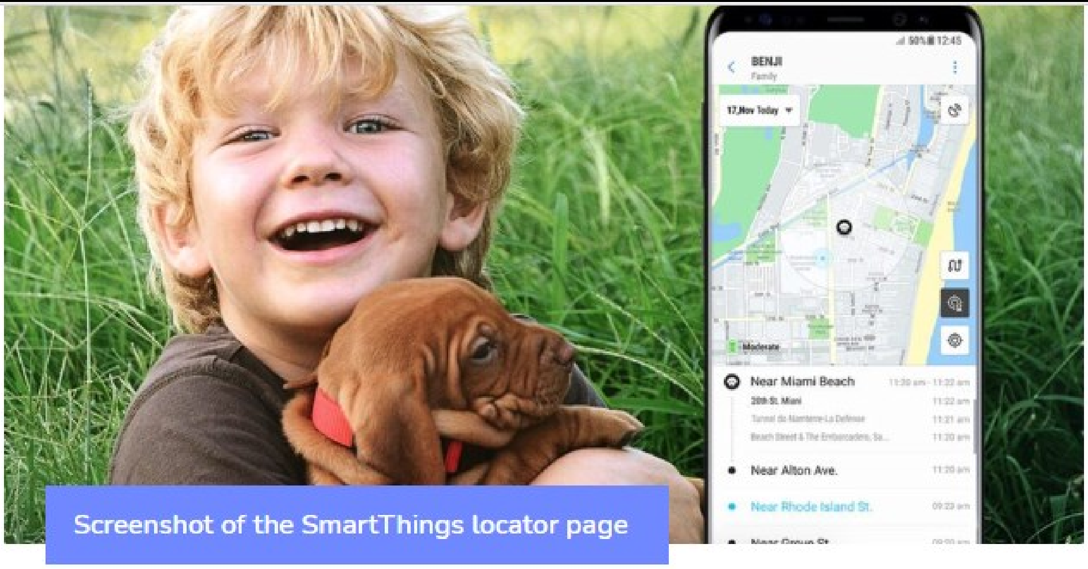
There are two terms associated with the IPSGeofence, "IPS" and "Geofence".

IPS means "Indoor positioning system", it is a network of devices used to locate people or objects where GPS and other satellite technologies lack precision or fail entirely, such as inside multistory buildings, airports, alleys, parking garages, and underground locations.

A large variety of techniques and devices are used to provide indoor positioning ranging from reconfigured devices already deployed such as smartphones, WiFi and Bluetooth antennas, digital cameras, and clocks, to purpose-built installations with relays and beacons strategically placed throughout a defined space. Lights, radio waves, magnetic fields, acoustic signals, and behavioural analytics are all used in IPS networks. Source

Id.





Screenshot of the SmartThings locator page

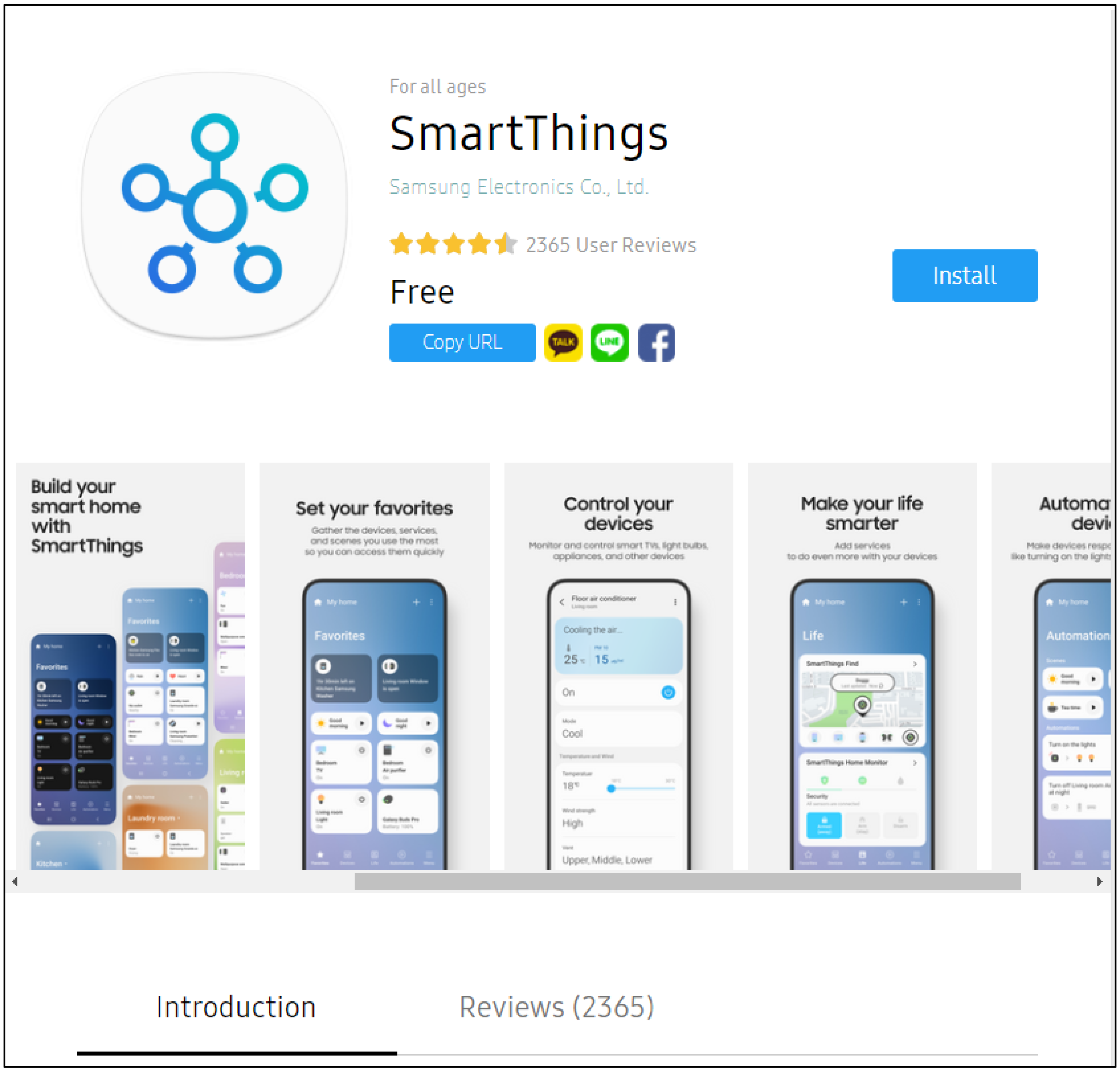
Source: Samsung

The acronym IPS, meanwhile, stands for *Indoor Positioning System* , which, as its name suggests, aims to allow more precise localization in closed places (house, shopping center, etc.). .). It can make it possible to find the tracker already mentioned more precisely in a smaller space.

Let's take the description sentence of IPSGeofence: " *It can only be activated if a user activates services that pass through IPSGeofence*. This app therefore seems to allow the proper functioning of a *tracking* system like the one used by SmartThings. As of now, Samsung hasn't confirmed that IPSGeofencing has anything to do with its SmartThings app and tracking features, although this is the most likely solution. Contacted, the company has not yet returned to us.

In any case, this is not a new or revolutionary solution: this kind of system has been working for a long time and can be programmed on different smartphones to generate specific small actions when your device enters an area (for example, turning on your bulbs connected when you enter the perimeter of your home or switch your smartphone to Wi-Fi automatically when you enter your home). With its AirTags , Apple is even developing an object that works in the same way as Samsung's roller.

https://www.numerama.com/tech/619559-ipsgeofence-a-quoi-sert-lapp-samsung-quil-est-impossible-de-desactiver.html



For all ages

SmartThings

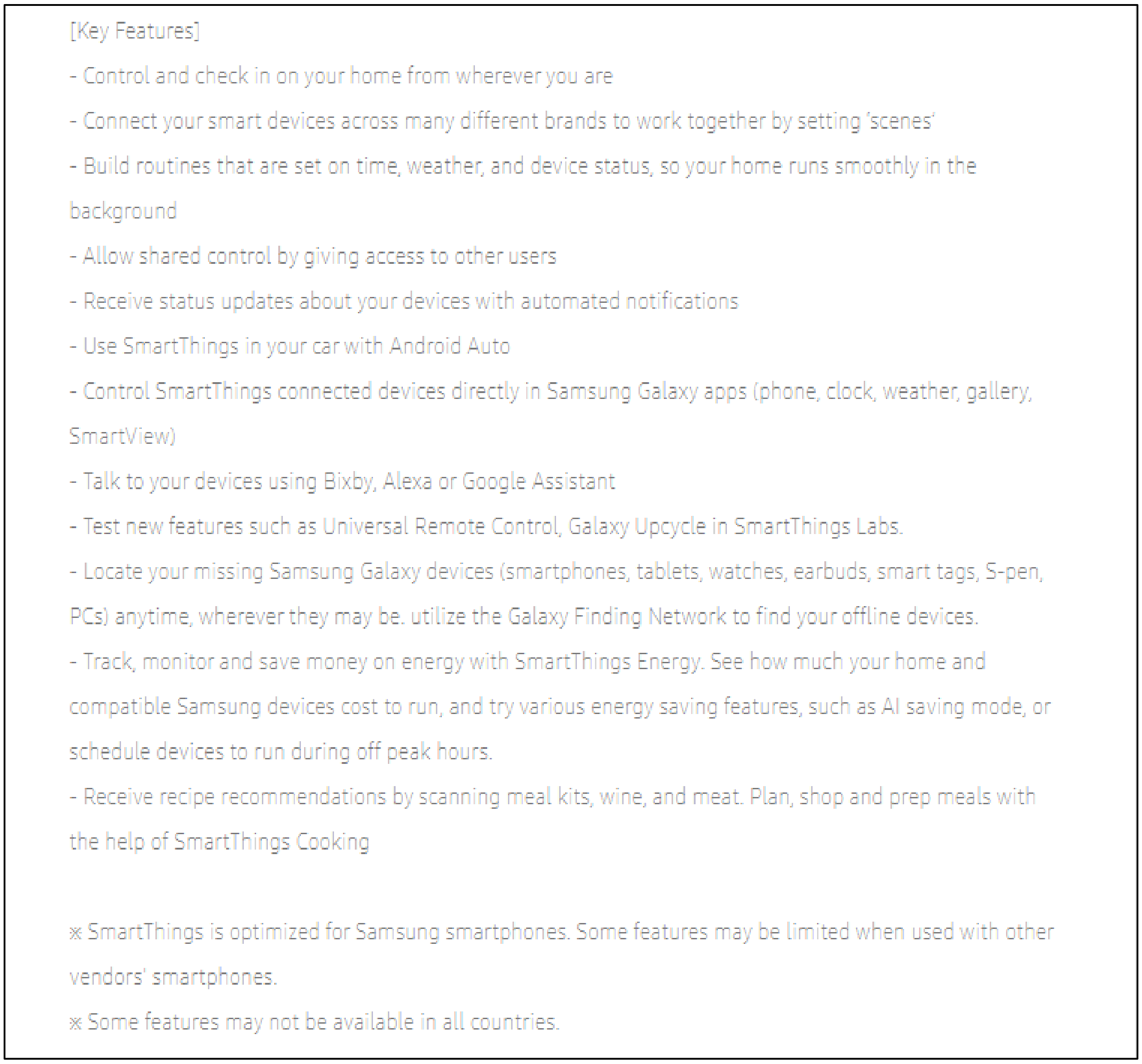
Samsung Electronics Co., Ltd.

2365 User Reviews

Free

Install
Copy URL
Build your smart home with SmartThings
Set your favorites
Gather the devices, services, and scenes you use the most so you can access them quickly
Control your devices
Monitor and control smart TVs, light bulbs, appliances, and other devices
Make your life smarter
Add services to do even more with your devices
Automa devi
Introduction
Reviews (2365)

https://galaxystore.samsung.com/prepost/000004262296?langCd=en



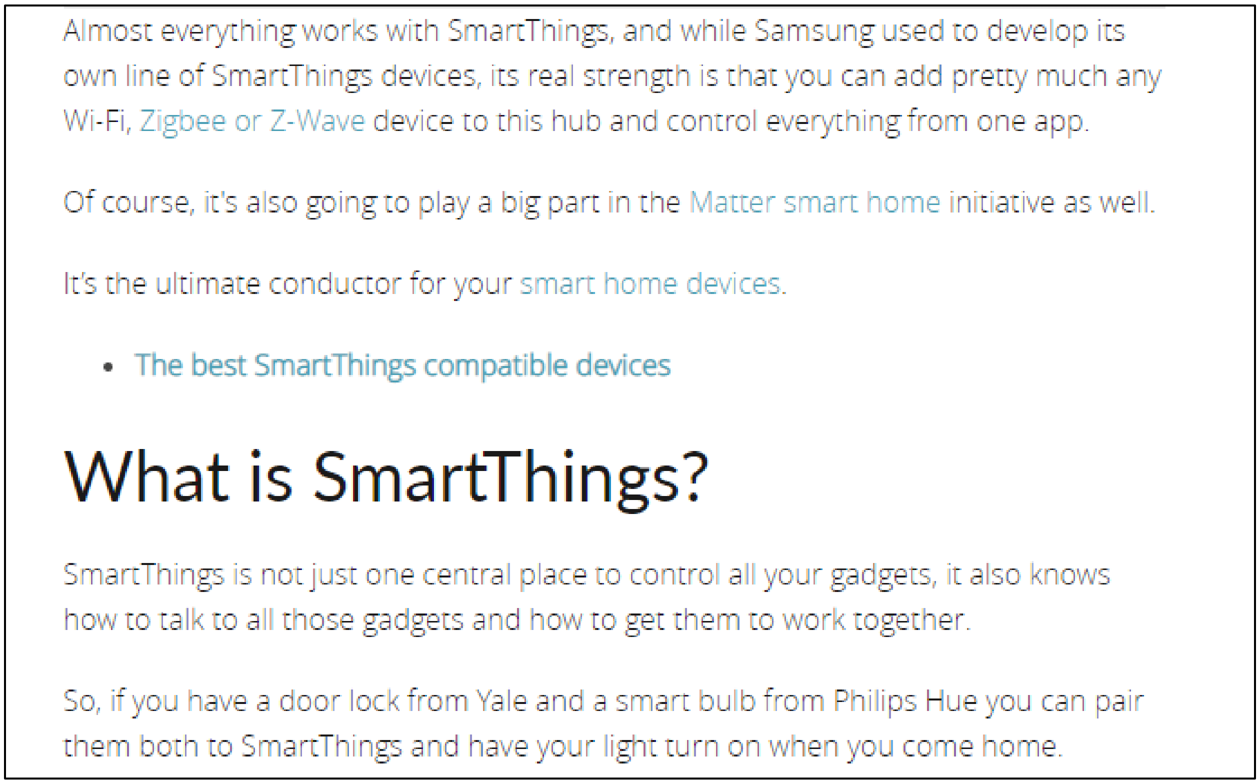
[Key Features]

- Control and check in on your home from wherever you are
- Connect your smart devices across many different brands to work together by setting 'scenes'
- Build routines that are set on time, weather, and device status, so your home runs smoothly in the background
- Allow shared control by giving access to other users
- Receive status updates about your devices with automated notifications
- Use SmartThings in your car with Android Auto
- Control SmartThings connected devices directly in Samsung Galaxy apps (phone, clock, weather, gallery, SmartView)
- Talk to your devices using Bixby, Alexa or Google Assistant
- Test new features such as Universal Remote Control, Galaxy Upcycle in SmartThings Labs.
- Locate your missing Samsung Galaxy devices (smartphones, tablets, watches, earbuds, smart tags, S-pen, PCs) anytime, wherever they may be. utilize the Galaxy Finding Network to find your offline devices.
- Track, monitor and save money on energy with SmartThings Energy. See how much your home and compatible Samsung devices cost to run, and try various energy saving features, such as AI saving mode, or schedule devices to run during off peak hours.
- Receive recipe recommendations by scanning meal kits, wine, and meat. Plan, shop and prep meals with the help of SmartThings Cooking

※ SmartThings is optimized for Samsung smartphones. Some features may be limited when used with other vendors' smartphones.

※ Some features may not be available in all countries.

Id.



Almost everything works with SmartThings, and while Samsung used to develop its own line of SmartThings devices, its real strength is that you can add pretty much any Wi-Fi, Zigbee or Z-Wave device to this hub and control everything from one app.

Of course, it's also going to play a big part in the Matter smart home initiative as well.

It's the ultimate conductor for your smart home devices.

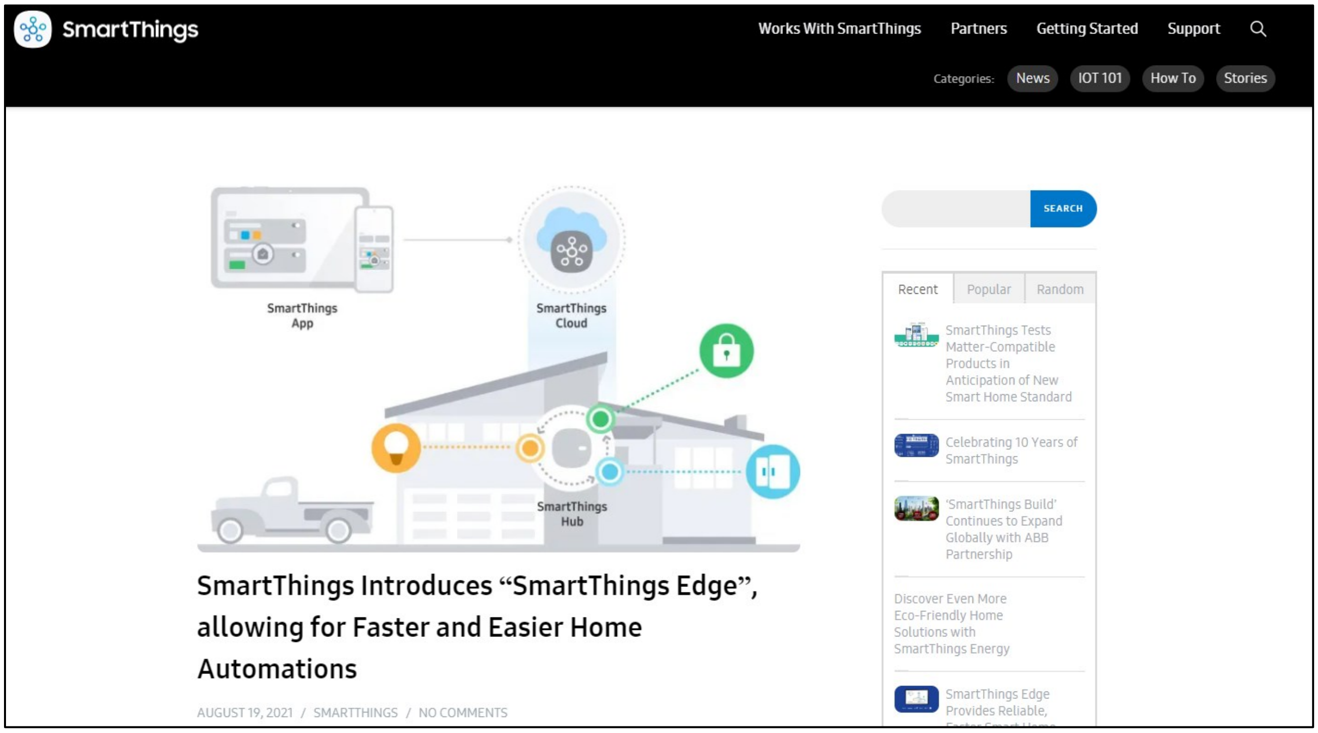
- The best SmartThings compatible devices

What is SmartThings?

SmartThings is not just one central place to control all your gadgets, it also knows how to talk to all those gadgets and how to get them to work together.

So, if you have a door lock from Yale and a smart bulb from Philips Hue you can pair them both to SmartThings and have your light turn on when you come home.

https://www.the-ambient.com/guides/samsung-smartthings-guide-smart-home-163



SmartThings Introduces "SmartThings Edge", allowing for Faster and Easier Home Automations

AUGUST 19, 2021 / SMARTTHINGS / NO COMMENTS

SmartThings Tests Matter-Compatible Products in Anticipation of New Smart Home Standard

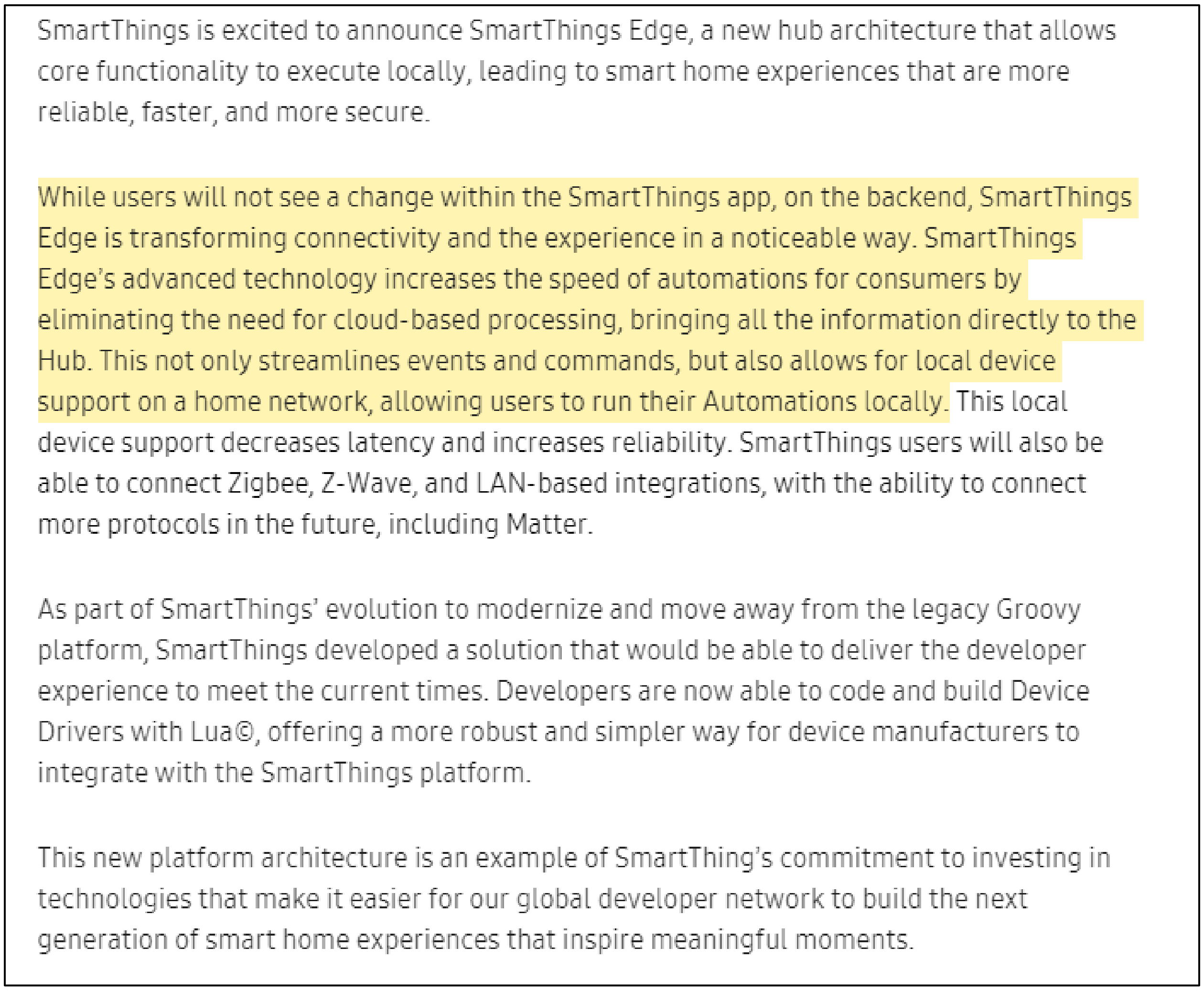
Celebrating 10 Years of SmartThings

'SmartThings Build' Continues to Expand Globally with ABB Partnership

Discover Even More Eco-Friendly Home Solutions with SmartThings Energy

SmartThings Edge Provides Reliable,

https://blog.smartthings.com/roundups/smartthings-introduces-smartthings-edgeallowing-for-faster-and-easier-home-automations/

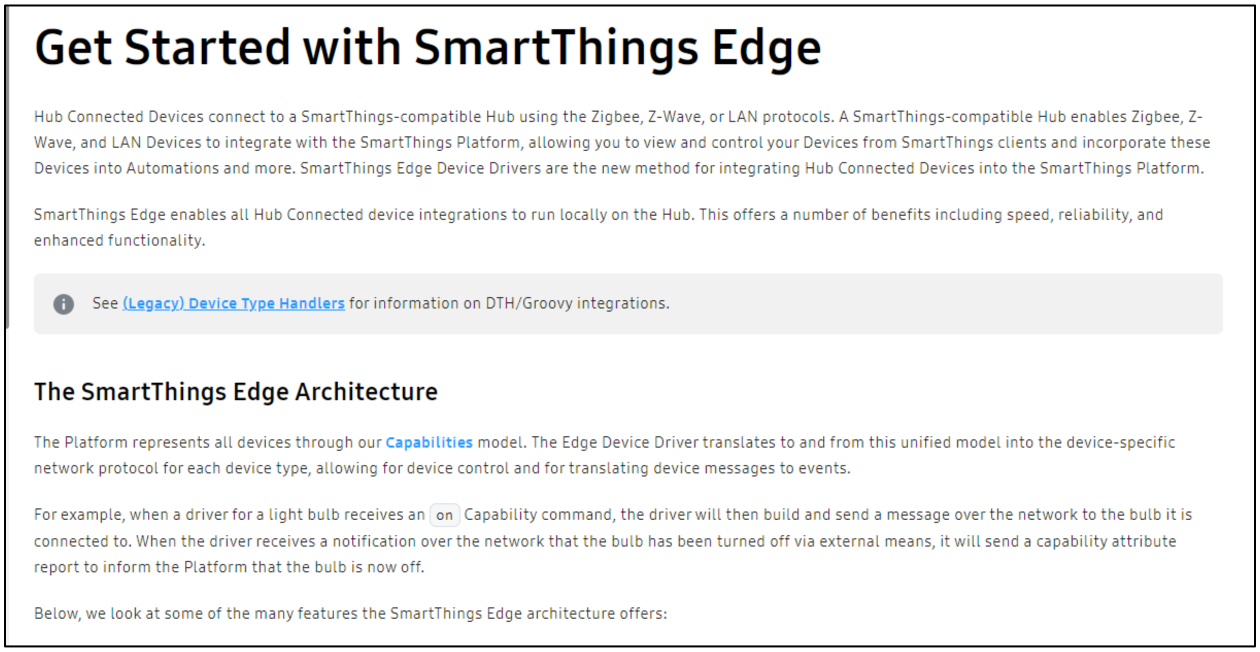

SmartThings is excited to announce SmartThings Edge, a new hub architecture that allows core functionality to execute locally, leading to smart home experiences that are more reliable, faster, and more secure.

While users will not see a change within the SmartThings app, on the backend, SmartThings Edge is transforming connectivity and the experience in a noticeable way. SmartThings Edge's advanced technology increases the speed of automations for consumers by eliminating the need for cloud-based processing, bringing all the information directly to the Hub. This not only streamlines events and commands, but also allows for local device support on a home network, allowing users to run their Automations locally. This local device support decreases latency and increases reliability. SmartThings users will also be able to connect Zigbee, Z-Wave, and LAN-based integrations, with the ability to connect more protocols in the future, including Matter.

As part of SmartThings' evolution to modernize and move away from the legacy Groovy platform, SmartThings developed a solution that would be able to deliver the developer experience to meet the current times. Developers are now able to code and build Device Drivers with Lua©, offering a more robust and simpler way for device manufacturers to integrate with the SmartThings platform.

This new platform architecture is an example of SmartThing's commitment to investing in technologies that make it easier for our global developer network to build the next generation of smart home experiences that inspire meaningful moments.

Id.



Get Started with SmartThings Edge

Hub Connected Devices connect to a SmartThings-compatible Hub using the Zigbee, Z-Wave, or LAN protocols. A SmartThings-compatible Hub enables Zigbee, Z-Wave, and LAN Devices to integrate with the SmartThings Platform, allowing you to view and control your Devices from SmartThings clients and incorporate these Devices into Automations and more. SmartThings Edge Device Drivers are the new method for integrating Hub Connected Devices into the SmartThings Platform.

SmartThings Edge enables all Hub Connected device integrations to run locally on the Hub. This offers a number of benefits including speed, reliability, and enhanced functionality.

See (Legacy) Device Type Handlers for information on DTH/Groovy integrations.

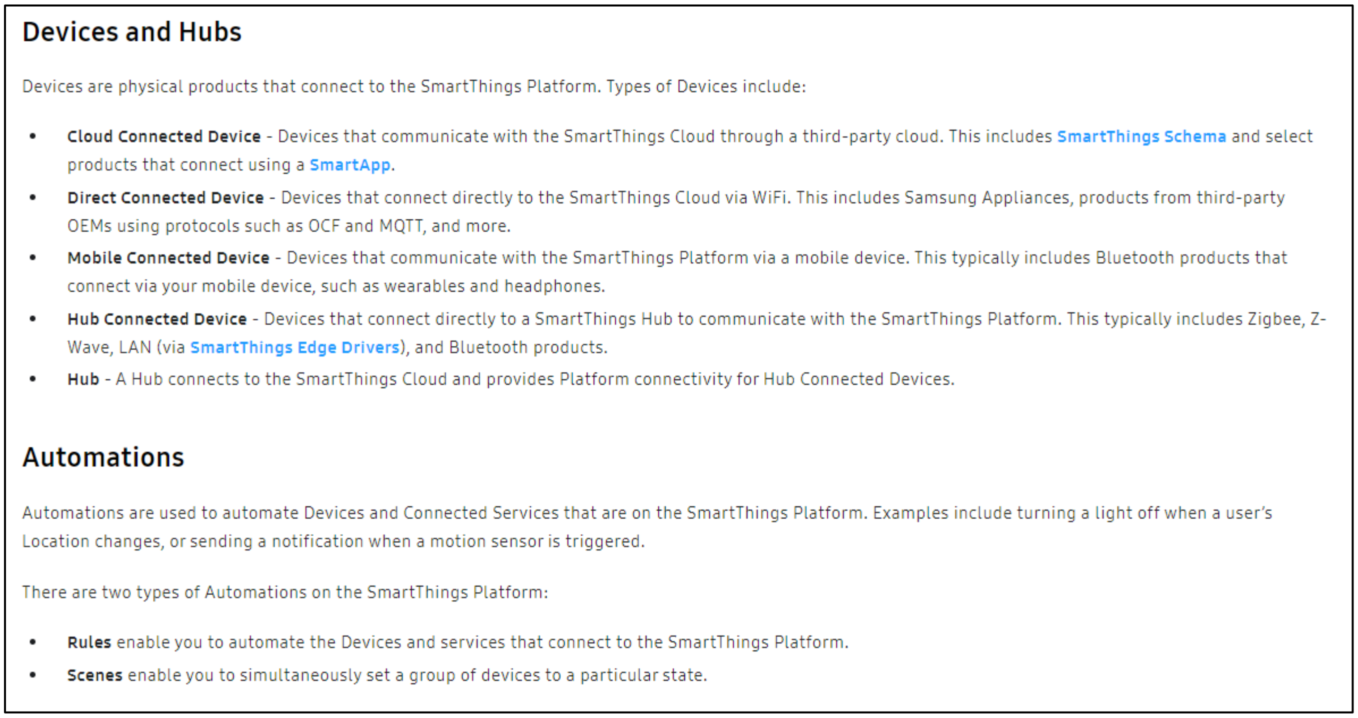
The SmartThings Edge Architecture

The Platform represents all devices through our **Capabilities** model. The Edge Device Driver translates to and from this unified model into the device-specific network protocol for each device type, allowing for device control and for translating device messages to events.

For example, when a driver for a light bulb receives an `on` Capability command, the driver will then build and send a message over the network to the bulb it is connected to. When the driver receives a notification over the network that the bulb has been turned off via external means, it will send a capability attribute report to inform the Platform that the bulb is now off.

Below, we look at some of the many features the SmartThings Edge architecture offers:

https://developer-preview.smartthings.com/docs/devices/hub-connected/get-started/



Devices and Hubs

Devices are physical products that connect to the SmartThings Platform. Types of Devices include:

- **Cloud Connected Device** - Devices that communicate with the SmartThings Cloud through a third-party cloud. This includes **SmartThings Schema** and select products that connect using a **SmartApp**.
- **Direct Connected Device** - Devices that connect directly to the SmartThings Cloud via WiFi. This includes Samsung Appliances, products from third-party OEMs using protocols such as OCF and MQTT, and more.
- **Mobile Connected Device** - Devices that communicate with the SmartThings Platform via a mobile device. This typically includes Bluetooth products that connect via your mobile device, such as wearables and headphones.
- **Hub Connected Device** - Devices that connect directly to a SmartThings Hub to communicate with the SmartThings Platform. This typically includes Zigbee, Z-Wave, LAN (via **SmartThings Edge Drivers**), and Bluetooth products.
- **Hub** - A Hub connects to the SmartThings Cloud and provides Platform connectivity for Hub Connected Devices.

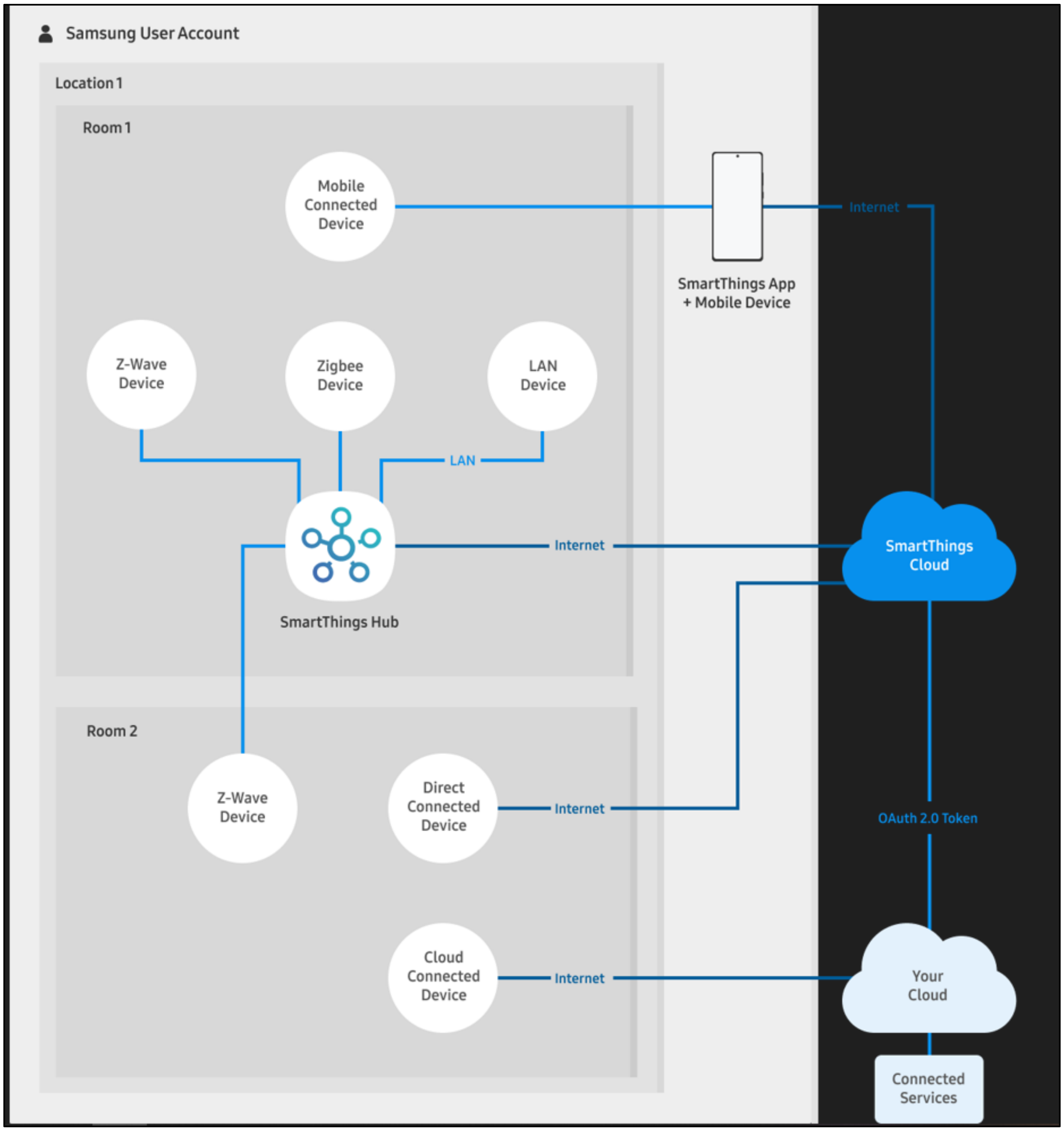
Automations

Automations are used to automate Devices and Connected Services that are on the SmartThings Platform. Examples include turning a light off when a user's Location changes, or sending a notification when a motion sensor is triggered.

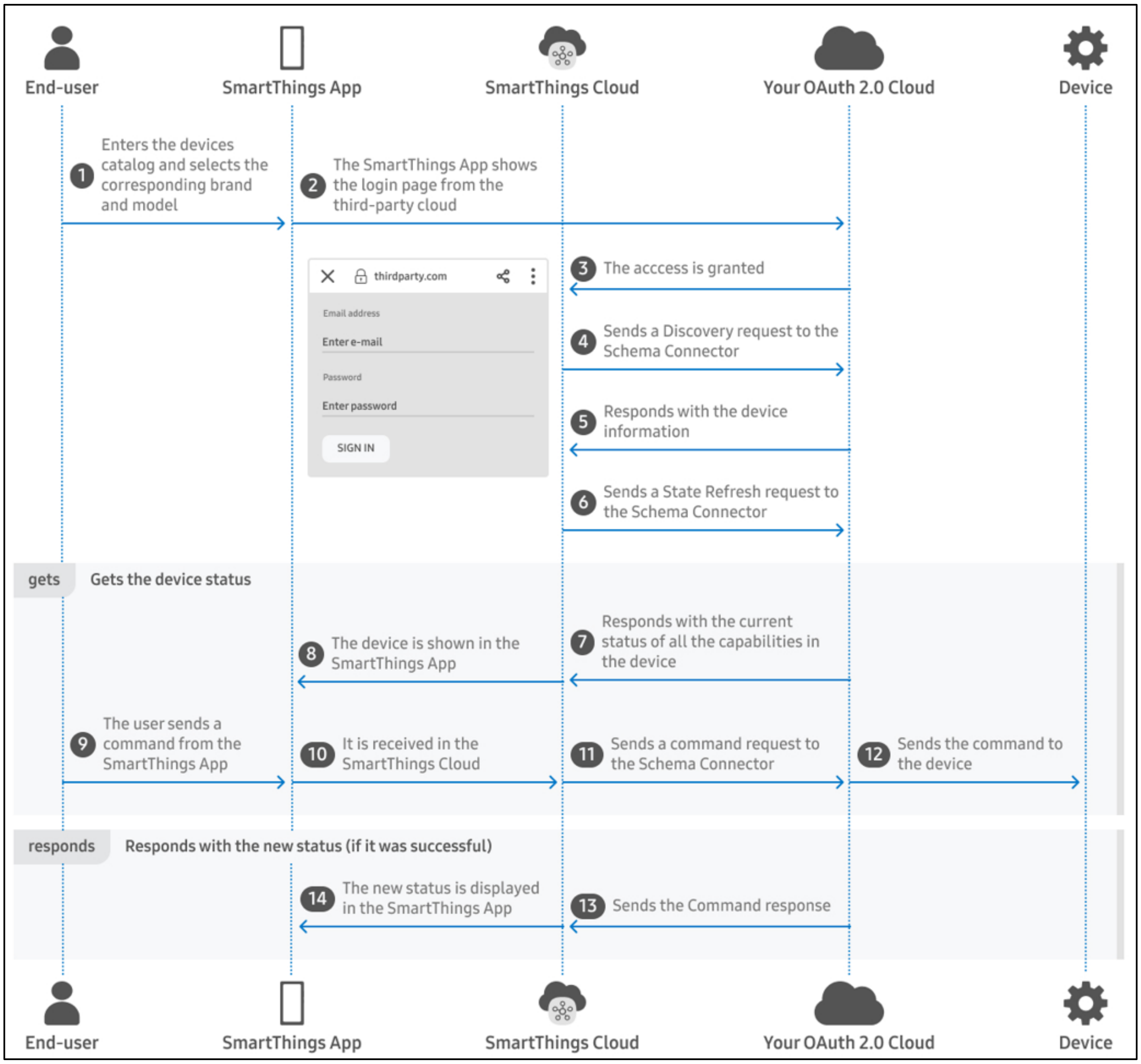
There are two types of Automations on the SmartThings Platform:

- **Rules** enable you to automate the Devices and services that connect to the SmartThings Platform.
- **Scenes** enable you to simultaneously set a group of devices to a particular state.

Id.



https://developer-preview.smartthings.com/docs/getting-started/architecture-of-smartthings


End-user
SmartThings App
SmartThings Cloud
Your OAuth 2.0 Cloud
Device
1 Enters the devices catalog and selects the corresponding brand and model
2 The SmartThings App shows the login page from the third-party cloud
thirdparty.com
Email address
Enter e-mail
Password
Enter password
SIGN IN
3 The acccess is granted
4 Sends a Discovery request to the Schema Connector
5 Responds with the device information
6 Sends a State Refresh request to the Schema Connector
gets Gets the device status
7 Responds with the current status of all the capabilities in the device
8 The device is shown in the SmartThings App
9 The user sends a command from the SmartThings App
10 It is received in the SmartThings Cloud
11 Sends a command request to the Schema Connector
12 Sends the command to the device
responds Responds with the new status (if it was successful)
13 Sends the Command response
14 The new status is displayed in the SmartThings App
End-user
SmartThings App
SmartThings Cloud
Your OAuth 2.0 Cloud
Device

https://developer-preview.smartthings.com/docs/devices/cloud-connected/get-started

52. By engaging in the conduct described herein, Samsung has injured Plaintiffs and is thus liable for infringement of the ’247 Patent, pursuant to 35 U.S.C. § 271. Samsung has committed these acts of infringement without license or authorization.

53. As a result of Samsung's infringement of the '247 Patent, Plaintiffs have suffered monetary damages and are entitled to a monetary judgment in an amount adequate to compensate for Samsung's past infringement, together with interests and costs. In addition, Samsung's infringement is causing irreparable harm and monetary damage to Plaintiffs and will continue to do so unless and until Samsung is enjoined by the Court.

DEMAND FOR JURY TRIAL

54. Plaintiffs hereby demand a trial by jury on all claims so triable.

PRAYER FOR RELIEF

WHEREFORE, Plaintiffs respectfully request that this Court enter judgment in their favor and grant the following relief:

A. Adjudge that Samsung infringes the Asserted Patents;

B. Adjudge that the Asserted Patents are valid and enforceable;

C. Award Plaintiffs damages in an amount adequate to compensate Plaintiffs for Samsung's infringement of the Asserted Patents, but in no event less than a reasonable royalty under 35 U.S.C. § 284;

D. Award enhanced damages pursuant to 35 U.S.C. § 284;

E. Award Plaintiffs pre-judgment and post-judgment interest to the full extent allowed under the law, as well as their costs;

F. Enter an order finding that this is an exceptional case and awarding Plaintiffs their reasonable attorneys' fees pursuant to 35 U.S.C. § 285;

G. Enter a permanent injunction against all Samsung products found to infringe the Asserted Patents;

H. Award, in lieu of an injunction, a compulsory forward royalty;

I. Order an accounting of damages; and

J. Award such other relief, including equitable relief, as the Court may deem appropriate and just under the circumstances.

DATED: September 2, 2022

Respectfully submitted,

/s/ Thomas M. Melsheimer
Thomas M. Melsheimer
Texas Bar No. 13922550
tmelsheimer@winston.com
M. Brett Johnson
Texas Bar No. 00790975
mbjohnson@winston.com
Rex A. Mann
Texas Bar No. 24075509
rmann@winston.com
Chad B. Walker
Texas Bar No. 24056484
cbwalker@winston.com
Ahtoosa A. Dale
Texas Bar No. 24101443
adale@winston.com
WINSTON & STRAWN LLP
2121 North Pearl Street, Suite 900
Dallas, TX 75201
Telephone: (214) 453-6500

J. Tyler Boyce
Texas Bar No. 24127214
tboyce@winston.com
WINSTON & STRAWN LLP
800 Capitol Street, Suite 2400
Houston, TX 77002
Telephone: (713) 651-2600

Matt Hopkins (to be admitted *pro hac vice*)
mhopkins@winston.com
WINSTON & STRAWN LLP
1901 L Street NW
Washington, D.C. 20036
Telephone: (202) 282-5000

Wesley Hill
Texas Bar. No. 24032294
Charles Everingham IV
Texas Bar No. 00787447
WARD, SMITH & HILL, PLLC

1507 Bill Owens Parkway
Longview, Texas 75604
(903) 757-6400 (main line)
(903) 757-2323 (facsimile)
wh@wsfirm.com
ce@wsfirm.com

ATTORNEYS FOR PLAINTIFFS