

Exhibit A - U.S. Patent No. 9,179,359 (“’359 Patent”)

Accused Instrumentalities: Google Pixel phones, tablets, and wearables, and all versions and variations thereof since the issuance of the asserted patent

Claim 1

Issued Claim(s)	Public Documentation
<p>1. A wireless end-user device, comprising:</p>	<p>The Accused Instrumentalities are “a wireless end-user device.” For example, the Google Pixel 9 is a “wireless end-user device.”</p>  The image shows two views of a Google Pixel 9 smartphone. On the left is the front view, displaying a screen with a colorful, abstract pattern of green and red. On the right is the back view, showing a light green or mint-colored back cover with a large, dark camera module in the upper left corner and a faint 'G' logo in the center.
<p>1[a] a wireless wide area network (WWAN) modem to receive and transmit Internet data between the device and at least one WWAN when configured for and connected to the at least one WWAN;</p>	<p>The Accused Instrumentalities comprise “a wireless wide area network (WWAN) modem to receive and transmit Internet data between the device and at least one WWAN when configured for and connected to the at least one WWAN.”</p> <p>For example, the Google Pixel 9 includes a WWAN modem for communicating with mobile service base stations on a carrier-provided wireless network.</p>

**Network &
Connectivity**

5G

5G Non-Standalone (NSA), Standalone (SA), Sub6 / mmWave

LTE

Enhanced 4x4 MIMO, Up to 7CA, LTE Cat.20

Up to 2.0Gbps Download / Up to 200Mbps Upload

Wi-Fi

Wi-Fi 802.11 a/b/g/n/ac/ax 2.4G+5GHz+6GHz, HE160, MIMO, 1024-QAM

Up to 2.4Gbps Download / Up to 2.4Gbps Upload

Bluetooth

Bluetooth® v 5.2, USB type-C, NFC, Location(GPS, Galileo, Glonass, BeiDou)

Ultra Wide Band

*Requires optimal connection. Actual speed may vary depending on country, carrier and user environment.

*The bandwidths supported by the device may vary depending on the region or service provider.

*Download and upload speeds reaching up to 2.4Gbps only available with Wi-Fi 6E. Wi-Fi 6E only supported on Galaxy S22 Ultra and S22+.

Galaxy S22 has Wi-Fi 6.

*Galileo and BeiDou coverage may be limited. BeiDou may not be available for certain countries.

	<p>Network⁴¹</p> <p>5G mmWave + Sub 6GHz Model G2YBB⁴²</p> <p>GSM/EDGE: Quad-band (850, 900, 1800, 1900 MHz)</p> <p>UMTS/HSPA+/HSDPA: Bands 1,2,4,5,6,8,19</p> <p>LTE: Bands B1/2/3/4/5/7/8/12/13/14/17/18/19/20/25/26/28/29/30/38/40/41/48/66/71</p> <p>5G Sub-6 : Bands n1/2/3/5/7/8/12/14/20/25/26/28/29/30/38/40/41/48/66/70/71/77/78⁴³</p> <p>5G mmWave : Bands n258/260/261⁴⁴</p> <p>eSIM</p> <p>https://store.google.com/product/pixel_9_specs?hl=en-US</p>
<p>1[b] a wireless local area network (WLAN) modem to receive and transmit Internet data between the device and at least one WLAN when configured for and connected to the at least one WLAN; and</p>	<p>The Accused Instrumentalities comprise “a wireless local area network (WLAN) modem to receive and transmit Internet data between the device and at least one WLAN when configured for and connected to the at least one WLAN.”</p> <p>For example, the Google Pixel 9 includes a WLAN modem for communicating with a wi-fi network.</p>

Connectivity and Location

Wi-Fi 7 (802.11be) with 2.4GHz+5GHz+6GHz, 2x2+2x2 MIMO

Bluetooth® v5.3 with dual antennas for enhanced quality and connection

NFC

Google Cast

Dual Band GNSS

GPS, GLONASS, Galileo

https://store.google.com/product/pixel_9_specs?hl=en-US

	<div><div><h1>ConnectivityManager</h1><div>Added in API level 1</div><div>Kotlin Java</div></div><div><pre>public class ConnectivityManager extends Object</pre><div>java.lang.Object ↳ android.net.ConnectivityManager</div></div><div><p>Class that answers queries about the state of network connectivity. It also notifies applications when network connectivity changes.</p><p>The primary responsibilities of this class are to:</p><ol style="list-style-type: none">1. Monitor network connections (Wi-Fi, GPRS, UMTS, etc.)2. Send broadcast intents when network connectivity changes3. Attempt to "fail over" to another network when connectivity to a network is lost4. Provide an API that allows applications to query the coarse-grained or fine-grained state of the available networks5. Provide an API that allows applications to request and select networks for their data traffic</div></div>
<p>1[c] one or more processors configured to apply a stored WWAN differential traffic control policy to Internet data service provided using the WWAN modem and the at least one WWAN, and indicate to a particular application, via an application program interface (API), one or more network access conditions based on the applied WWAN differential traffic</p>	<p>The Accused Instrumentalities comprise “one or more processors configured to apply a stored WWAN differential traffic control policy to Internet data service provided using the WWAN modem and the at least one WWAN, and indicate to a particular application, via an application program interface (API), one or more network access conditions based on the applied WWAN differential traffic control policy, wherein the one or more network access conditions includes a network access condition that indicates the unavailability to the particular application of an Internet data service that is currently available via the WWAN modem to a different application, but that is indicated as unavailable to the particular application.”</p>

control policy, wherein the one or more network access conditions includes a network access condition that indicates the unavailability to the particular application of an Internet data service that is currently available via the WWAN modem to a different application, but that is indicated as unavailable to the particular application.

For example, Google’s phones and tablets, including the Google Pixel 9, comprise one or more processors.

For example, the Pixel 9 uses a Google Tensor G4 processor

See https://store.google.com/product/pixel_9_specs?hl=en-US

Processor

Google Tensor G4

Titan M2 security coprocessor

https://store.google.com/intl/en_uk/ideas/articles/google-tensor-pixel-smartphone/

What is Google Tensor and how does it make smartphones more helpful?

Google Tensor is the system-on-chip AI processor co-designed with the Google DeepMind team, which is at the forefront of the latest innovations in machine learning and generative AI. Tensor was conceptualised in 2016 and first integrated on Pixel 6 in 2021. Tensor plays a key role in optimising large AI models for mobile devices, so Pixel can continue to have the latest innovations, such as instantly translating messages and videos⁵ or [summarising a long audio recording](#) without significantly draining your battery.

The processors are configured to “apply a stored WWAN differential traffic control policy to Internet data service provided using the WWAN modem and the at least one WWAN, and indicate to a particular application, via an application program interface (API), one or more network access conditions based on the applied WWAN differential traffic control policy, wherein the one or more network access conditions includes a network access condition that indicates the unavailability to the particular application of an Internet data service that is

currently available via the WWAN modem to a different application, but that is indicated as unavailable to the particular application.”

For example, Google Pixel phones and tablets include the Android operating system which include control policies that apply to Internet data service (e.g., wireless data communications).

Operating System

Launched with Android 14

https://store.google.com/product/pixel_9_specs?hl=en-US

As specific examples, Data Saver, Battery Saver, Extreme Battery Saver, Doze Mode, App Standby, Adaptive Battery, and/or Job Scheduler each apply one or more policies.

On information and belief, the Accused Instrumentalities include various features and functionalities which comprise one or more policies and indicate to an application one or more network access conditions, including, for example, the following features: Network Policy, ActivityManager, ConnectivityManager, netd, eBPF, Android’s Firewall, Data Usage, Data Saver, Power Saving, Doze Mode, App Standby, Adaptive Battery, JobScheduler, MediaPlayer, Media3, ExoPlayer, MediaSession APIs, Media APIs, Media Modules, TrafficStats, NetworkStats, App Freezer, Freeaccess, Managed App Restrictions (MARs), Sleeping Apps, Deep Sleeping Apps.

As a further example, a user may configure, on a per-application basis, whether applications are permitted to communicate while on a Wifi network and on a mobile, carrier-provided network, or only on one such network. The user interface in the Google devices allows the user to make these selections. This is shown such as by using Data Saver, Battery Saver, Extreme Battery Saver, Doze Mode, App Standby, Adaptive Battery, and/or Job Scheduler.

	<p>As a specific example, a particular configuration of Network Policy and/or ConnectivityManager can be an example of a control policy, and a different configuration of Network Policy and/or ConnectivityManager can be an example of a control policy.</p> <p>As further examples, settings for Data Saver, Data Usage, Power Saving Mode, Doze mode, App standby, Adaptive Battery, Managed App Restrictions (MARs), AppFreezer (FreeCess), Roaming Reduction, Device care, Background usage limits, sleeping/deep-sleeping apps and/or JobScheduler each apply one or more policies, alone or in combination with other features of Google's Network Policy and/or ConnectivityManager classes.</p> <p>As a further example, a user may configure, on a per-application basis, whether applications are permitted to communicate while on a Wifi network and on a mobile, carrier- provided network, or only on one such network. The user interface in the Google devices allows the user to make these selections. These selections are then implemented in one or more software components of the device operating system, such as Network Policy, ConnectivityManager, and/or related components. A configuration of Network Policy and/or ConnectivityManager also implements one or more additional features which work together with the per-application network settings to reduce data usage by applications on the app's permitted network(s) and which comprise additional aspects of the Network Policy and/or ConnectivityManager configurations and thus additional policies. This is shown such as by using Google's Data Usage, Data Saver, Power Saving Mode, Doze mode, App standby, Adaptive Battery, Managed App Restrictions (MARs), AppFreezer (FreeCess), Roaming Reduction, Device care, Background usage limits, and sleeping/deep-sleeping apps.</p> <p>With respect to Data Saver, the Android Operating System in applies a control policy and indicates unavailability based on the applied control policy via an API to an application, such as where the Network Policy Manager, Network Policy Manager Service, ConnectivityManager, and Activity Manager indicate unavailability to a particular applications such as Facebook, Instagram, etc., while indicating availability to other applications. The Android OS Network Policy Manager, Network Policy Manager Service, ConnectivityManager, and Activity Manager, alone and/or in combination, make this indication on a consistent and regular basis. As an example, Data Saver can be configured to apply different policies to different applications, such as a policy for applications in the background, and a second policy for applications in the foreground.</p>
--	---

Use less mobile data with Data Saver

To help use less mobile data on a limited data plan, you can turn on Data Saver. This mode lets most apps and services get background data only via Wi-Fi. Currently active apps and services can use mobile data.

Important: You're using a custom Android version. If these steps don't work for you, [check with your device manufacturer](#). **Important:** Some of these steps work only on Android 8.0 and up. [Learn how to check your Android version](#).

Turn Data Saver on or off

1. Open your phone's Settings app.
2. Tap **Network & internet** > **Data Saver**.
3. Turn **Data Saver** on or off.
 - In your status bar, when Data Saver is on, you'll see the Data Saver icon .
 - You'll also see a notification at the top of your phone's Settings app.

Tip: You can add Data Saver to your settings bar. [Learn how to customize Quick Settings](#).

Prevent apps from being interrupted when there's no Wi-Fi

Some apps and services won't work as expected unless you let them run in the background even when you're not using them. To let apps run in the background using mobile data, you can turn on "Unrestricted data" for those apps.

1. Open your phone's Settings app.
2. Tap **Network & internet** > **Data Saver** > **Unrestricted data**.
3. Turn on the app or service you'd like to use mobile data while Data Saver is on.

<https://support.google.com/pixelphone/answer/7055392?hl=en>

As another example, on information and belief, Google's "Power Saving," "Doze Mode," "App Standby," "Adaptive Battery," and/or "JobScheduler" features, all rely on the Network Policy, ConnectivityManager, and Activity Manager in materially the same manner as Data Saver for purposes of infringement. With respect to these features, Network Policy and/or ConnectivityManager likewise indicates conditions via an API based on an applied policy in connection with "Power Saving," "Doze Mode," "App Standby," "Adaptive Battery," "Managed App Restrictions" (MARs), "AppFreezer" (FreeCess), "Roaming Reduction," "Device

	care,” “Data Usage,” “JobScheduler,” Background usage limits, and/or sleeping/deep-sleeping apps features
--	---

Use Battery Saver on a Pixel phone

To make your battery last longer, you can set Battery Saver to turn on automatically. You can also turn on Battery Saver at any time. To save even more power on your device, you can turn on Extreme Battery Saver.

Important: While Battery Saver is on, a Pixel phone with 5G may use 4G service that depends on your carrier network. [Learn what changes while Battery Saver is on.](#)

Important:

- Some of these steps work only on Android 11 and up. [Learn how to check your Android version.](#)
- To improve battery performance on Pixel 6 and later, check battery diagnostics:
 1. Open your phone's Settings app.
 2. Tap **Battery** > **Battery diagnostics**.

How do I know if Battery Saver is turned on or off?

When Battery Saver is on, at the top of your screen, you'll find Battery Saver on . To turn Battery Saver on or off:

1. At the top of your screen, swipe down.
2. Tap Battery Saver .

Turn Battery Saver on automatically

1. Open your phone's Settings app.
2. Tap **Battery** > **Battery Saver** > **Schedule and reminders**.
3. Make sure that "Turn on based on battery level" is on.
4. Slide the percentage bar to the level you want.

Tip: The next time your battery level falls to the percentage you have set, Battery Saver will turn on automatically for you.



<https://support.google.com/pixelphone/answer/6187458?hl=en#:~:text=At%20the%20top%20of%20your%20screen%2C%20swipe%20down,to%20the%20level%20you%20want.>

Optimize network data usage

Over the life of a smartphone, the cost of a cellular data plan can easily exceed the cost of the device itself. On Android 7.0 (API level 24) and higher, users can enable Data Saver on a device-wide basis in order to optimize their device's data usage, and use less data. This ability is especially useful when roaming, near the end of the billing cycle, or for a small prepaid data pack.

When a user enables Data Saver in **Settings** and the device is on a metered network, the system blocks background data usage and signals apps to use less data in the foreground wherever possible. Users can allow specific apps to use background metered data usage even when Data Saver is turned on.

Android 7.0 (API level 24) extends the `ConnectivityManager` API to provide apps with a way to [retrieve the user's Data Saver preferences](#) and [monitor preference changes](#). It is considered good practice for apps to check whether the user has enabled Data Saver and make an effort to limit foreground and background data usage.

Check data saver preferences

On Android 7.0 (API level 24) and higher, apps can use the `ConnectivityManager` API to determine what data usage restrictions are being applied. The `getRestrictBackgroundStatus()` method returns one of the following values:

`RESTRICT_BACKGROUND_STATUS_DISABLED`

Data Saver is disabled.

`RESTRICT_BACKGROUND_STATUS_ENABLED`

The user has enabled Data Saver for this app. Apps should make an effort to limit data usage in the foreground and gracefully handle restrictions to background data usage.

`RESTRICT_BACKGROUND_STATUS_WHITELISTED`

The user has enabled Data Saver but the app is allowed to bypass it. Apps should still make an effort to limit foreground and background data usage.

Limit data usage whenever the device is connected to a metered network, even if Data Saver is disabled or the app is allowed to bypass it. The following sample code uses `ConnectivityManager.isActiveNetworkMetered()` and `ConnectivityManager.getRestrictBackgroundStatus()` to determine how much data the app should use:

<https://developer.android.com/training/basics/network-ops/data-saver:>

Optimize for Doze and App Standby

Starting from Android 6.0 (API level 23), Android introduces two power-saving features that extend battery life for users by managing how apps behave when a device is not connected to a power source. *Doze* reduces battery consumption by deferring background CPU and network activity for apps when the device is unused for long periods of time. *App Standby* defers background network activity for apps with which the user has not recently interacted.

While the device is in Doze, apps' access to certain battery-intensive resources is deferred until maintenance windows. The specific restrictions are listed in [Power Management Restrictions](#).

Doze and App Standby manage the behavior of all apps running on Android 6.0 or higher, regardless whether they are specifically targeting API level 23. To ensure the best experience for users, test your app in Doze and App Standby modes and make any necessary adjustments to your code. The sections below provide details.

Understanding Doze

If a user leaves a device unplugged and stationary for a period of time, with the screen off, the device enters Doze mode. In Doze mode, the system attempts to conserve battery by restricting apps' access to network and CPU-intensive services. It also prevents apps from accessing the network and defers their jobs, syncs, and standard alarms.

Periodically, the system exits Doze for a brief time to let apps complete their deferred activities. During this *maintenance window*, the system runs all pending syncs, jobs, and alarms, and lets apps access the network.

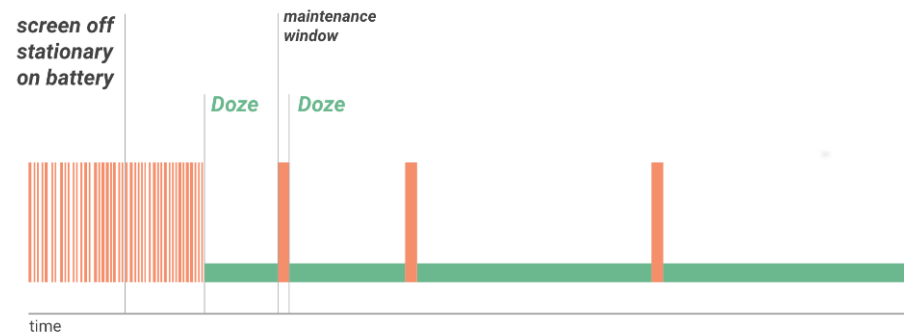


Figure 1. Doze provides a recurring maintenance window for apps to use the network and handle pending activities.

At the conclusion of each maintenance window, the system again enters Doze, suspending network access and deferring jobs, syncs, and alarms. Over time, the system schedules maintenance windows less and less frequently, helping to reduce battery consumption in cases of longer-term inactivity when the device is not connected to a charger.

As soon as the user wakes the device by moving it, turning on the screen, or connecting a charger, the system exits Doze and all apps return to normal activity.

The Doze restriction on network access is also likely to affect your app, especially if the app relies on real-time messages such as tickles or notifications. If your app requires a persistent connection to the network to receive messages, you should use [Firebase Cloud Messaging \(FCM\)](#) if possible.

<https://developer.android.com/training/monitoring-device-state/doze-standby>

App Standby Buckets

Android 9 (API level 28) and higher support **App Standby Buckets**. App Standby Buckets help the system prioritize apps' requests for resources based on how recently and how frequently the apps are used. Based on app usage patterns, each app is placed in one of five priority **buckets**. The system limits the device resources available to each app based on which bucket the app is in.

Priority buckets

The system dynamically assigns each app to a priority bucket, reassigning the apps as needed. The system may rely on a preloaded app that uses machine learning to determine how likely each app is to be used, and assigns apps to the appropriate buckets. If the system app is not present on a device, the system defaults to sorting apps based on how recently they were used. More active apps are assigned to buckets that give the apps higher priority, making more system resources available to the app. In particular, the bucket determines how frequently the app's jobs run, and how often the app can trigger alarms. These restrictions apply only while the device is on battery power; the system does not impose these restrictions on apps while the device is charging.

★ **Note:** Every manufacturer can set their own criteria for how non-active apps are assigned to buckets. You should not try to influence which bucket your app is assigned to. Instead, focus on making sure your app behaves well in whatever bucket it might be in. Your app can find out what bucket it's currently in by calling [UsageStatsManager.getAppStandbyBucket\(\)](#).

The buckets are:

1. **Active:** App is currently being used or was very recently used.
2. **Working set:** App is in regular use.
3. **Frequent:** App is often used, but not every day.
4. **Rare:** App is not frequently used.
5. **Restricted:** App consumes a great deal of system resources, or may exhibit undesirable behavior.

In addition, there's a special **never** bucket for apps that have been installed but have never been run. The system imposes severe restrictions on these apps.

<https://developer.android.com/topic/performance/appstandby>

Reduce & manage mobile data usage

Data usage is how much data your phone uploads or downloads using mobile data. To make sure that you're not using too much data on your data plan, you can check and change your data usage.

Important: You're using a custom Android version. If these steps don't work for you, [check with your device manufacturer](#).

Important: Some of these steps work only on Android 8.0 and up. [Learn how to check your Android version](#).

Check your mobile data usage

Reduce & manage mobile data usage

If you're on a Pixel device:

1. Open your phone's Settings app.
2. Tap **Network & internet** > **SIMs**.
3. At the top you'll find how much total data you use.
4. To get graphs and details, tap **App data usage**.
 - To pick a time period, tap the Down arrow ▾.
 - To know how much data each app uses, look below the graph.

Set mobile data usage warning or limit

<https://support.google.com/pixelphone/answer/2819524?hl=en&sjid=6915074452152146761-NC>

<https://developer.android.com/topic/performance/background-optimization>;
<https://developer.android.com/reference/android/app/job/JobScheduler>;
<https://developer.android.com/guide/background/persistent>;
<https://developer.android.com/guide/components/services>;
<https://developer.android.com/guide/components/activities/intro-activities>;
<https://developer.android.com/reference/java/net/URLConnection>;
<https://developer.android.com/training/articles/security-ssl>;
<https://developer.android.com/reference/android/net/DnsResolver>;
<https://developer.android.com/guide/topics/media>; <https://developer.android.com/media>;
<https://developer.android.com/guide/topics/media/platform/mediaplayer>; *see also*
<https://techshift.net/does-data-saver-apply-to-wi-fi/>;

For further examples, Google Pixel phones and tablets classify whether apps are running in the foreground or in the background. *See e.g.*,

1. A **foreground process** is one that is required for what the user is currently doing. Various application components can cause its containing process to be considered foreground in different ways. A process is considered to be in the foreground if any of the following conditions hold:

- It is running an **Activity** at the top of the screen that the user is interacting with (its **`onResume()`** method has been called).
- It has a **BroadcastReceiver** that is currently running (its **`BroadcastReceiver.onReceive()`** method is executing).
- It has a **Service** that is currently executing code in one of its callbacks (**`Service.onCreate()`**, **`Service.onStart()`**, or **`Service.onDestroy()`**).

There will only ever be a few such processes in the system, and these will only be killed as a last resort if memory is so low that not even these processes can continue to run. Generally, at this point, the device has reached a memory paging state, so this action is required in order to keep the user interface responsive.

<https://developer.android.com/guide/components/activities/process-lifecycle>;

Definition of background work

An app is running in the *background* when both the following conditions are satisfied:

- None of the app's activities are currently visible to the user.
- The app isn't running any [foreground services](#) that started while an activity from the app was visible to the user.

Otherwise, the app is running in the *foreground*.

<https://developer.android.com/guide/background>.

ConnectivityManager

Added in API level 1

[Kotlin](#) | **Java**

```
public class ConnectivityManager  
    extends Object
```

[java.lang.Object](#)

↳ [android.net.ConnectivityManager](#)

Class that answers queries about the state of network connectivity. It also notifies applications when network connectivity changes.

The primary responsibilities of this class are to:

1. Monitor network connections (Wi-Fi, GPRS, UMTS, etc.)
2. Send broadcast intents when network connectivity changes
3. Attempt to "fail over" to another network when connectivity to a network is lost
4. Provide an API that allows applications to query the coarse-grained or fine-grained state of the available networks
5. Provide an API that allows applications to request and select networks for their data traffic

For example, Android Operating System includes one or more stored WWAN differential traffic control policies applicable to Internet data service provided using the WWAN modem and the at least one WWAN, such as policy(ies) corresponding to Data Saver, Battery Saver, Extreme Battery Saver, Doze Mode, App Standby, Adaptive Battery, and/or Job Scheduler.

Claim 26

Issued Claim(s)	Public Documentation
<p>26. The wireless end-user device of claim 1, wherein the one or more processors are further configured to determine that the different application is an active application based on a user interaction with the different application.</p>	<p>The Accused Instrumentalities comprise the “wireless end-user device of claim 1, wherein the one or more processors are further configured to determine that the different application is an active application based on a user interaction with the different application.”</p> <p><i>See</i>, for example, the disclosures identified for claim 1[a]-1[c].</p> <p>On information and belief, Android Operating System in the Accused Instrumentalities is configured to determine whether an application is an active application based on a user interaction with an application, including, for example, in connection with the following features: Network Policy, ActivityManager, ConnectivityManager, netd, eBPF, Android’s Firewall, Data Usage, Data Saver, Power Saving, Doze Mode, App Standby, Adaptive Battery, JobScheduler, MediaPlayer, Media3, ExoPlayer, MediaSession APIs, Media APIs, Media Modules, TrafficStats, NetworkStats, App Freezer, Freecess, Managed App Restrictions (MARs), Sleeping Apps, Deep Sleeping Apps.</p> <p>On information and belief, Android Operating System includes one or more stored WWAN differential traffic control policies applicable to Internet data service provided using the WWAN modem and the at least one WWAN, such as policy(ies) corresponding to Data Usage, Data Saver, Power Saving Mode, Doze mode, App standby (later Adaptive Battery), Device care, Background usage limits, and sleeping/deep-sleeping, and/or Network Policy. Network Policy is an Android component that implements one or more stored WWAN differential traffic control policies in connection with one or more of these features. As one non-limiting example, Data Saver as implemented with Network Policy shows that the accused products comprise a stored WWAN differential traffic control policy. As another non-limiting example, Power Saving Mode / Battery Saver as implemented with Network Policy shows that the accused products comprise a stored WWAN differential traffic control policy. As another non-limiting example, Doze as implemented with Network Policy shows that the accused products comprise a stored WWAN differential traffic control policy. For example, such examples show traffic controls that are configured to be applicable to the WWAN and that are configured to differentiate between applications and allow certain applications to access the WWAN and that indicate availability of the WWAN to one application while also indicating unavailability to a different application of Internet data service that is currently available via the WWAN, thereby differentiating control</p>

among different applications. For example, through the Android Operating System, including, e.g., ActivityManager, NetworkPolicy, ConnectivityManager, and related components, the accused products with Android Operating System determines whether or not individual applications are in an interactive state such that the device classifies the application as having a state in which the user is directly interacting with the application or perceiving a benefit from the application, such as based on whether an application is running an Activity or a Foreground Service, versus where an application is merely running a service in the background that the user is not perceiving or interacting with. For example, the Android Operating System differentiates between a first particular application that the user of the device is not directly interacting with or perceiving a benefit from and a different application that has an Activity running at the Top of the screen which is in an interactive state or that has a Foreground Service in an interactive state and/or that the user is perceiving a benefit from. Then, via an API such as ConnectivityManager API, Android API, Network Policy, and related components, Android OS will, based on the applied differential traffic control policy, indicate network access conditions such as indicating unavailability to the particular application which is classified as background and ConnectivityManager also indicates availability to the different application which is classified as in the foreground of user interaction.

For example, on information and belief, the Android Operating System, including, e.g., ActivityManager, NetworkPolicy, ConnectivityManager, and related components, tracks whether a cellular network (WWAN) is currently available to the device for use by applications, whether a Wi-Fi network is currently available to the device for use by applications, and whether each application is requesting to use a network, such as an application requesting to use a cellular network. As an example, the rules in Network Policy are configured for several modes of operation to allow cellular network access to a requesting application that is considered by Android OS to be an active application based on a user interaction with the application and to block cellular network access for another requesting application that is not in active use according to Android OS.