Exposure Notification

Android API Documentation

Preliminary - Subject to Modification and Extension

April 2020
v1.2

Table of contents

Android Exposure Notification API

This document shows developers how to use the Exposure Notification API provided by Apple and Google to build Android apps for notifying users of possible exposure to confirmed COVID-19 cases.

This document contains the following sections:

- Architecture: provides an understanding of how the system is distributed across different sub-systems.
- Data structures and Methods: explain how to interact with the API from your app.
- App requirements: describes the functionality your app must provide to be complete.
- Glossary: provides definitions of terms specific to Exposure Notification.
- API reference: provides the reference documentation for all of the classes, methods, and fields.

Architecture

The Exposure Notification system spans several sub-systems on compatible Android devices. The following list explains this distribution.

Google Play services

Bluetooth functionality, including all broadcast and scanning for BLE beacons and local database storage, happens within Google Play services on-device. Your app will need to have the `BLUETOOTH` permission in its manifest, but does not require `ACCESS_FINE_LOCATION` or `BLUETOOTH_ADMIN`. For your app to work on a device, the device must be running Android version 5.0 (API version 21) or higher.

The following is the full list of the parts of the system that Google Play services handles:

- Manages daily random keys:
    - Generates daily random Temporary Exposure Keys and Rotating Proximity Identifiers (RPIs) based on them.
    - Provides keys to the application for diagnosed users, including an interval number that indicates the key date.
    - Accepts keys from the application for exposure detection, including an interval number that indicates the key date and an transmission risk level.
    - Stores key data in an on-device data store.
- Manages Bluetooth broadcast and collection:
    - Enables broadcast beacons.
    - Scans for other beacons.
    - Stores observed Rotating Proximity Identifiers (RPIs) in an on-device data store.
- Identifies whether the holder of the device was in close contact with a COVID-19 confirmed case.
    - Calculates and provides an exposure risk level to the app.
- Presents permission requests to users at the following points:
    - Before starting to scan for and broadcast beacons.
    - Before providing user keys to the app for uploading to the internet-accessible server once the user has been positively diagnosed with COVID-19.

This part of the system has no user interface other than the permission dialogs and notification message, and it runs in a low-power-consuming and privacy-centric way.

Mobile app

Your app does the following:

- Triggers dialogs in Google Play services for user permission flows.
- Enables users to start and stop broadcasting and scanning.
- Provides temporary tracing keys, key start time number, and key transmission risk level from your internet-accessible server to the Google Play services API.
- Retrieves keys from the on-device data store and submits them to your internet-accessible server after a user has been confirmed by a medical provider as having tested positive, and the user has provided permission.
- Shows a notification to the user with instructions on what to do next when the user has been exposed to another user who has tested positive for COVID-19.
- Provide the ability to delete all collected tracing keys from the on-device database. The deletion is performed by Google Play services, and is triggered by your app calling `resetAllData()`.

Authenticated medical provider validation mechanism

Your app must provide functionality that confirms that the holder of a device has been positively diagnosed with COVID-19. This functionality controls the release of diagnosis keys stored on a device whose user is positively diagnosed. The released keys are sent to your app, which then uploads them to your internet-accessible server.

Internet-accessible server

Your app must be able to communicate with an internet-accessible server that you own and that performs the following functions:

- Collecting diagnosis keys from users who have been diagnosed with COVID-19.
- When polled, distribute diagnosis keys to devices whose users have come in close contact with a user who has been diagnosed with COVID-19.

Data structures

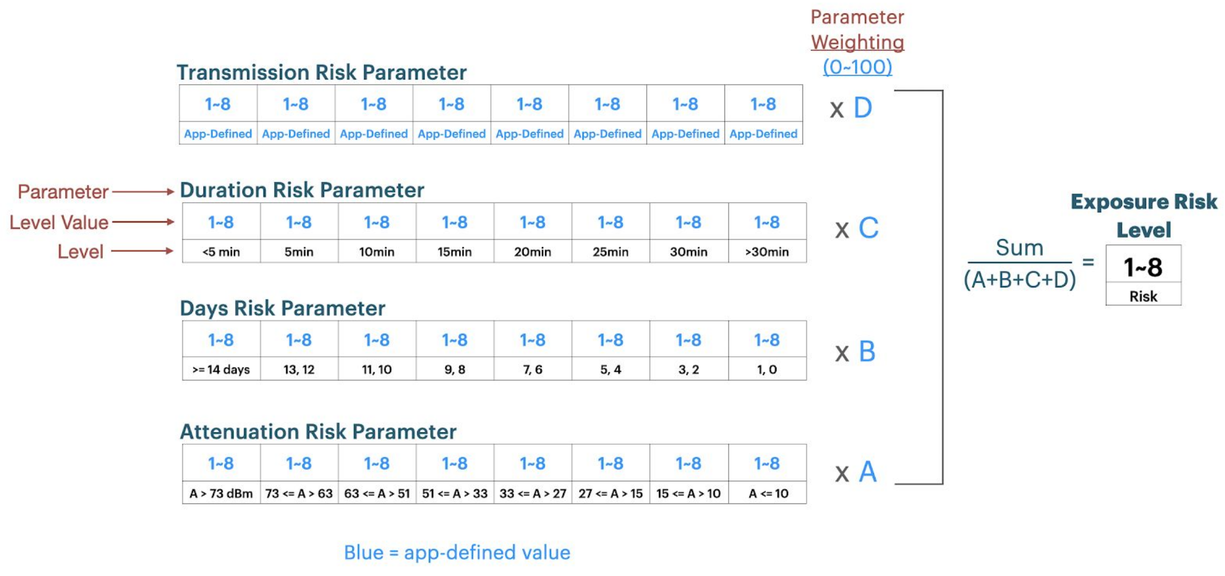
The API contains the data classes described in the following list.

`Status`

The Status class contains codes that represent the state of the service on the device.

`ExposureConfiguration`

The following diagram illustrates the data structure and formula used to calculate the Exposure Risk Level.



Parameter Weighting (0~100)
Transmission Risk Parameter
1~8 | 1~8 | 1~8 | 1~8 | 1~8 | 1~8 | 1~8 | 1~8
App-Defined | App-Defined | App-Defined | App-Defined | App-Defined | App-Defined | App-Defined | App-Defined
x D
Parameter
Level Value
Level
Duration Risk Parameter
1~8 | 1~8 | 1~8 | 1~8 | 1~8 | 1~8 | 1~8 | 1~8
<5 min | 5min | 10min | 15min | 20min | 25min | 30min | >30min
x C
Days Risk Parameter
1~8 | 1~8 | 1~8 | 1~8 | 1~8 | 1~8 | 1~8 | 1~8
>= 14 days | 13, 12 | 11, 10 | 9, 8 | 7, 6 | 5, 4 | 3, 2 | 1, 0
x B
Attenuation Risk Parameter
1~8 | 1~8 | 1~8 | 1~8 | 1~8 | 1~8 | 1~8 | 1~8
A > 73 dBm | 73 <= A > 63 | 63 <= A > 51 | 51 <= A > 33 | 33 <= A > 27 | 27 <= A > 15 | 15 <= A > 10 | A <= 10
x A
Sum (A+B+C+D) =
Exposure Risk Level
1~8
Risk
Blue = app-defined value

The `ExposureConfiguration` class contains fields that your app can pass in to the `Start` method:

- `minimumRiskScore`, an integer value between 1 and 8 that specifies how strong a calculated risk needs to be in order to be recorded.
- `attenuationScores`, an array in which you specify how much risk to associate with the Bluetooth attenuation value from an exposure. These values are multiplied by the value of the `attenuationWeight` field to determine a value for each range.
- `daysSinceLastExposureScores`, an array in which you specify how much risk to associate to an exposure based on the number of days since the exposure happened. These values are multiplied by the value of the `daysSinceLastExposureWeight` field to determine a value for each range.
- `durationScores`, an array in which you specify how much risk to associate with an exposure based on the duration of the exposure. These values are multiplied by the value of the `durationWeight` field to determine a value for each range.
- `transmissionRiskScores`, an array of custom values that specify how much risk to associate with a risk factor you designate. For example, you might associate the Highest Risk value with a user who has recently tested positive, and the Medium Risk for a user who came into contact with a positively-diagnosed person. These values are multiplied by the value of the `transmissionRiskWeight` field to determine a final value for each `transmissionRiskScores` value.

These fields are used in the following calculation to determine a `RiskScore`, which must exceed the `minimumRiskScore` value, for the exposure to be recorded:

```
RiskSum = (attenuationScore * attenuationWeight) +
(daysSinceLastExposureScore * daysSinceLastExposureWeight) +
(durationScore * durationWeight) + (transmissionRiskScore *
transmissionRiskWeight)
```

```
RiskScore = RiskSum / (attenuationWeight + daysSinceLastExposureWeight +
durationWeight + transmissionRiskWeight)
```

`TemporaryExposureKey`

The `TemporaryExposureKey` class holds the following:

- `keyData` that is used to generate broadcasts that are collected on the other devices. These connect to provide a record of the interaction between two devices. This information remains on a device until and unless the holder of the device tests positive, at which point the holder of the device can choose to share that information with the internet-accessible server.
- a `rollingStartNumber` that describes the time at which the key was generated.
- A `transmissionRiskLevel`, which specifies the level of risk of cross-exposure during the interaction between devices.

`ExposureSummary`

The `ExposureSummary` class stores the data in the following list:

- The number of days since the last match to a diagnosis key (`daysSinceLastExposure`).
- The number of days since the first match to a diagnosis key (`daysSinceFirstExposure`).
- The number of matched diagnosis keys (`matchedKeyCount`).
- The highest transmission risk level of all exposure incidents (`maximumRiskScore`).

Your app can present this information to users.

`ExposureInformation`

The `ExposureInformation` class stores information about an interaction with another device. It stores the day that the interaction occurred (`dateMillisSinceEpoch`), the duration of the exposure in five-minute increments (`durationMinutes`), the signal strength of the interaction (`attenuationValue`), and a transmission risk value (`transmissionRiskLevel`). You can use this information to gauge a level of risk of the exposure to filter out low-risk interactions, such as sub-five-minute interactions with a low signal strength attenuation that occurred 13 days earlier.

Methods

The API provides functionality using the methods shown in the following list. These are provided in a roughly chronological order of usage by an app.

Return values from methods are wrapped in a `Task` object to enable asynchronous operations.

These methods might produce errors. In those cases, use Task.getException(), cast the returned object to the `APIException` class, and then get the `Status` via `getStatus()` method, which provides information for troubleshooting. For example, the status might indicate that the user hasn't opted-in, the service isn't running, or that there is insufficient storage on the device to complete an operation.

`start()`

Takes an `ExposureConfiguration` object. It tells Google Play services to start the broadcasting and scanning process. The first time that this method is called after installation of the app, it prompts Google Play services to display a dialog box, where the user is asked to give permission to broadcast and scan.

`isEnabled()`

Indicates if exposure notifications are running.

`stop()`

Disables broadcasting and scanning. You can call this directly, and it is also called when users uninstall the app. When it's called as part of the uninstallation process, the database and keys are deleted from the device.

`getTemporaryExposureKeyHistory()`

Retrieves key history from the data store on the device for uploading to your internet-accessible server. Calling this method prompts Google Play services to display a dialog, requesting permission from the user to gather and upload their tracing keys.

`provideDiagnosisKeys()`

Inserts a list of up to `getMaxDiagnosisKeys()` number of exposure keys into the on-device database. Provide the keys of confirmed cases retrieved from your internet-accessible server to the Google Play service once requested from the API. If the list of keys is bigger than `getMaxDiagnosisKeys()`, you must paginate your requests.

`getMaxDiagnosisKeys()`

A utility function that provides a maximum number of keys to pass into `provideDiagnosisKeys()` per batch.

`getExposureSummary()`

Retrieves an `ExposureSummary` object for a high-level overview of the exposure that a user has experienced.

`getExposureInformation()`

Provides a more in-depth version of the information provided by `getExposureSummary()`. It provides a list of `ExposureInformation` objects, from which you can gauge the level of risk of the exposure with the user.

Plans to ensure user transparency in the use of this function are currently being evaluated.

`resetAllData()`

Provides a way for your app to delete all of the data collected by the system.

Glossary

BLE beacons

Bluetooth Low Energy (BLE) beacons enable Bluetooth-equipped devices to share information when they are within range of each other.

Diagnosis key

Daily keys provided by the server that have been confirmed to have a positive diagnosis of COVID-19. Used by the app to check for exposure.

Rotating Proximity Identifiers (RPIs)

A random ID derived from the device's daily tracing key. The key is generated on the device and a new key is generated every 10-20 minutes.

Temporary Exposure Key

A key derived from the device's tracing key once every 24 hours. It remains on the device for up to 14 days. In the event there is a positive diagnosis of COVID-19, and upon granting permission from the user, these keys are uploaded to the app's server.

Transmission Risk

A value defined by your app that specifies how much risk is associated with a given exposure based on a value of importance to your health authority. This enables you to add an additional, non-API-specified value to the computation of total risk.

API reference

```
/**
 * Starts BLE broadcasts and scanning based on the defined protocol.
 *
 * If not previously used, this shows a user dialog for consent to start exposure
 * detection and get permission. Exposure configuration options can be provided to
 * tune the matching algorithm (for example, setting a signal strength attenuation
 * or duration threshold).
 *
 * Callbacks regarding exposure status and requesting diagnosis keys from the
 * server will be provided via a BroadcastReceiver. Clients should register
 * a receiver in their AndroidManifest which can handle the following actions:
 * <ul>
 *   <li><code>
 *       com.google.android.gms.exposurenotification.ACTION_EXPOSURE_STATE_UPDATED
 *       </code>
 *   <li><code>
         com.google.android.gms.exposurenotification.ACTION_REQUEST_DIAGNOSIS_KEYS
 *       </code>
 * </ul>
 *
 * This receiver should also be guarded by the <code>
 * com.google.android.gms.nearby.exposurenotification.EXPOSURE_CALLBACK</code>
 * permission so that other apps are not able to fake this broadcast.
 */
Task<Void> start(ExposureConfiguration exposureConfiguration);

@IntDef({...})
@interface Status {
  int SUCCESS = 0;
  int FAILED_REJECTED_OPT_IN = 1;
  int FAILED_SERVICE_DISABLED = 2;
  int FAILED_BLUETOOTH_SCANNING_DISABLED = 3;
  int FAILED_TEMPORARILY_DISABLED = 4;
  int FAILED_INSUFFICENT_STORAGE = 5;
  int FAILED_INTERNAL = 6;
}
```

```
/**
 * Exposure configuration parameters that can be provided when initializing the
 * service.
 *
 * These parameters are used to calculate risk for each exposure incident using
 * the following formula:
 *
 * <p><code>
 *   RiskSum = (attenuationScore * attenuationWeight)
 *       + (daysSinceLastExposureScore * daysSinceLastExposureWeight)
 *       + (durationScore * durationWeight)
 *       + (transmissionRiskScore * transmissionRiskWeight)
 *   RiskScore = RiskSum
 *       / (attenuationWeight
 *          + daysSinceLastExposureWeight
 *          + durationWeight
 *          + transmissionRiskWeight)
 * </code>
 *
 * <p>Scores are in the range 1-8. Weights are in the range 0-100. RiskScore is in
 * the range 1-8.
 */
class ExposureConfiguration {
  /**
   * Minimum risk score. Excludes exposure incidents with scores lower than this.
   *
   * Defaults to no minimum.
   */
  int minimumRiskScore;

  /**
   * Scores for attenuation buckets. Must contain 8 scores, one for each bucket
   * as defined below:
   *
   * <p><code>{@code
   * attenuationScores[0] when Attenuation > 73
   * attenuationScores[1] when 73 >= Attenuation > 63
   * attenuationScores[2] when 63 >= Attenuation > 51
   * attenuationScores[3] when 51 >= Attenuation > 33
   * attenuationScores[4] when 33 >= Attenuation > 27
   * attenuationScores[5] when 27 >= Attenuation > 15
   * attenuationScores[6] when 15 >= Attenuation > 10
   * attenuationScores[7] when 10 >= Attenuation
   * }</code>
   */
  int[] attenuationScores;
```

```
/** Weight to apply to the attenuation score. Must be in the range 0-100. */
int attenuationWeight;

/**
 * Scores for days since last exposure buckets. Must contain 8 scores, one for
 * each bucket as defined below:
 *
 * <p><code>{@code
 * daysSinceLastExposureScores[0] when Days >= 14
 * daysSinceLastExposureScores[1] when Days >= 12
 * daysSinceLastExposureScores[2] when Days >= 10
 * daysSinceLastExposureScores[3] when Days >= 8
 * daysSinceLastExposureScores[4] when Days >= 6
 * daysSinceLastExposureScores[5] when Days >= 4
 * daysSinceLastExposureScores[6] when Days >= 2
 * daysSinceLastExposureScores[7] when Days >= 0
 * }</code>
 */
int[] daysSinceLastExposureScores;

/**
 * Weight to apply to the days since last exposure score. Must be in the
 * range 0-100.
 */
int daysSinceLastExposureWeight;

/**
 * Scores for duration buckets. Must contain 8 scores, one for each bucket as
 * defined below:
 *
 * <p><code>{@code
 * durationScores[0] when Duration == 0
 * durationScores[1] when Duration <= 5
 * durationScores[2] when Duration <= 10
 * durationScores[3] when Duration <= 15
 * durationScores[4] when Duration <= 20
 * durationScores[5] when Duration <= 25
 * durationScores[6] when Duration <= 30
 * durationScores[7] when Duration  > 30
 * }</code>
 */
int[] durationScores;

/** Weight to apply to the duration score. Must be in the range 0-100. */
double durationWeight;
```

```
  /**
   * Scores for transmission risk buckets. Must contain 8 scores, one for each
   * bucket as defined below:
   *
   * <p><code>{@code
   * transmissionRiskScores[0] when RISK_SCORE_LOWEST
   * transmissionRiskScores[1] when RISK_SCORE_LOW
   * transmissionRiskScores[2] when RISK_SCORE_LOW_MEDIUM
   * transmissionRiskScores[3] when RISK_SCORE_MEDIUM
   * transmissionRiskScores[4] when RISK_SCORE_MEDIUM_HIGH
   * transmissionRiskScores[5] when RISK_SCORE_HIGH
   * transmissionRiskScores[6] when RISK_SCORE_VERY_HIGH
   * transmissionRiskScores[7] when RISK_SCORE_HIGHEST
   * }</code>
   */
  int[] transmissionRiskScores;

  /**
   * Weight to apply to the transmission risk score. Must be in the range 0-100.
   */
  double transmissionRiskWeight;
}

/** Risk level defined for an {@link TemporaryExposureKey}. */
@IntDef({...})
@interface RiskLevel {
  int RISK_LEVEL_INVALID = 0;
  int RISK_LEVEL_LOWEST = 1;
  int RISK_LEVEL_LOW = 2;
  int RISK_LEVEL_LOW_MEDIUM = 3;
  int RISK_LEVEL_MEDIUM = 4;
  int RISK_LEVEL_MEDIUM_HIGH = 5;
  int RISK_LEVEL_HIGH = 6;
  int RISK_LEVEL_VERY_HIGH = 7;
  int RISK_LEVEL_HIGHEST = 8;
}

/** A key generated for advertising over a window of time. */
class TemporaryExposureKey {
  /** The randomly generated Temporary Exposure Key information. */
  byte[] keyData;

  /**
   * A number describing when a key starts.
   * It is equal to startTimeOfKeySinceEpochInSecs / (60 * 10).
   */
  long rollingStartNumber;
  /**
   * A number describing how long a key is valid.
   * It is expressed in increments of 10 minutes (e.g. 144 for 24 hrs).
   */
  long rollingDuration;


  /** Risk of transmission associated with the person this key came from. */
  @RiskLevel int transmissionRiskLevel;
}
```

```
/**
 * Disables advertising and scanning. Contents of the database and keys will
 * remain.
 *
 * If the client app has been uninstalled by the user, this will be automatically
 * invoked and the database and keys will be wiped from the device.
 */
Task<Void> stop();
```

```
/**
 * Indicates whether exposure notifications are currently running for the
 * requesting app.
 */
Task<Boolean> isEnabled();
```

```
 /**
 * Gets {@link TemporaryExposureKey} history to be stored on the server.
 *
 * This should only be done after proper verification is performed on the
 * client side that the user is diagnosed positive.
 *
 * The keys provided here will only be from previous days; keys will not be
 * released until after they are no longer an active exposure key.
 *
 * This shows a user permission dialog for sharing and uploading data to the
 * server.
 */
Task<List<TemporaryExposureKey>> getTemporaryExposureKeyHistory();
```

```
/**
 * Provides a list of diagnosis key files for exposure checking. The files are to
 * be synced from the server. Diagnosis keys older than the relevant period will be
 * ignored.
 *
 * When invoked after the
 * <code>com.google.android.gms.exposurenotification.ACTION_REQUEST_DIAGNOSIS_KEYS
 * </code> broadcast, this triggers a recalculation of exposure status which can be
 * obtained via {@link #getExposureSummary} after the calculation has finished.
 * When invoked outside of that action, diagnosis keys will still be stored and
 * matching will be performed in the near future.
 *
 * Should be called with a maximum of {@link #getMaxDiagnosisKeys()} keys at a
 * time, waiting until the Task finishes before providing the next batch.
 */
Task<Void> provideDiagnosisKeys(List<TemporaryExposureKey> keys);

/**
 * The maximum number of keys to pass into {@link #provideDiagnosisKeys} at any
 * given time.
 */
Task<Integer> getMaxDiagnosisKeys();
 */
Task<Void> provideDiagnosisKeys(List<ParcelFileDescriptor> keyFiles);
```

```
/**
 * Gets a summary of the latest exposure calculation. The calculation happens
 * asynchronously, the most recent check’s result will be returned immediately.
 */
Task<ExposureSummary> getExposureSummary();

/**
 * Summary information about recent exposures.
 *
 * The client can get this information via {@link #getExposureSummary}.
 */
class ExposureSummary {
  /**
   * Days since last match to a diagnosis key from the server. 0 is today, 1 is
   * yesterday, etc. Only valid if {@link #getMatchedKeysCount} > 0.
   */
  int daysSinceLastExposure;

  /** Number of matched diagnosis keys. */
  int matchedKeyCount;

  /** The highest risk score of all exposure incidents, it will be a value 1-8. */
  int maximumRiskScore;
```

```
}

/**
 * Gets detailed information about exposures that have occurred. The calculation
 * happens asynchronously, the most recent check’s result will be returned
 * immediately.
 *
 * When multiple {@link ExposureInformation} objects are returned, they can
 * be:
 * <ul>
 *   <li>Multiple encounters with a single diagnosis key.
 *   <li>Multiple encounters with the same device across key rotation boundaries.
 *   <li>Encounters with multiple devices.
 * </ul>
 *
 * Plans to ensure user transparency in the use of this function are currently
 * being evaluated.
 */
Task<List<ExposureInformation>> getExposureInformation();

/**
 * Information about an exposure, meaning a single diagnosis key
 * over a contiguous period of time specified by durationMinutes.
 *
 * The client can get the exposure information via {@link #getExposureInformation}.
 */
class ExposureInformation {
  /** Day level resolution that the exposure occurred. */
  long dateMillisSinceEpoch;

  /** Length of exposure in 5 minute increments, with a 30 minute maximum. */
  int durationMinutes;

  /**
   * Signal strength attenuation, representing the closest the two devices were
   * within the duration of the exposure. This value is the advertiser's TX power
   * minus the receiver's maximum RSSI.
   *
   * Note: This value may be misleading, higher attenuation does not necessarily
   * mean farther away. Phone in pocket vs hand can greatly affect this value,
   * along with other situations that can block the signal.
   *
   * This value will be in the range 0-255.
   */
  int attenuationValue;

  /** The transmission risk associated with the matched diagnosis key. */
  @RiskLevel int transmissionRiskLevel;

  /**
   * The total risk calculated for the exposure. See {@link ExposureConfiguration}
   * for more information about what is represented by the risk score.
   */
  int totalRiskScore;
}
```

```
/**
 * Delete all stored data associated with the user including exposure keys,
 * bluetooth scan history, and previously detected exposures.
 */
Task<Void> resetAllData();
```

Revision history

v1.2 - April 29, 2020

- Add conceptual material to the document
- Add risk level to `TemporaryExposureKey`
- Add risk score to `ExposureSummary` and `ExposureInformation`
- Rename `MatchingOptions` to `ExposureConfiguration`
    - Allow clients to pass in risk score parameters
- Remove handleIntent and `ExposureNotificationCallback` in favor of registering a `BroadcastReceiver`
- Pass `FileDescriptors` with signatures instead of raw keys to `provideDiagnosisKeys`
- Removed `resetTemporaryExposureKey`, which will instead be handled internally

v1.1 - April 21, 2020

- Renamed from Contact Tracing -> Exposure Notification
- Update `start()`
    - Allow sending in matching options, including `attenuationValueThreshold` and `durationMinutesThreshold`
    - Update documentation
- Update `ExposureNotificationCallback` javadocs and method naming to better conform to Android conventions
- Renamed `DailyTracingKey` to `TemporaryExposureKey`
    - date to `rollingStartNumber` per crypto spec
- Renamed `startSharingTemporaryTracingKeys` to `getTemporaryExposureKeyHistory` and return the keys directly
- Renamed `hasContact` to `getExposureSummary` and added `ExposureSummary`
- Changed `ContactInfo` to `ExposureInformation`
    - Added `attenuationValue`
    - Documented maximum `durationMinutes` to 30.
- Updated return types for all methods to conform with Google Play services style conventions
- Updated documentation on `provideDiagnosisKeys` that older keys will be ignored
- Added methods for a client to `resetAllData` or `resetTemporaryExposureKey`
- General doc cleanup

v1.0 - April 10, 2020

- Initial draft