Exposure Notification

iOS Framework Documentation

(API)

Preliminary — Subject to Modification and Extension

April 2020

v1.2

Contents

Overview

The ExposureNotification framework is designed to help you implement a privacy-preserving solution. It covers two user roles:

- *Affected User*. A user who has a confirmed or suspected diagnosis of COVID-19, the disease caused by the coronavirus pathogen.
- *Exposed User*. A user who has a potential exposure.

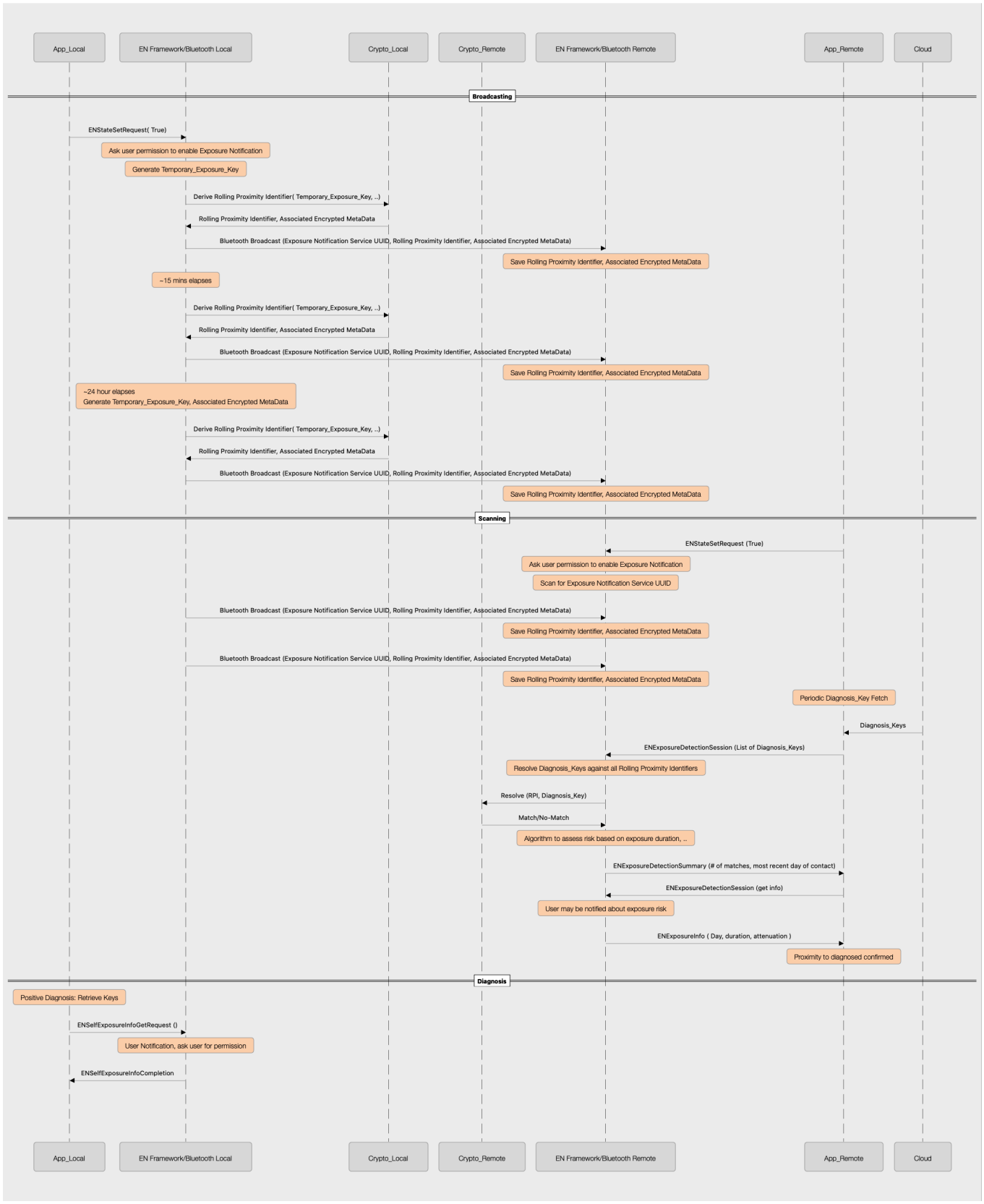
Affected User

When a user is diagnosed as positive, their Temporary Exposure Keys should be shared with other users to alert them to potential exposure. These Temporary Exposure Keys are retrieved using `ENSelfExposureInfoRequest`.

Exposed User

Given a set of Temporary Exposure Keys that indicate a positive diagnosis, the framework allows you to determine whether those Temporary Exposure Keys were observed locally by the user. If so, additional information such as date and duration may also be retrieved using `ENExposureDetectionFinishHandler`.

The following diagram outlines the flow of the ExposureNotification Framework for iOS.



ENErrorCode

Overview

A typedef that represents the error codes in the framework.

ENErrorCodeSuccess

Operation succeeded.

ENErrorCodeUnknown

Underlying failure with an unknown cause.

ENErrorCodeBadParameter

Missing or incorrect parameter.

ENErrorCodeNotEntitled

Calling process doesn't have the correct entitlement.

ENErrorCodeNotAuthorized

User denied this process access to Exposure Notification functionality.

ENErrorCodeUnsupported

Operation is not supported.

ENErrorCodeInvalidated

Invalidate was called before the operation completed normally.

ENErrorCodeBluetoothOff

Bluetooth was turned off the by user.

ENErrorCodeInsufficientStorage

Insufficient storage space to enable exposure notification.

ENErrorCodeNotEnabled

Exposure Notification has not been enabled.

ENErrorCodeAPIMisuse

API was used incorrectly.

ENErrorCodeInternal

Internal error indicating a bug in this framework.

ENErrorCodeInsufficientMemory

Not enough memory to perform an operation.

ENErrorCodeRateLimited

API called too frequently. See API for acceptable frequency.

ENAuthorizationStatus

Overview

An enumeration that indicates the status of authorization for the app.

ENAuthorizationStatusUnknown

Authorization status has not yet been determined.

ENAuthorizationStatusRestricted

This app is not authorized to use Exposure Notification. The user cannot change this app's authorization status. This status may be due to active restrictions, such as parental controls being in place.

ENAuthorizationStatusNotAuthorized

The user denied authorization for this app.

ENAuthorizationStatusAuthorized

The user has authorized this app to use Exposure Notification.

ENStatus

Overview

An enumeration that represents the overall status of Exposure Notification on the system.

ENStatusUnknown

The status of Exposure Notification is unknown. This is the status before `ENManager` has activated successfully.

ENStatusActive

Exposure Notification is active on the system.

ENStatusDisabled

Exposure Notification is disabled. Use `setExposureNotificationEnabled:completionHandler` to enable it.

ENStatusBluetoothOff

Bluetooth has been turned off on the system. Bluetooth is required for Exposure Notification.

Note: this may not match the state of Bluetooth as reported by CoreBluetooth.

Exposure Notification is a system service and can use Bluetooth in situations when apps cannot. For the purposes of Exposure Notification, it's better to use this API instead of CoreBluetooth.

ENStatusRestricted

Exposure Notification is not active due to system restrictions, such as parental controls. When in this state, the app cannot enable Exposure Notification.

ENManager

Overview

A class that manages Exposure Notification functionality.

Discussion

Before an instance of this class can be used, you must call `activateWithCompletionHandler`. If the completion handler completes successfully, the remaining properties and methods on the class can be used. Note that activating this object doesn't enable Exposure Notification; it only allows this object to be used. Once activated, Exposure Notification can be enabled with `setExposureNotificationEnabled:completionHandler`, if needed.

If the app no longer needs an instance of this class, you must call `invalidate`. This stops any outstanding operations and causes the invalidation handler to be invoked.

Note: Invalidation is asynchronous, so it's possible for handlers to be invoked after calling `invalidate`.

The invalidation handler is invoked when invalidation finishes. No handlers are invoked after that. The invalidation handler is invoked exactly once, even if `invalidate` is called multiple times.

Once `invalidate` is called, the object cannot be reused. A new object must be created for subsequent use. All strong references are cleared when invalidation completes to break potential retain cycles. You don't need to use weak references within your handlers to avoid retain cycles when using this class.

@property dispatch_queue_t dispatchQueue;

The dispatch queue on which to invoke handlers. The default is the main queue.

@property ENStatus exposureNotificationStatus;

The overall status of Exposure Notification. Key Value Observing may be used to monitor for changes.

@property dispatch_block_t invalidationHandler;

The handler that is invoked exactly once when invalidation completes. This property is cleared before it's invoked to break retain cycles.

- (void)activateWithCompletionHandler:(ENErrorHandler) completionHandler;

Activates the object to prepare it for use. Properties may not be usable until the completion handler reports success.

- (void)invalidate;

Stops any outstanding operations and invalidates this object. Once this is called, the object can no longer be used. To start using `ENManager` again, create and activate a new instance of the class.

@property ENAuthorizationStatus authorizationStatus;

This property reports the current authorization status of the app, and never prompts the user. This property can be used by the app to preflight authorization in order to determine if the user may be prompted.

@property BOOL exposureNotificationEnabled;

This property indicates whether Exposure Notification is enabled on the system. Key Value Observing may be used to monitor for changes. The value of this property will be `NO` until `activateWithCompletionHandler` has completed successfully.

Note that even if Exposure Notification is enabled, it may be inactive for other reasons, such as Bluetooth being turned off. The `exposureNotificationStatus` property can be monitored for the overall status of Exposure Notification.

- (void)setExposureNotificationEnabled:(BOOL)enabled completionHandler:(ENErrorHandler)completionHandler;

Enables or disables Exposure Notification. If not previously authorized, this prompts the user for consent to enable Exposure Notification.

Note: Disabling stops Bluetooth advertising and scanning related to Exposure Notification, but the Diagnosis Keys and data remain.

typedef void (^ENGetDiagnosisKeysHandler)(NSArray <ENTemporaryExposureKey *> * _Nullable keys, NSError * _Nullable error);

The completion handler that is invoked when `getDiagnosisKeysWithCompletionHandler` completes. If the completion handler completes successfully, `keys` contains the Diagnosis Keys for this device and error is `nil`. If it fails, keys is nil and `error` indicates the reason it failed.

- (void)getDiagnosisKeysWithCompletionHandler:(ENGetDiagnosisKeysHandler)completionHandler;

Requests the Temporary Exposure Keys used by this device to share with a server. Each use of this API will present the user with the system's user interface for authorization.

- (void)resetAllDataWithCompletionHandler:(ENErrorHandler) completionHandler;

Resets diagnosis keys and all data related to Exposure Notification. Each use of this API will present the user with the system’s user interface for authorization.

ENRiskLevel

Overview
An enumeration that represents an estimated risk level.

ENRiskLevelInvalid
Invalid level. Used when risk level isn't available.

ENRiskLevelLowest
The lowest risk.

ENRiskLevelLow
Low risk.

ENRiskLevelLowMedium
Risk between low and medium.

ENRiskLevelMedium
Medium risk.

ENRiskLevelMediumHigh
Risk between medium and high.

ENRiskLevelHigh
High risk.

ENRiskLevelVeryHigh
Very high risk.

ENRiskLevelHighest
Highest risk.

ENExposureDetectionSession

Overview

Performs exposure detection based on previously collected proximity data and keys.

Discussion

Here are the steps to use this class:

1. Create an instance of `ENExposureDetectionSession`.
2. Optionally, set the dispatch queue if you want the completion handler to be invoked on something other than the main queue.
3. Call `activateWithCompletion` to asynchronously prepare the session for use.
4. Wait for the completion handler for `activateWithCompletion` to be invoked with a `nil` error.
5. Call `addDiagnosisKeys` with up to `maxKeyCount` keys.
6. Wait for the completion handler for `addDiagnosisKeys` to be invoked with a `nil` error.
7. Repeat the previous two steps until all keys are provided to the system or an error occurs.
8. Call `finishedDiagnosisKeysWithCompletion`.
9. Wait for the completion handler for `finishedDiagnosisKeysWithCompletion` to be invoked with a `nil` error.

If the user is interested in receiving more detailed info about the matches:

1. Call `getExposureInfoWithMaxCount`. Use a reasonable maximum count, such as 100.
2. Wait for the completion handler for `getExposureInfoWithMaxCount` to be invoked with a `nil` error.
3. If the value of the completion handler's `done` parameter is `NO`, repeat the previous two steps until the value is `YES` or an error occurs.

When the app is done with the session, call `invalidate`.

Note: Exposure detection sessions are rate-limited to prevent abuse.

@property ENAuthorizationStatus authorizationStatus;

This property reports the current authorization status of the app. The user is never prompted. Apps can use this property to preflight authorization in order to determine if the user may be prompted.

@property ENExposureConfiguration configuration;

This property holds the configuration to use for assigning values, weighing risks, and other information.

@property dispatch_queue_t dispatchQueue;

This property holds the dispatch queue to invoke handlers on. The main queue is the default queue.

@property dispatch_block_t invalidationHandler;

This handler is Invoked exactly once when invalidation completes, and is cleared before it's invoked to break retain cycles.

@property NSUInteger maximumKeyCount;

This property contains the maximum number of keys allowed to be outstanding (passed to `addDiagnosisKeys`, but not yet completed). This property's value updates after each operation completes and before the completion handler is invoked. Use this property to throttle the downloading of keys and avoid excessive buffering of keys in memory.

- (void)activateWithCompletionHandler: (ENErrorHandler)completionHandler;

Activates the object to prepare it for use. Properties may not be usable until the completion handler reports success.

- (void)invalidate;

Stops any outstanding operations and invalidates this object.

- (void)addDiagnosisKeys:(NSArray <ENTemporaryExposureKey *> *)keys completion:(ENErrorHandler)completionHandler;

Asynchronously adds the specified keys to the session to allow them to be checked for exposure. Each call to this method must not include more keys than specified by the current value of `maxKeyCount`.

typedef void (^ENExposureDetectionFinishCompletion) ENExposureDetectionSummary * _Nullable summary, NSError * _Nullable error);

The type definition for the completion handler that's invoked when exposure detection finishes.

- (void)finishedDiagnosisKeysWithCompletion: (ENExposureDetectionFinishCompletion)completionHandler;

Indicates that all of the available keys have been provided. Any remaining detection is performed and the completion handler is invoked with the results.

typedef void (^ENGetExposureInfoCompletion)(NSArray <ENExposureInfo *> * _Nullable exposures, BOOL done, NSError * _Nullable error);

The type definition for the completion handler that's invoked when an exposure information request completes.

- (void)getExposureInfoWithMaxCount:(NSUInteger)maximumCount completionHandler:(ENGetExposureInfoCompletion) completionHandler;

Gets info about each exposure. Plans to ensure user transparency in the use of this function are currently being evaluated. This method can only be called after the detector finishes.

The `maximumCount` parameter indicates the maximum number of exposures the caller is prepared to obtain in a single call. Using this parameter helps you avoid excessive buffering by limiting the number of results.

The `done` parameter to the handler indicates whether more exposure incidents are available.

The handler may provide fewer exposures than requested, even if there are more available, due to memory constraints.

ENExposureDetectionSummary

Overview

Provides a summary of exposures.

@property NSInteger daysSinceLastExposure;

Number of days since the most recent exposure: 0 = today, 1 = yesterday, and so on. Only valid if `matchedKeyCount` > 0.

@property uint64_t matchedKeyCount;

This property holds the number of keys that matched for an exposure detection.

@property ENRiskScore maximumRiskScore;

This property holds the highest transmission risk level of all exposure incidents.

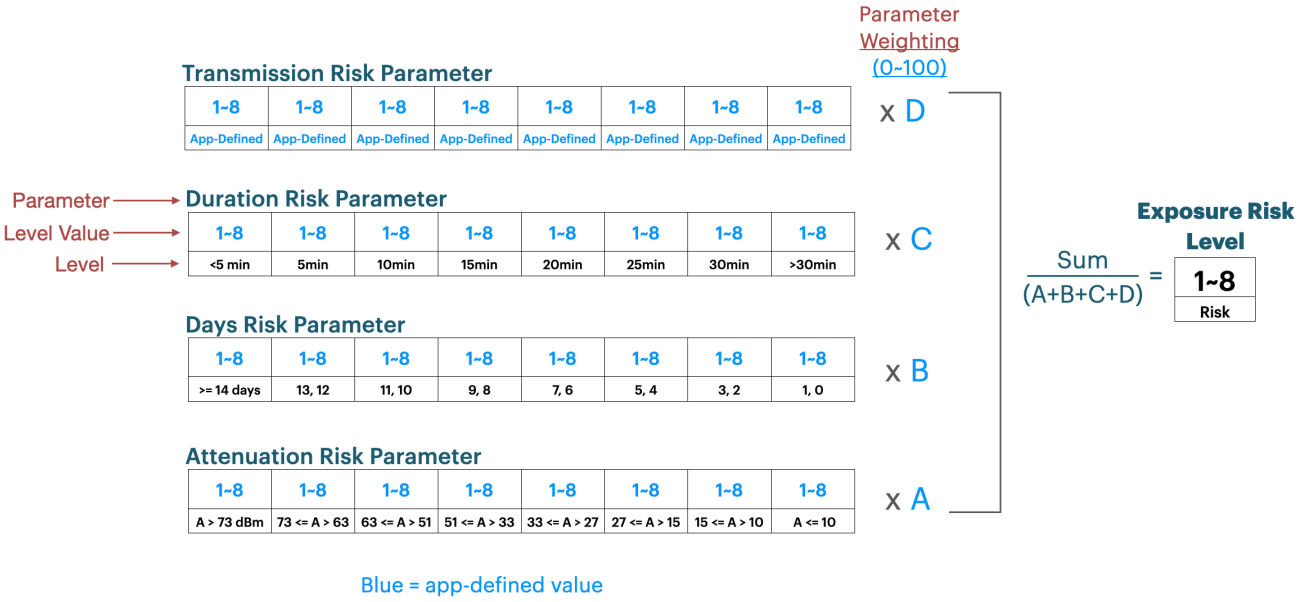
ENExposureConfiguration

Overview

Contains configuration parameters for exposure detection.

Discussion

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



The terms used in this data structure are defined as follows:

Exposure Risk Level Parameters

The four parameters used to calculate the Exposure Risk Level are:

Transmission Risk — An app-defined flexible value to tag a specific positive key. This value could be tagged based on symptoms, level of diagnosis verification, or other determination from the app or a health authority.

Duration (measured by API) — Cumulative duration of the exposure.

Days (measured by API) - Days since the exposure incident.

Attenuation (measured by API) - Minimum Bluetooth signal strength attenuation (Transmission Power subtract RSSI).

Level Value

The value, ranging from 1 to 8, that the app assigns to each Level in each of the Exposure Risk Level Parameters.

Level

The eight levels contained within each Exposure Risk Level Parameter.

Exposure Risk Level Parameter Weights (A, B, C, D)
The weights defined by the app (ranging from 0-100) that assign the relative importance to each of the Exposure Risk Level Parameters.

Exposure Risk Level
An integer that represents the result of the exposure risk level calculation from 1-8. It is the maximum score per person over a 24-hour period. The formula for calculating the Exposure Risk Level is:

WeightedSum = (attenuationScore * attenuationWeight) + (daysScore * daysSinceLastExposureWeight) + (durationScore * durationWeight) + (transmissionRiskScore * transmissionRiskWeight)

totalRiskScore = WeightedSum / (attenuationWeight + daysSinceLastExposureWeight + durationWeight + transmissionRiskWeight)

@property ENRiskScore minimumRiskScore;
This property holds the minimum risk level. It excludes exposure incidents with scores lower than this. The default is no minimum.

@property NSArray <NSNumber *> *attenuationScores;
This property holds values for “attenuation” buckets. It must contain 8 values, one for each bucket, as defined here:

- 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.

@property double attenuationWeight;
This property holds the weight to apply to the “attenuation” value. It must be in the range 0-100.

@property NSArray <NSNumber *> *daysSinceLastExposureScores;
This property holds values for the “days since last exposure” buckets. It must contain 8 values, one for each bucket, as defined here:

- 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.

@property double daysSinceLastExposureWeight;

This property holds the weight to apply to the “days since last exposure” value. It must be in the range 0-100.

@property NSArray <NSNumber *> *durationScores;

This property holds values for the “duration” buckets. It must contain 8 values, one for each bucket, as defined here:

- durationScores[0] when Duration equals 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.

@property double durationWeight;

This property holds the weight to apply to the “duration” value. It must be in the range 0-100.

@property NSArray <NSNumber *> *transmissionRiskScores;

This property holds values for the “transmission risk” buckets. It must contain 8 values, one for each bucket, as defined here:

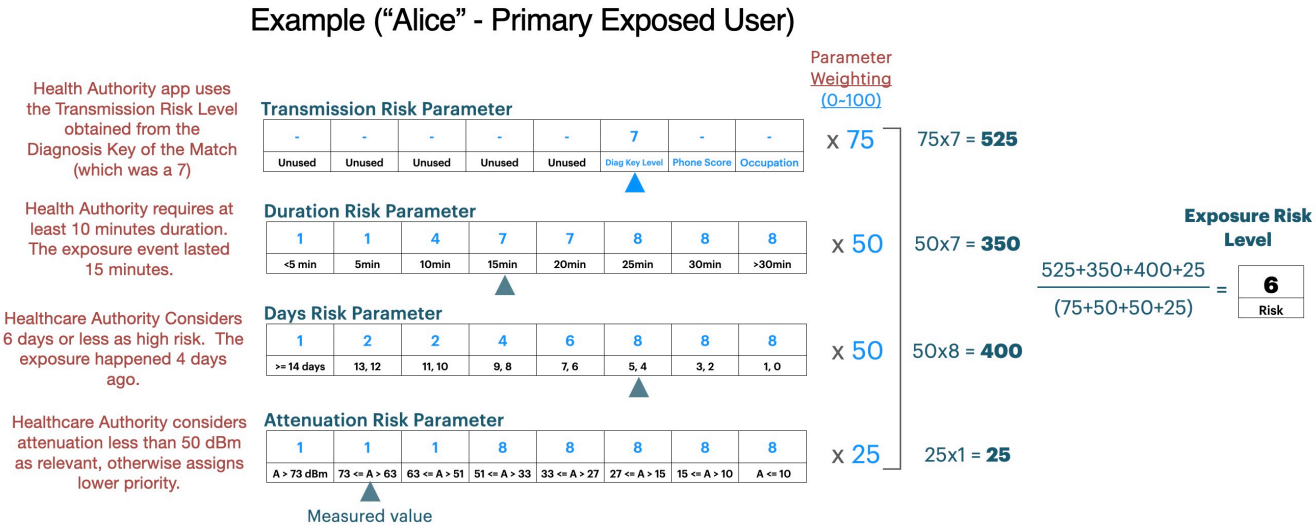
- transmissionRiskScores[0] for ENRiskLevelLowest.
- transmissionRiskScores[1] for ENRiskLevelLow.
- transmissionRiskScores[2] for ENRiskLevelLowMedium.
- transmissionRiskScores[3] for ENRiskLevelMedium.
- transmissionRiskScores[4] for ENRiskLevelMediumHigh.
- transmissionRiskScores[5] for ENRiskLevelHigh.
- transmissionRiskScores[6] for ENRiskLevelVeryHigh.
- transmissionRiskScores[7] for ENRiskLevelHighest.

@property double transmissionRiskWeight;

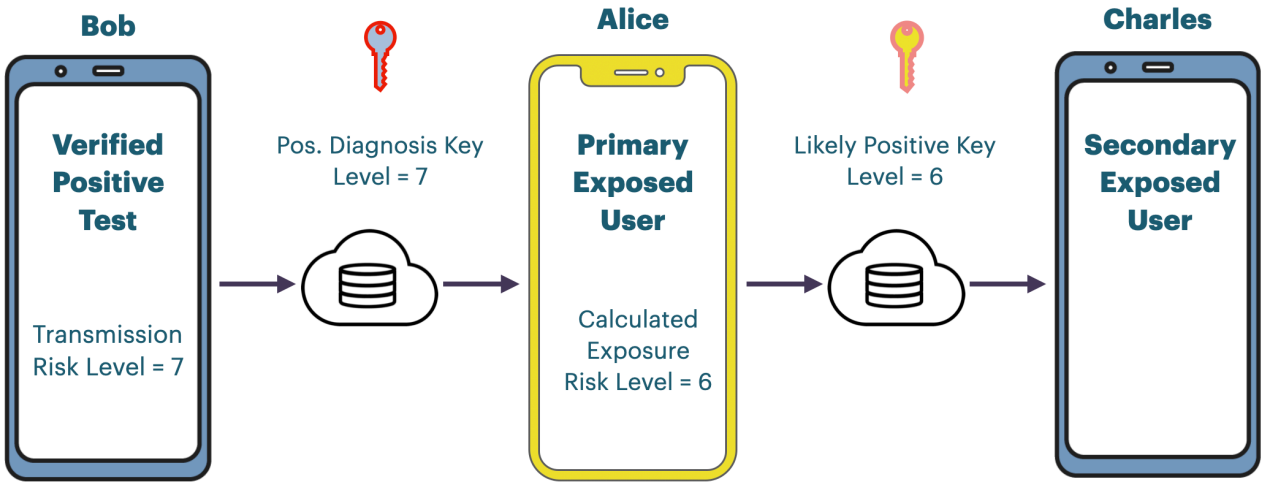
This property holds the weight to apply to the “transmission risk” value. It must be in the range 0-100.

Exposure Risk Level Calculation Example

The following diagram illustrates an example of an Exposure Risk Level calculated for a person named Alice who was exposed to person (Bob) who was diagnosed positive. So, Alice is considered a primary exposed user.



The following diagram shows how Exposure Risk Levels can be transferred along with the Diagnosis Keys via the key-sharing schema. In this way, a secondary exposed user (Charles) who was recently near Alice, may optionally be notified by the app (should it determine that Alice's exposure risk level is high enough that she is likely COVID-19 positive).



ENExposureInfo

Overview

Contains information about a single contact incident.

A way to provide the duration at high and low attenuation values is being evaluated.

@property ENAttenuation attenuationValue;

This property holds the attenuation value of the peer device at the time the exposure occurred. The attenuation is the Reported Transmit Power - Measured RSSI.

@property NSDate *date;

This property holds the date when the exposure occurred. The date may have reduced precision, such as within one day of the actual time.

@property NSTimeInterval duration;

This property holds the duration (in minutes) that the contact was in proximity of the user. The minimum duration is 5 minutes. Other valid values are 10, 15, 20, 25, and 30. The duration is capped at 30 minutes.

@property ENRiskScore totalRiskScore;

This property indicates the exposure risk level calculated for this user in relation to the exposure incident. Note that this value may be negative.

@property ENRiskLevel transmissionRiskLevel;

This property indicates the transmission risk level associated with the diagnosis key.

ENTemporaryExposureKey

Overview

The key used to generate Rolling Proximity Identifiers.

@property NSData *keyData;

This property contains the Temporary Exposure Key information.

@property ENIntervalNumber rollingStartNumber;

This property contains the interval number when the key's `TKRollingPeriod` started.

@property ENRiskLevel transmissionRiskLevel;

This property contains the risk of transmission associated with the person this key came from.

Revision History

v1.2 - April 29, 2020

- Added explanations and examples related to Exposure Risk Level calculations in the `ENExposureConfiguration` section
- Added `ENErrorCodeRateLimited` to `ENErrorDomain`.
- Added `ENRiskLevel` enumeration.
- Added `transmissionRiskLevel` to `ENTemporaryExposureKey`.
- Renamed `ENExposureDetectionGetExposureInfoCompletion` to `ENGetExposureInfoCompletion`.
- Added new properties to `ENExposureDetectionSession` and `ENExposureDetectionSummary`.
- Added documentation for `ENExposureConfiguration`, `ENStatus`, and `ENManager`.

v1.1 - April 23, 2020

- Renamed “Daily Tracing Keys” to “Temporary Exposure Keys.”
- Renamed the framework from “Contact Tracing” to “Exposure Notification,, along with all related terminology.
- Added `ENIntervalNumber` to indicate when “Temporary Exposure Key” started.
- Clarified that `ENSelfExposureInfoRequest` returns the previous 14 days of keys.
- Added ability to filter by `attenuationThreshold` and `durationThreshold` in `ENExposureDetectionSession`.
- Added `daysSinceLastExposure` to `ENExposureDetectionSummary`.
- Added `attenuationValue` to `ENExposureInfo`, and modified the maximum duration value to 30 minutes.
- Added `ENSelfExposureResetRequest` to reset notification history and keys.
- Added a note about evaluating ways of ensuring user transparency when `getExposureInfoWithMaxCount` is invoked for more details on an exposure event.
- Reformatted the title page and table of contents for consistency across documents.