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 by the 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.
ENAuthorizationMode

Overview
An enumeration that specifies the app's preference for authorization with Exposure Notification.

ENAuthorizationModeDefault
Let the system choose whether to prompt. This is the best option unless you have specific needs. For most cases, it will prompt the user to authorize (same as ENAuthorizationModeUI). This gives the system some flexibility, such as not to prompt if the app is in the background.

ENAuthorizationModeNonUI
Authorization will be checked, but it won't prompt the user if the app isn't authorized. If the app is authorized, the app is allowed to use the service. If the user hasn't been prompted or they denied the app, operations will fail with ENErrorCodeNotAuthorized.

ENAuthorizationModeUI
Authorization will be checked and it will prompt the user to authorize, if needed. If the app is authorized by the user, the app is allowed to use the service. If the user denies the app, operations will fail with ENErrorCodeNotAuthorized.
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.
ENActivatable

Overview
A protocol for objects that support asynchronous operations and cancellation.

Discussion
Classes conforming to this protocol follow a similar life-cycle pattern:

1. Create an instance of the class.
2. Set properties to configure it, if needed.
3. Call `activateWithCompletion`.
4. Wait for the activate completion handler to be invoked.
5. Use methods and properties of the class.
6. Call `invalidate` to stop any outstanding operations, release resources, and break any retain cycles.
7. Upon completion of invalidation, invoke an `invalidationHandler` if it was specified before activating.
8. Release the instance.

If `activateWithCompletion` is called, `invalidate` must always be called before the object is released.

**Note:** Invalidation is asynchronous, so it’s possible for handlers to be invoked after calling `invalidate`. When invalidation is finished, the `invalidationHandler` is invoked exactly once, even if `invalidate` is called multiple times. No handlers are invoked after that. 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 finishes, in order to break potential retain cycles. You don’t need to use weak references within your handlers to avoid retain cycles when using this protocol.

@property dispatch_queue_t dispatchQueue;
This property holds the dispatch queue used to invoke completion handlers. The default is the main queue.

@property dispatch_block_t invalidationHandler;
This property holds the invalidation handler. It is invoked exactly once: when invalidation completes. This property is cleared before it’s invoked in order to break retain cycles.

- (void)activateWithCompletion:(ENErrorHandler) inCompletion;
Activates the object to prepare it for use. Properties may not be used until the completion handler reports success.

- (void)invalidate;
Stops any outstanding operations and invalidates the object.
ENAuthorizable

Overview
A protocol for objects that require authorization from the user before they can be used.

@property ENAuthorizationStatus authorizationStatus;
This property reports the current authorization status of the app and never prompts the user. It can be used by the app for preflight authorization to determine if the user may be prompted.

@property ENAuthorizationMode authorizationMode;
This property specifies the app's preference for authorization with Exposure Notification. It defaults to prompting the user to authorize, if needed.
ENSettings

Overview
Defines nonmodifiable settings for Exposure Notification.

@property ENMultiState enableState;
This property turns Exposure Notification on or off.

- (ENMutableSettings *)mutableCopy;
Returns a mutable copy of this object.
ENMutableSettings

Overview
Defines modifiable settings for Exposure Notification.

Discussion
Use with ENSettingsChangeRequest to apply changed settings to the system. All settings have default values that represent "not changed" to allow creating an empty object and setting only the properties that need to be changed, without affecting other properties.

@property ENMultiState enableState;
This property turns Exposure Notification on or off.

- (ENSettings *)copy;
Returns an immutable copy of this object.
ENSettingsGetRequest

Overview
Requests the current settings for Exposure Notification.

Discussion
Use this class as follows:

1. Create an instance of ENSettingsGetRequest.
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 get settings from the system.
4. When the activation completes successfully, the settings property is valid to access.
5. Access the needed information in the settings object. You may retain it for use after invalidating the request, if needed.
6. Call invalidate.

@property ENSettings *settings;
This property is set with a snapshot of the settings if activation completes successfully.
ENSettingsChangeRequest

Overview
Changes settings for Exposure Notification after authorization by the user.

Discussion
Use this class as follows:

1. Create an instance of ETMutableSettings (or call mutableCopy on an immutable ENSettings instance).
2. Set properties of the settings instance to change any settings.
3. Create an instance of ENSettingsChangeRequest.
4. Set the settings property to the settings instance.
5. Optionally set the dispatch queue if you want the completion handler to be invoked on something other than the main queue.
6. Call activateWithCompletion to asynchronously change settings if authorized by the user. Authorization happens when activateWithCompletion is called. The user may be prompted at this point.
7. When the activation completes successfully, the new settings are applied.
8. Call invalidate.

@property ENSettings *settings;
This property specifies the settings to change. It must be set before calling activateWithCompletion.
ENExposureDetectionSession

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

Discussion
Exposure detection sessions are rate-limited to prevent abuse.

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 summary indicates matches were found, notify the user of exposure. If the user is interested in sharing more details with the app:

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 inDone 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.

@property uint8_t attenuationThreshold;
This property holds the largest amount of signal attenuation allowable when detecting matches. A value of 0 means there's no limit.

Attenuation is calculated by subtracting the measured RSSI from the reported transmit power. Values larger than the attenuation limit aren't checked for exposure matches.

Note: The attenuation value may be misleading because more attenuation doesn't necessarily mean the device is farther away. For example, two people could be physically very close and facing each other with their phones in their back pockets. In this case, a higher attenuation (a weaker signal) may be reported even though the individuals are very close together.
@property NSTimeInterval durationThreshold;
This property holds the minimum exposure duration to be considered an exposure incident. If a detected exposure duration is less than the value of this property, it’s excluded from the results. The default is 0, indicating no filtering by duration.

@property NSInteger maxKeyCount;
This property contains the maximum number of keys to provide to this API at a time. 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)addDiagnosisKeys:(NSArray <ENTemporaryExposureKey *> *)inKeys completion:(ENErrorHandler)inCompletion;
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 inSummary, NSError * _Nullable inError);
The type definition for the completion handler that’s invoked when exposure detection finishes.

- (void)finishedDiagnosisKeysWithCompletion: (ENExposureDetectionFinishCompletion)inCompletion;
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 (^ENExposureDetectionGetExposureInfoCompletion) (NSArray <ENExposureInfo *> * _Nullable inExposures, BOOL inDone, NSError * _Nullable inError);
The type definition for the completion handler that’s invoked when an exposure information request completes.

- (void)getExposureInfoWithMaxCount:(uint32_t)inMaxCount completion: (ENExposureDetectionGetExposureInfoCompletion) inCompletion;
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 inMaxCount 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 inDone parameter to the handler indicates whether more exposure incidents are available.

The handler may be invoked multiple times. An empty array indicates the final invocation of the handler.
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.
ENSelfExposureInfoRequest

Overview
Requests the Temporary Exposure Keys used by this device to share with a server.

Discussion
This request is intended to be called when a user has received a positive diagnosis. Once the keys are shared with a server, other users can use the keys to check if their device has been in close proximity with any positively diagnosed users, enough to cause an exposure incident. Each request results in the user being notified by the operating system.

Keys are reported for the previous 14 days of exposure notification. The returned keys are at least 24 hours old.

The app must have previously enabled Exposure Notification through the settings API (which requires approval by the user). If the app hasn’t done that, this request fails with ENErrorCodeNotEnabled.

How to use this class:

1. Create an instance of ENSelfExposureInfoRequest.

2. Optionally set the dispatch queue if you want the completion to be invoked on something other than the main queue.

3. Call activateWithCompletion to asynchronously start the request.

4. When the activation completes successfully, the selfExposureInfo property is valid to access.

5. Access information in the selfExposureInfo object. You may retain it for use after invalidating the request, if needed.

6. Call invalidate.

@property ENSelfExposureInfo selfExposureInfo;
This property is set to a snapshot of the results if activation completes successfully.
ENSelfExposureResetRequest

Overview
Deletes all collected exposure data and Temporary Exposure Keys.

Discussion
Note: This object eliminates the ability to detect exposure that may have occurred before the point of reset. Each request prompts the user to authorize the request.
ENExposureInfo

Overview
Contains information about a single contact incident.

@property uint8_t 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.
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.
Revision History
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.
- Clarify that ENSelfExposureInfoRequest returns the previous 14 days of key.
- 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 exposure event.
- Reformatted the title page and table of contents for consistency across documents.