Contact Tracing

Framework Documentation

(API)

Preliminary - Subject to Modification and Extension

April 2020

Contents

Overview

The ContactTracing Framework is designed to help developers implement a privacy-preserving contact tracing solution. It covers two user roles:

1. *Affected User*. A user who reports themself as positively diagnosed as having the virus.
2. *Exposed User*. A user who has notified themself as potentially exposed to an Affected User.

Affected User

When a user is positively affected, their Daily Tracing Keys should be shared with other uses to alert them to potential exposure. These Daily Tracing Keys are retrieved using `CTSelfTracingInfoRequest`.

Exposed User

Given a set of positively affected Daily Tracing Keys, the framework allows you to determine whether those Daily Tracing Keys were observed locally by the user, indicating potential exposure. If so, additional information such as date and duration may also be retrieved. Possible observations can be retrieved using `CTExposureDetectionFinishHandler`, and additional information using `CTExposureDetectionContactHandler`.

The following illustration outlines the flow of the ContactTracing Framework for iOS.

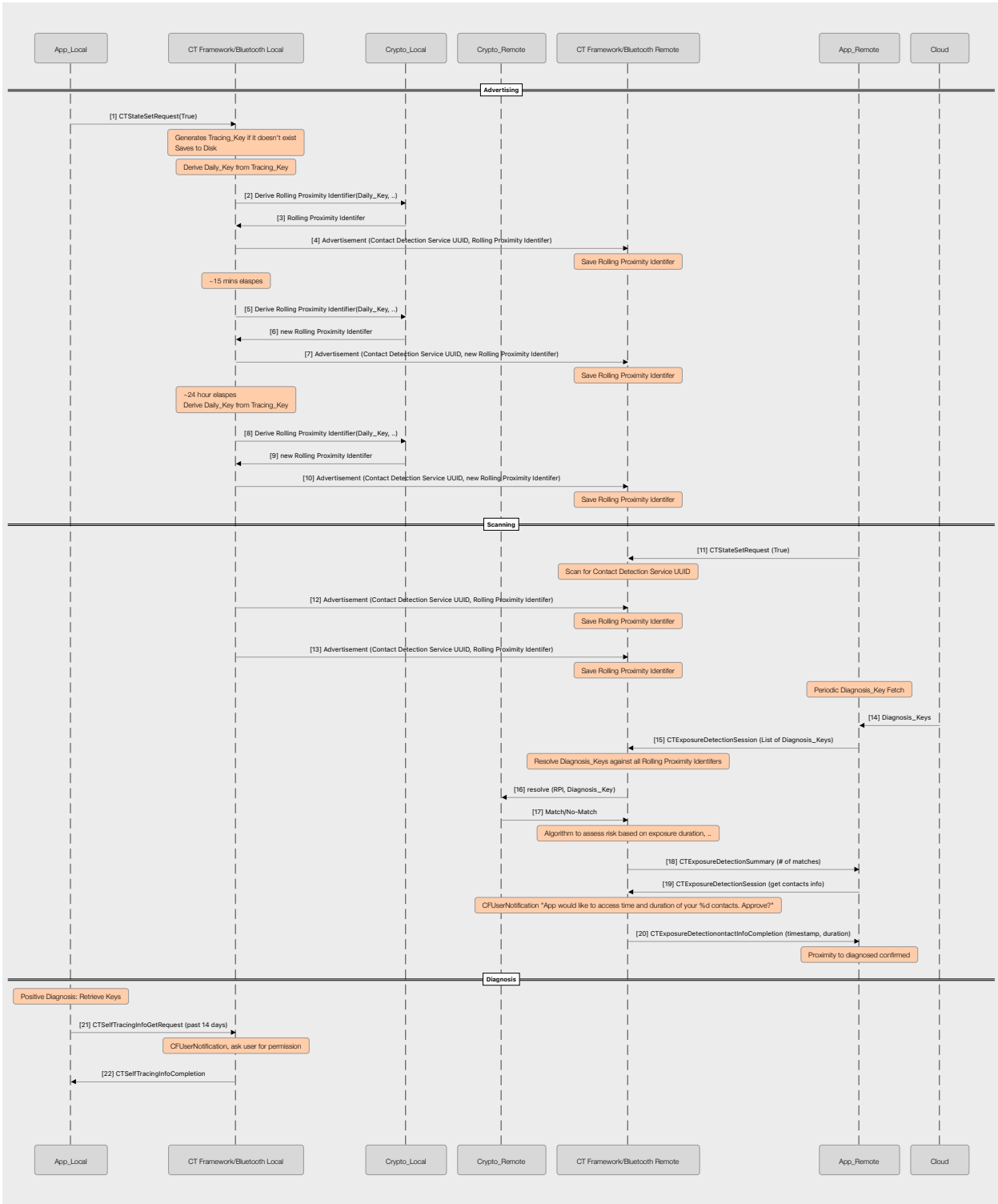

Figure 1. The Contact Tracing Flow

CTStateGetRequest

Overview

Requests whether contact tracing is on or off on the device.

typedef void (^CTErrorHandler)(NSError * _Nullable inError);

The type definition for the completion handler.

@property CTErrorHandler completionHandler;

This property holds the completion handler that framework invokes when the request completes. The property is cleared upon completion to break any potential retain cycles.

@property dispatch_queue_t dispatchQueue;

This property holds the the dispatch queue used to invoke handlers on. If this property isn’t set, the framework uses the main queue.

@property CTManagerState state;

This property contains the snapshot of the state when the request was performed. It’s valid only after the framework invokes the completion handler.

- (void)perform;

Asynchronously performs the request to get the state, and invokes the completion handler when it's done.

- (void)invalidate;

Invalidates a previously initiated request. If there is an outstanding completion handler, the framework will invoke it with an error.

Don’t reuse the request after this is called. If you require another request, create a new one.

CTStateSetRequest

Overview

Changes the state of contact tracing on the device.

@property CTErrorHandler completionHandler;

This property holds the completion handler that framework invokes when the request completes. The property is cleared upon completion to break any potential retain cycles.

@property dispatch_queue_t dispatchQueue;

This property holds the the dispatch queue used to invoke handlers on. If this property isn’t set, the framework uses the main queue.

@property CTManagerState state;

This property contains the state to set Contact Tracing to. Call the `perform` method to apply the state once set.

- (void)perform;

Asynchronously performs the request to get the state, and invokes the completion handler when it's done.

- (void)invalidate;

Invalidates a previously initiated request. If there is an outstanding completion handler, the framework will invoke it with an error.

Don’t reuse the request after this is called. If you require another request, create a new one.

CTExposureDetectionSession

Overview

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

@property dispatch_queue_t dispatchQueue;

This property holds the the dispatch queue used to invoke handlers on. If this property isn't set, the framework uses the main queue.

@property NSInteger maxKeyCount;

This property contains the maximum number of keys to provide to this API at once. This property's value updates after each operation complete and before the completion handler is invoked. Use this property to throttle key downloads to avoid excessive buffering of keys in memory.

- (void) activateWithCompletion:(CTErrorHandler) inCompletion;

Activates the session and requests authorization for the app with the user. Properties and methods cannot be used until this completes successfully.

- (void)invalidate;

Invalidates the session. Any outstanding completion handlers will be invoked with an error. The session cannot be used after this is called. A new session must be created if another detection is needed.

- (void) addPositiveDiagnosisKeys:(NSArray <CTDailyTracingKey *> *) inKeys completion:(CTErrorHandler) inCompletion;

Asynchronously adds the specified keys to the session to allow them to be checked for exposure. Each call to this method must include more keys than specified by the current value of <maxKeyCount>.

typedef void (^CTExposureDetectionFinishHandler) (CTExposureDetectionSummary * _Nullable inSummary, NSError * _Nullable inError);

The type definition for the completion handler.

- (void) finishedPositiveDiagnosisKeysWithCompletion: (CTExposureDetectionFinishHandler) inFinishHandler;

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

typedef void (^CTExposureDetectionContactHandler)(NSArray <CTContactInfo *> * _Nullable inContacts, NSError * _Nullable inError);

The type definition for the completion handler.

- (void)getContactInfoWithHandler: (CTExposureDetectionContactHandler) inHandler;

Obtains information on each incident. This can only be called once the detector finishes. The handler may be invoked multiple times. An empty array indicates the final invocation of the hander.

CTExposureDetectionSummary

Overview

Provides a summary on exposures.

@property NSInteger matchedKeyCount;

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

CTSelfTracingInfoRequest

Overview

Requests the daily tracing keys used by this device to share with a server.

Discussion

This request is intended to be called when a user has 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 contact with any positive diagnosis users. Each request will require the user to authorize access.

Keys will be reported for the previous 14 days of contact tracing. The app will also be launched every day after the daily tracing key changes to allow it to request again to get the key for each previous day for the next 14 days.

typedef void (^CTSelfTracingInfoGetCompletion) (CTSelfTracingInfo * _Nullable inInfo, NSError * _Nullable inError);

The type definition for the completion handler.

@property CTSelfTracingInfoGetCompletion completionHandler;

This property invokes this completion handler when the request completes and clears the property to break any potential retain cycles.

@property dispatch_queue_t dispatchQueue;

This property holds the the dispatch queue used to invoke handlers on. If this property isn't set, the framework uses the main queue.

- (void)perform;

Asynchronously performs the request to get the state, and invokes the completion handler when it's done.

- (void)invalidate;

Invalidates a previously initiated request. If there is an outstanding completion handler, the framework will invoke it with an error.

Don't reuse the request after this is called. If you require another request, create a new one.

CTSelfTracingInfo

Overview

Contains the Daily Tracing Keys.

@property NSArray <CTDailyTracingKey *> * dailyTracingKeys;

Daily tracing keys available at the time of the request.

CTContactInfo

Overview

Contains information about a single contact incident.

@property NSTimeInterval duration;

How long the contact was in proximity. Minimum duration is 5 minutes and increments by 5 minutes: 5, 10, 15, etc.

@property CFAbsoluteTime timestamp;

This property contains the time when the contact occurred. This may have reduced precision, such as within one day of the actual time.

CTDailyTracingKey

Overview

The Daily Tracing Key object.

@property NSData *keyData;

This property contains the Daily Tracing Key information.