MagicFIDO iOS SDK 개발가이드

Version 1.3.1

DREAM SECURITY
Digital Trust

제.개정 이력

개정번호	내용	개정일자
1.3.1	setAuthenticatorOptions API 설정 값 추가 - shuffleDisable	2020.02.10
1.3.0	Authentication 클래스 API 추가 - setDivideAuthStep - resumeAuthentication	2019.08.23
	FidoResult Code 추가 - RESULT_AUTH_READY_SUCCESS - ERROR_AUTH_REQUEST_NOT_EXIST	
1.2.7	MagicFidoUtil 클래스 API 추가 - setPasscodeRegularExpressionList	2019.04.25
	setAuthenticatorOptions API 설정 값 추가 - maxLockCount	
	FidoResult Code 추가 - ERROR_MAX_LOCK_COUNT_OVER - IGNORE_MAX_LOCK_COUNT_SET	
1.2.6	MagicFidoUtil 클래스 API 추가 - setPasscodeIgnoreList	2019.04.08
	인증장치 상태값 관련 FidoResult 명칭 변경 (TouchID -> Biometry)	
1.2.5	MagicFidoUtil 클래스 API 추가 - setKeypadAccessibilityOnSpeakerEnable	2019.03.26
1.2.4	MagicFidoUtil 클래스 API 추가 - setPasscodeUIType	2019.03.18
1.2.3	FidoResult Code 추가 작성 및 수정	2019.02.20
1.2.2	MagicFidoUtil 클래스 API 변경 - setAuthenticatorResetCallbackEnable 변경	2019.02.18
	ContextKey 안내 추가	
1.2.1	패스코드 옵션설정 키값 추가 - KEY_REGULAR_EXPRESSION	2018.12.10
	MagicFidoUtil 클래스 API 변경 - setAuthenticatorResetCallbackEnable 추가 - setPasscodeMaxInputAutoNextStepEnable 추가	
	Authentication 클래스 API 변경 - setBioVerifyStateCheckEnable 추가	
	인증장치 화면 UI 추가	

1.2.0	MagicFidoUtil 클래스 API 변경 - getVersion 추가	2018.11.20
1.0.9	패스코드 옵션 관련 키값 추가	2018.09.18
1.0.8	Deregistration 설정 값 추가에 따른 API 추가/변경	2018.07.31
	MagicFidoUtil 클래스 변경 - setPatternStyle 삭제 - setAuthenticatorOptions 추가	2018.07.31'
1.0.7	라이브러리 적용 가이드 추가	2018.05.08
1.0.6	MagicFidoUtil Class 함수 추가	2018.02.09
	FidoResult 추가	2018.02.09
	등록절차 수정	2018.02.09
1.0.5	MagicFidoUtil 클래스 추가 - Registration/Authentication/Deregistration 의 isAvailableFIDO 와 getTouchIDState 를 MagicFidoUtil 로 통합 - SSL 검증 활성화/비활성화 API 추가	2017.11.24
1.0.4	에러코드 추가 및 isAvailableFIDO() 리턴 값 추가	2017.04.12
1.0.3	결과코드에 대한 유의사항 작성	2017.04.04
1.0.1	ResultCode 추가	2017.01.16
1.0.0	초안 작성	2016.12.02

목 차

1. 개요

1.1 목적

본 문서는 FIDO 기술규격을 준수하여 InApp 방식으로 FIDO 인증기술을 적용할 수 있는 방법을 기술한다.

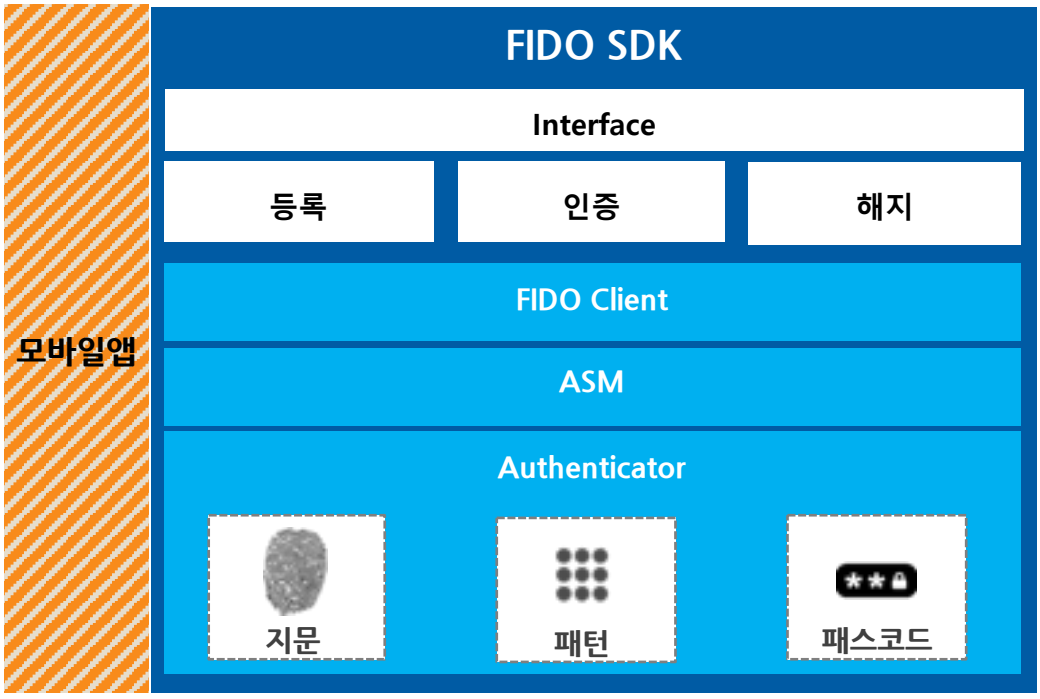
1.2 문서의 구성 및 범위

본 문서는 다음과 같은 사항에 대해 기술하며, 특별히 규정하지 않은 사항에 대해서는 관련 규격을 따른다.

- 패키지 구성 및 설치방법
- API 개발 가이드
- 사용예제

1.3 FIDO 모듈구성

FIDO 모듈은 FIDO SDK 로 구성되며 FIDO 를 적용할 모바일앱은 제공된 SDK 를 이용하여 인증장치 등록, 인증, 인증장치 해지를 간단하게 구현할 수 있다.



[그림 1] FIDO SDK 모듈구성

1.4 서비스 흐름도

FIDO 를 적용하려는 모바일앱에서 FIDO 시스템을 이용해 인증을 수행하려면 FacetID 를 서버에 등록해야 한다.
서버에 FacetID 를 정상적으로 등록했으면 FIDO 를 통해서 인증을 수행할 수 있으며, 사용자가 더 이상 FIDO 를 사용하지 않을 경우 제공된 API 를 통해서 인증장치를 해제할 수 있다.

1.4.1 FacetID 등록 방법

FIDO 를 적용하려는 앱의 번들아이디를 FIDO 규격에 맞춰서 제공사에 전달한다.

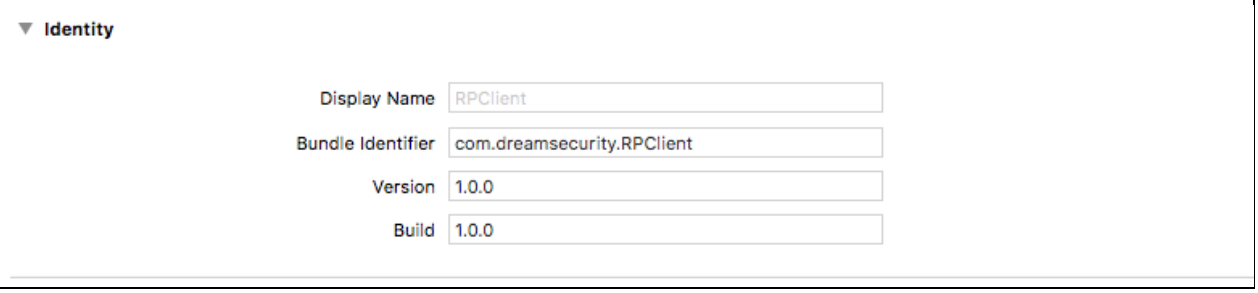
Ios:bundle-id:번들아이디

번들 아이디(NSString)를 획득하는 방법은 아래와 같다.

- 소스에서 구하는 방법.

[[NSBundle mainBundle] bundleIdentifier]

- 프로젝트내에서 확인하는 방법.



2. 개발환경 설정

2.1 패키지 구성

[FIDO 를 사용 하기 위한 패키지 구성]

구 조	파일명	설 명
Doc	FIDO iOS_SDK 개발가이드_vx.x.x.pdf	본 문서
Libs	Libraries	라이브러리 폴더
	Headers	헤더 폴더
	Resources	리소스 폴더
Sample	RPClient	FIDO 연동 샘플 RP Client 프로젝트

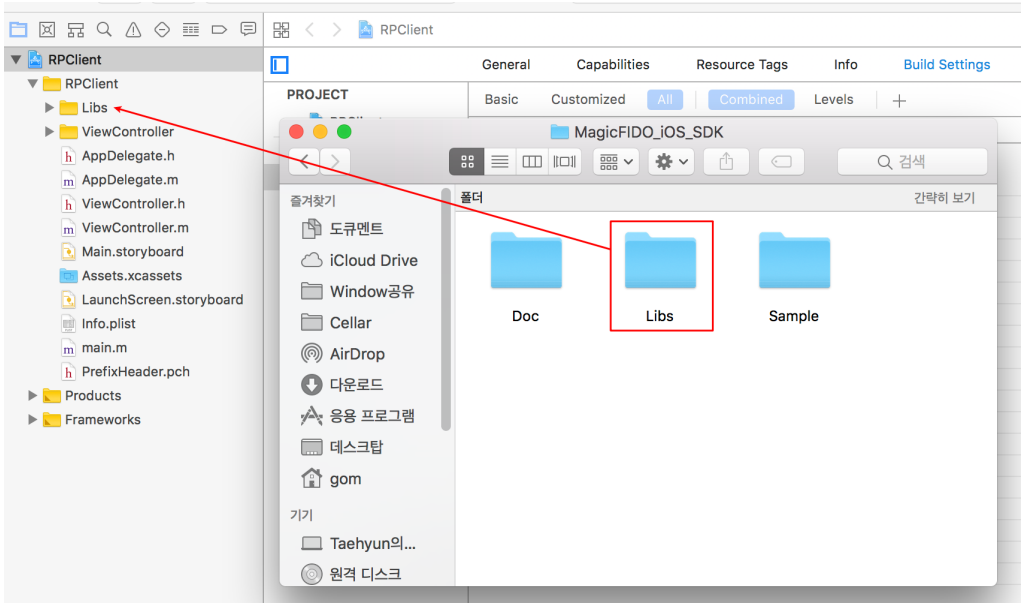
2.2 라이브러리 프로젝트 설정

[FIDO 라이브러리 적용 가이드]

- **라이브러리를 프로젝트에 추가**

제공되는 라이브러리의 Libs 폴더를 3rd party 프로젝트에 추가한다.

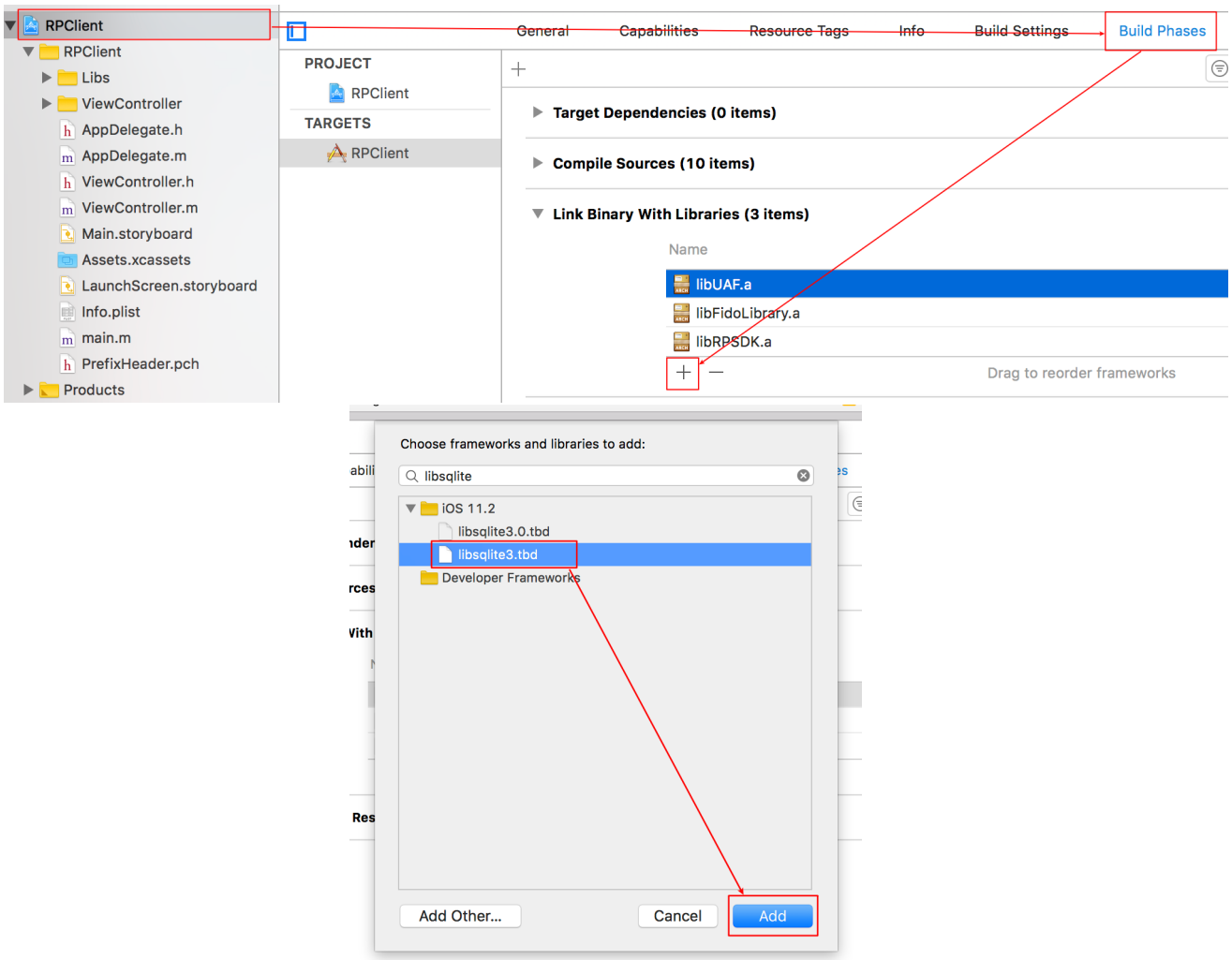
프로젝트에 모든 라이브러리 파일을 복사



[그림 1] Fido 라이브러리 추가

- **기타 라이브러리 추가**

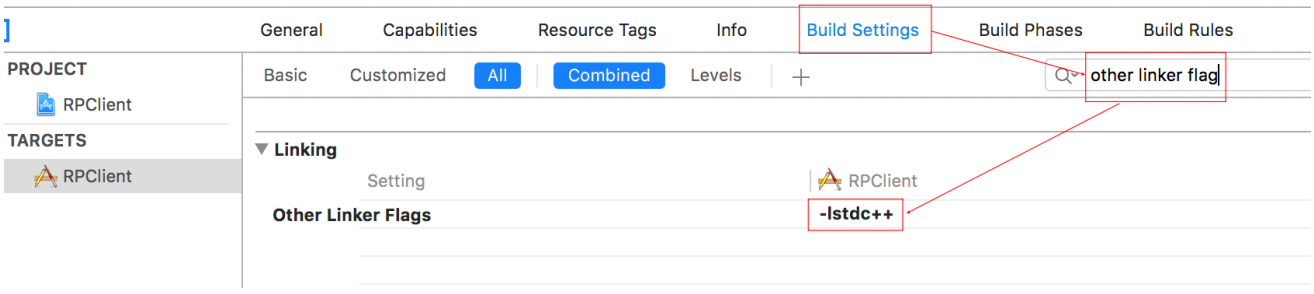
프로젝트 설정 -> Build Phases -> Link Binary With Libraries 로 이동하여 libsqlite3.tbd 파일을 추가한다.



[그림 2] libsqlite 라이브러리 추가

- **Other Linker Flag 추가**

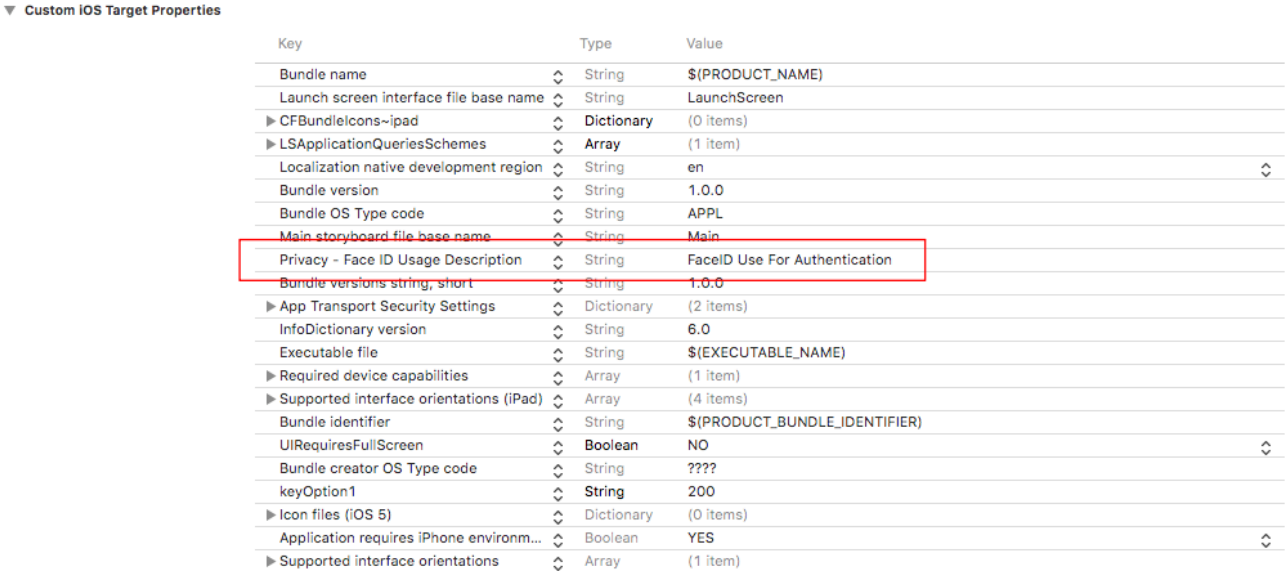
프로젝트 설정 -> Build Settings 로 이동하여 Other Linker Flags 에 –lstdc++을 추가한다.



[그림 3] Other Linker Flag 추가

- **Privacy – Face ID Usage Description (NSFaceIDUsageDescription)필드 추가**

프로젝트 설정 -> Info 로 이동하여 Privacy – Face ID Usage Description 필드를 추가하고 권한 요청 팝업에 쓰일 문구를 적는다.



[그림 4] NSFaceIDUsageDescription 추가

2.3 iOS 지원단말 및 OS

구분	지원 OS
iOS	9.x ~

- 지원 단말 여부에 따른 메뉴 활성화 처리
  FIDO 지원단말 확인은 FIDO Client SDK 에서 제공해주는 isAvailableFIDO() API 를 호출해서 리턴 값이 true 이면 FIDO 메뉴를 활성화 하고 false 가 반환되면 해당 메뉴를 사용자에게 보이지 않도록 구성할 수 있다.

3 API Reference

본 API 는 FIDO 를 이용하기 위한 중요기능인 등록, 인증, 해지 기능을 제공하며 이를 원활하게 사용할 수 있도록 유틸, 에러확인 등의 클래스로 구성되어 있다. 등록, 인증, 해지에서 제공되는 API 는 순자적으로 호출되어야 한다.

3.1 Registration Class

3.1.1 Class Overview

인증장치를 FIDO 서버에 등록하기 위한 메시지 생성 및 분석하는 기능을 제공한다.

3.1.2 Summery

Public Constructors	
id	getInstance 인스턴스를 생성한다.

Public Methods	
void	setReqURL:(NSString *)fidoReqURL WithResURL:(NSString *)fidoResURL WithTimeOut:(int)timeOut FIDO Server 요청 url, 응답 url, 응답대기시간(Default 30 초) 설정한다.
~~BOOL~~	~~isAvailableFIDO~~ ~~삭제~~ ~~사용자의 단말기가 FIDO 를 사용 할 수 있는지 확인 한다.~~
void	startRegistration:(NSMutableDictionary *) contextTable 인증장치 등록 요청 및 진행 한다.
NSString	getToken FIDO 서버로부터 전달 받은 Token 을 획득한다.
~~FidoResult~~	~~getTouchIDState~~ ~~삭제~~ ~~Local 인증을 수행하기 위한 단말기의 상태(사용가능 단말, 불가능 단말, 지문등록 여부)를 확인 할 수 있다~~

3.1.3 Public Constructor

```
+ (id) getInstance;
```

인스턴스를 생성한다.

3.1.4 Public Methods

```
- (void) setReqURL:(NSString *)fidoReqURL WithResURL:(NSString *)fidoResURL
```

WithTimeOut:(int)timeOut;

FIDO Server 요청 url, 응답 url, 응답대기시간(Default 30 초) 설정한다.

Parameters

fidoReqURL 정책 요청 FIDO 서버 URL
fidoResURL 등록 요청 FIDO 서버 URL
timeOut FIDO 서버 접속 및 응답 Timeout

Remarks

startRegistration 을 호출 하기 전에 실행한다. 네트워크 타임아웃을 세팅하지 않을 경우 30 초로 정해진다

- (void) **startRegistration**:(NSMutableDictionary *) contextTable;

인증장치 등록 요청 및 진행 한다.

Parameters

contextTable FIDO 서버에서 사용자를 구분하기 위한 Dictionary (샘플코드 및 개발가이드의 4.1 참고)

Remarks

* 등록 요청 한 결과를 클래스에서 설정한 delegate 를 통해 리턴 받는다.
* 에러코드 ERROR_ALREADY_REG_USER 를 받을 경우, 서버에는 요청한 아이디로 이미 등록 되어 있지만 단말에는 실제 값이 없기 때문에 "FIDO 해지" 진행 후 다시 재등록을 요청해야 한다.
* ContextKey List

키	나타내는 값
< 필수 >	
Key_Context_UserID	사용자 ID
< 옵션 >	
Key_Context_DeviceID	단말기 ID
Key_Context_DeviceModel	단말기 모델명
Key_Context_DeviceOS	단말기 OS
Key_Context_AuthCode	인증 코드

* Delegate 에 대한 유의 사항 (샘플코드 및 개발가이드의 4.4 참고)

- (NSString *) **getToken;**

FIDO 서버로부터 전달 받은 Token 을 획득한다.

Returns

Token

Remarks

인증장치 등록 메시지에 대한 결과가 성공일 경우에 값을 획득 할 수 있다.

3.2 Authentication Class

3.2.1 Class Overview

사용자 인증을 위한 메시지 생성 및 분석하는 기능을 제공한다.

3.2.2 Summery

Public Constructors	
id	getInstance 인스턴스를 생성한다.

Public Methods	
void	setReqURL:(NSString *)fidoReqURL WithResURL:(NSString *)fidoResURL WithTimeOut:(int)timeOut FIDO Server 요청 url, 응답 url, 응답대기시간(Default 30 초) 설정한다.
~~BOOL~~	~~isAvailableFIDO~~ ~~삭제~~ ~~사용자의 단말기가 FIDO 를 사용 할 수 있는지 확인 한다.~~
void	startAuthentication:(NSMutableDictionary *) contextTable 사용자 인증 요청 및 진행 한다.
NSString	getToken 서버로부터 전달 받은 Token 값을 받는다.
~~FidoResult~~	~~getTouchIDState~~ ~~삭제~~ ~~Local 인증을 수행하기 위한 단말기의 상태(사용가능 단말, 불가능 단말, 지문등록 여부)를 확인 할 수 있다.~~
void	setBioVerifyStateCheckEnable:(BOOL)isEnable 생체인증장치의 상태 값 변경에 대한 결과를 받을 수 있도록 설정한다.
void	setDivideAuthStep:(BOOL)isEnable 인증 과정의 단계를 나눠서 진행할 수 있도록 설정한다.
void	resumeAuthentication 인증 과정의 다음 단계를 진행한다.

3.2.3 Public Constructor

+ (id) **getInstance**;

인스턴스를 생성한다.

3.2.4 Public Methods

- (void) **setReqURL**:(NSString *)fidoReqURL **WithResURL**:(NSString *)fidoResURL

WithTimeOut:(int)timeOut;

FIDO Server 요청 url, 응답 url, 응답대기시간(Default 30 초) 설정한다.

Parameters

fidoReqURL 정책 요청 FIDO 서버 URL
fidoResURL 인증 요청 FIDO 서버 URL
timeOut FIDO 서버 접속 및 응답 Timeout

Remarks

startAuthentication 을 호출 하기 전에 실행한다. 네트워크 타임아웃을 세팅하지 않을 경우 30 초로 정해진다

- (void) **startAuthentication**:(NSMutableDictionary *) contextTable

사용자 인증 요청 및 진행 한다.

Parameters

contextTable FIDO 서버에서 사용자를 구분하기 위한 Dictionary (샘플코드 및 개발가이드의 4.2 참고)

Remarks

* 인증 요청 한 결과를 생성자에서 설정 한 delegate 를 통해 리턴 받는다.

* 에러코드 ERROR_NOT_EXIST_USER_DATA(1009) 을 받을 경우, 서버에는 요청한 아이디로 이미 등록 되어 있지만 단말에는 실제 값이 없기 때문에 "FIDO 해지" 진행 후 다시 재등록을 요청해야 한다.

* ContextKey List

키	나타내는 값
< 필수 >	
Key_Context_UserID	사용자 ID
< 옵션 >	
Key_Context_DeviceID	단말기 ID
Key_Context_DeviceModel	단말기 모델명
Key_Context_DeviceOS	단말기 OS
Key_Context_TransactionText	Transaction 확인 텍스트

* Delegate 에 대한 유의 사항 (샘플코드 및 개발가이드의 4.5 참고)

- (NSString *) **getToken;**

FIDO 서버로부터 전달 받은 Token 을 획득한다.

Returns

Token

Remarks

인증장치 인증 메시지에 대한 결과가 성공일 경우에 값을 획득 할 수 있다.

- (void) **setBioVerifyStateCheckEnable**:(BOOL) isEnable

생체인증장치의 상태 값 변경에 대한 결과를 받을 수 있도록 설정한다.

Parameters

isEnable 활성화 여부
활성화 YES / 비활성화 NO

Remarks

생체인증장치(지문,안면 등)의 값이 FIDO 등록 했을 때와 비교하여 추가/변경/삭제 되었을 경우 상태가 변경되었다는 결과값을 받도록 설정 할 수 있다.

- (void) **setDivideAuthStep**:(BOOL) isEnable

인증 과정의 단계를 나눠서 진행할 수 있도록 설정한다.

Parameters

isEnable 활성화 여부
활성화 YES / 비활성화 NO

Remarks

인증 과정을 실제 사용자 인증 화면발생 전/후로 나누어 진행하도록 설정할 수 있다. startAuthentication API 호출 전에 설정해야 하며, 기본값은 Disable.

- (void) **resumeAuthentication**

인증 과정의 다음 단계를 진행한다.

Remarks

setDivideAuthStep 이 Enable 되어 있지 않은 경우 무시되며, startAuthentication API 가 호출된 이후 호출되어야 한다. startAuthentication API 의 결과로 RESULT_AUTH_READY_SUCCESS 를 반환하기 전에 resumeAuthentication API 가 호출될 경우, 콜백을 주지 않고 바로 이어서 진행된다.
화면전환등의 이유로 콜백을 받을 대상이 전환되었을 경우 Delegate 를 꼭 변경하여야 한다.

3.3 Deregistration Class

3.3.1 Class Overview

인증장치를 FIDO 서버에서 해지하기 위한 메시지 생성 및 분석하는 기능을 제공한다.

3.3.2 Summery

Public Constructors	
id	getInstance 인스턴스를 생성한다.

Public Methods	
void	setReqURL:(NSString *)fidoReqURL WithTimeout:(int)timeOut FIDO Server 요청 url, 응답대기시간(Default 30 초) 설정한다.
void	startDeregistration:(NSMutableDictionary *) contextTable; 인증장치 해지 요청 메시지를 생성한다.
void	startDeregistrationAll:(NSMutableDictionary *)contextTable; 전체 인증장치 해지 요청 메시지를 생성한다.
~~FidoResult~~	~~getTouchIDState~~ 삭제 ~~Local 인증을 수행하기 위한 단말기의 상태(사용가능 단말, 불가능 단말, 지문등록 여부)를 확인 할 수 있다.~~
~~BOOL~~	~~isAvailableFIDO~~ 삭제 ~~사용자의 단말기가 FIDO 를 사용 할 수 있는지 확인 한다.~~
NSString	getToken 서버로부터 전달 받은 Token 값을 받는다.

3.3.3 Public Constructor

+ (id) **getInstance**

인스턴스를 생성한다.

3.3.4 Public Methods

- (void) **setReqURL**:(NSString *)fidoReqURL **WithTimeout**:(int)timeOut;

FIDO Server 요청 url, 응답대기시간(Default 30 초) 설정하고 인스턴스를 생성한다.

Parameters

fidoReqURL 정책 요청 FIDO 서버 URL

timeOut FIDO 서버 접속 및 응답 Timeout

Remarks

startDeregistration 을 호출 하기 전에 실행한다. 네트워크 타임아웃을 세팅하지 않을 경우 30 초로 정해진다.

- (void) **startDeregistration**:(NSMutableDictionary *) contextTable

인증장치 해지 요청 메시지를 생성한다.

Parameters

contextTable FIDO 서버에서 사용자를 구분하기 위한 Dictionary (샘플코드 및 개발가이드의 4.3 참고)

Remarks

* 해지 요청 한 결과를 생성자에서 설정 한 delegate 를 통해 리턴 받는다.

* ContextKey List

키	나타내는 값
< 필수 >	
Key_Context_UserID	사용자 ID
< 옵션 >	
Key_Context_DeviceID	단말기 ID
Key_Context_DeviceModel	단말기 모델명
Key_Context_DeviceOS	단말기 OS

* Delegate 에 대한 유의 사항 (샘플코드 및 개발가이드의 4.6 참고)

- (void) **startDeregistrationAll**:(NSMutableDictionary *) contextTable

전체 인증장치 해지 요청 메시지를 생성한다.

Parameters

contextTable FIDO 서버에서 사용자를 구분하기 위한 Dictionary (사용 예제 참고)

Remarks

* 해지 요청 한 결과를 생성자에서 설정 한 delegate 를 통해 리턴 받는다.

* delegate 에 대한 유의 사항 – 위(startDeregistration)와 동일

- (NSString *) **getToken;**

FIDO 서버로부터 전달 받은 Token 을 획득한다.

Returns

Token

Remarks

인증장치 해지 메시지에 대한 결과가 성공일 경우에 값을 획득 할 수 있다.

3.4 MagicFidoUtil Class

3.4.1 Class Overview

FIDO 를 사용할 수 있는 기본 공통 함수들이 있으며, 인증장치(지문, 패턴, 패스코드)에 대해서 사용 가능 여부 및 인증장치(지문, 패턴, 패스코드)의 상태값 확인, 설정 된 값의 변경 또는 설정 된 값의 초기화 등에 관련된 기능을 제공한다.

3.4.2 Summery

Static Methods	
BOOL	isAvailableFIDO:(LOCAL_AUTH_TYPE)localType 인증장치 타입에 따른 사용자의 단말기가 인증장치를 사용할 수 있는지 확인한다.
NSInteger	getAvailableTypes 사용자의 단말기에 사용 가능한 인증장치를 확인 할 수 있다.
FidoResult	getLocalVerifyState:(LOCAL_AUTH_TYPE)localType FIDO 인증을 수행하기 위한 단말기의 상태(사용가능 단말, 불가능 단말, 지문등록 여부)를 확인 할 수 있다.
void	changeUserVerification:(LOCAL_AUTH_TYPE)localType completion:(void(^)(LOCAL_AUTH_TYPE requestType, FidoResult *result))completion 인증장치(패턴, 패스코드)의 설정값을 변경 요청한다. (Ex. 패턴 변경)
FidoResult	resetUserVerification:(LOCAL_AUTH_TYPE)localType 인증장치(패턴, 패스코드)의 설정값을 초기화한다.
void	~~setPatternStyle:(NSMutableDictionary *)styleDic~~ 삭제 (MagicFidoUtil 클래스의 setAuthenticatorOptions:(NSDictionary *)optionDic authenticatorType:(LOCAL_AUTH_TYPE)localType 를 이용) 패턴 인증장치의 스타일을 설정한다.
void	setAuthenticatorOptions:(NSDictionary *)optionDic authenticatorType:(LOCAL_AUTH_TYPE)localType 인증장치의 UI 나 제한사항을 설정한다.
BOOL	~~isAvailableFIDO~~ 삭제 (MagicFidoUtil 클래스의 isAvailableFIDO:(LOCAL_AUTH_TYPE)localType 를 이용) 사용자의 단말기가 FIDO 를 사용 할 수 있는지 확인 한다.
FidoResult	~~getTouchIDState~~ 삭제 (MagicFidoUtil 클래스의 getLocalVerifyState:(LOCAL_AUTH_TYPE)localType 를 이용) Local 인증을 수행하기 위한 단말기의 상태(사용가능 단말, 불가능 단말, 지문등록 여부)를 확인 할 수 있다.
void	setSSLEnable:(BOOL)enable SSL 검증 활성화 상태를 설정할 수 있다.
NSString	getVersion

	SDK 버전을 확인할 수 있다.
void	setAuthenticatorResetCallbackEnable:(BOOL)enable 인증 화면에서 재등록 버튼이 보이도록 설정할 수 있다.
void	setPasscodeMaxInputAutoNextStepEnable:(BOOL)enable 패스코드 인증장치 최대길이 입력 시 다음 단계로 진행하도록 설정할 수 있다.
void	setPasscodeUIType:(FIDO_UI_TYPE)type 패스코드 인증장치의 UI 를 변경할 수 있다.
void	setKeypadAccessibilityOnSpeakerEnable:(BOOL)enable 보안키패드의 VoiceOver 스피커 모드를 설정할 수 있다.
void	setPasscodeIgnoreList:(NSDictionary *)ignore 패스코드 인증장치의 값으로 허용하지 않을 값들을 설정할 수 있다.
void	setPasscodeRegularExpressionList:(NSDictionary *)expression 패스코드 인증장치값의 제약사항에 대한 정규표현식들을 설정할 수 있다.

3.4.3 Public Methods

+ (BOOL)**isAvailableFIDO:**(LOCAL_AUTH_TYPE)localType

사용자의 단말기에 사용 가능한 인증장치를 확인 할 수 있다.

Parameters

localType 확인하려는 인증장치 타입
LOCAL_FINGERPRINT_TYPE : 지문 타입
LOCAL_FACEPRINT_TYPE : 얼굴 타입
LOCAL_PATTERN_TYPE : 패턴 타입
LOCAL_PASSCODE_TYPE : 패스코드 타입

Return

가능하면 YES, 불가능하면 NO

Remarks

인증장치별 사용가능 여부를 확인 할 수 있다.
FaceID 의 경우 프로젝트 설정에 FaceID Privacy Description 을 입력하지 않았을 경우 NO 를 리턴한다.

+ (NSInteger)**getAvailableTypes**

사용자의 단말기에 사용 가능한 인증장치를 확인 할 수 있다.

Return

지문, 패턴, 패스코드 인증장치를 사용 가능한지에 대한 결과를 전달한다.

Remarks

* 결과값 확인 방법.

```
// 사용가능한 모든 인증장치 타입
NSInteger availables = [MagicFidoUtil getAvailableTypes];

if (availables & LOCAL_FINGERPRINT_TYPE) {
    NSLog(@"지문 인증장치가 사용 가능합니다.");
}
if (availables & LOCAL_FACEPRINT_TYPE) {
    NSLog(@"얼굴 인증장치가 사용 가능합니다.");
}
if (availables & LOCAL_PATTERN_TYPE) {
    NSLog(@"패턴 인증장치가 사용 가능합니다.");
}
if (availables & LOCAL_PASSCODE_TYPE) {
    NSLog(@"패스코드 인증장치가 사용 가능합니다.");
}
```

+ (FidoResult *)**getLocalVerifyState:**(LOCAL_AUTH_TYPE)localType

FIDO 인증을 수행하기 위한 단말기의 상태(사용가능 단말, 불가능 단말, 등록 여부)를 확인 할 수 있다.

Parameters

localType 확인하려는 인증장치 타입
LOCAL_BIOMETRY_TYPE : 생체 타입 (지문 혹은 얼굴)
LOCAL_PATTERN_TYPE : 패턴 타입
LOCAL_PASSCODE_TYPE : 패스코드 타입

Return

현재 상태를 나타내는 FidoResult

Remarks

결과값을 FidoResult 객체를 통해 확인할 수 있다.

상태	설명
FidoResult_Authenticator_State_Available	사용 가능
FidoResult_Authenticator_State_Not_Support	지원하지 않음
FidoResult_Authenticator_State_Not_Registed	지원하고 등록되지 않음
FidoResult_Authenticator_State_Locked	지원하고 잠겨있음
- 생체인증 전용	
FidoResult_Biometry_State_Not_Support	iOS 버전이 9 미만인 경우 생체인증 API 이용 불가
FidoResult_Biometry_State_Empty	Fido 등록이 안되어 있는 경우
FidoResult_Biometry_State_Not_Changed	생체인증 상태가 변하지 않은 상태
FidoResult_Biometry_State_Change	생체인증 상태가 변한 상태
FidoResult_Biometry_State_Not_Reg_Passcode	생체인증 패스코드가 등록이 안된 상태
FidoResult_Biometry_State_Not_Reg_Finger	생체인증이 등록되어 있지 않은 경우
FidoResult_Biometry_State_Lock	생체인증 시도 횟수 초과 등으로 잠긴 상태
FidoResult_Biometry_Not_Support	생체인증을 지원하지 않는 단말

+ (void)**changeUserVerification:**(LOCAL_AUTH_TYPE)localType **completion**:(void (^)(LOCAL_AUTH_TYPE requestType, FidoResult *result))completion

인증장치(패턴, 패스코드)의 설정값을 변경한다. (Ex. 패턴 변경)

Parameters

localType 변경하려는 인증장치 타입
LOCAL_PATTERN_TYPE : 패턴 타입
LOCAL_PASSCODE_TYPE : 패스코드 타입

completion 변경 결과 콜백 블럭

Remarks

인증장치의 값을 변경할 수 있으며, 단말기에서 관리 되는 인증장치(Ex. 지문)는 해당하지 않는다.

상태	설명
FidoResult_Authenticator_Change_Success	변경 성공
FidoResult_Authenticator_Change_Fail	변경 실패
FidoResult_Authenticator_Change_User_Cancel	변경 사용자 취소
FidoResult_Authenticator_Change_Invalid	유효하지 않은 인증장치

+ (FidoResult *)**resetUserVerification**:(LOCAL_AUTH_TYPE)localType

인증장치(지문 또는 패턴)의 설정값을 초기화한다.

Parameters

localType 초기화하려는 인증장치 타입
LOCAL_PATTERN_TYPE : 패턴 타입
LOCAL_PASSCODE_TYPE : 패스코드 타입

Return

초기화 상태를 나타내는 FidoResult

Remarks

인증장치 설정만을 초기화 하는 기능으로 인증장치에 등록된 사용자를 삭제 하지는 않는다. 해당 기능을 사용할 때에는 startDeregistration() API 함께 사용하는 것을 권고한다.

결과값을 FidoResult 객체를 통해 확인할 수 있다. 단말기에서 관리 되는 인증장치(Ex. 지문)는 초기화 되지 않는다.

상태	설명
FidoResult_Authenticator_Reset_Success	삭제 성공
FidoResult_Authenticator_Reset_Fail	삭제 실패
FidoResult_Authenticator_Reset_None	삭제 할 인증장치의 값이 없음
FidoResult_Authenticator_Invalid_Type	잘못 된 인증장치 타입

+ (void)**setAuthenticatorOptions:**(NSMutableDictionary *)optionDic
authenticatorType:(LOCAL_AUTH_TYPE)localType

인증장치의 UI 나 제한사항을 설정한다.

Parameters

optionDic 인증장치에 설정 할 사항들의 맵

localType 설정 하려는 인증장치 타입
LOCAL_PATTERN_TYPE : 패턴 타입
LOCAL_PASSCODE_TYPE : 패스코드 타입

Remarks

인증장치의 제한사항을 설정할 수 있다.

키	설명	값 타입
- 인증장치 공통		
KEY_RETRY_COUNT_TO_LOCK	인증장치 잠김까지 시도 가능 횟수	NSNumber(int)
KEY_LOCK_TIME	인증장치가 잠겨있는 시간	NSNumber(double)
KEY_MAX_LOCK_COUNT	인증장치 잠김 허용 횟수	NSNumber(int)
- 패턴 인증장치		
KEY_MINIMUM_NODE_COUNT	패턴을 구성 할 최소 노드 갯수	NSNumber(int)
KEY_NODE_SIZE_WIDTH	노드의 크기(정사각형의 사이즈이므로 가로값만 받음)	NSNumber(float)
KEY_NODE_IMAGE_DEFAULT	기본 노드 이미지 이름	NSString
KEY_NODE_IMAGE_SELECTED	선택 된 노드 이미지 이름	NSString
KEY_NODE_IMAGE_MISMATCH	패턴 불일치 노드 이미지 이름	NSString
KEY_LINE_WIDTH	패턴 라인 굵기	NSNumber(float)
KEY_LINE_COLOR	패턴 라인 색상	UIColor
KEY_MISMATCH_LINE_COLOR	패턴 불일치 라인 색상	UIColor
- 패스코드 인증장치		
KEY_MIN_LENGTH	패스코드로 등록할 수 있는 최소 길이	NSNumber(int)
KEY_MAX_LENGTH	패스코드로 등록할 수 있는 최대 길이	NSNumber(int)
KEY_USE_NUMBER_KEYPAD	숫자키패드 사용 여부	NSNumber(BOOL)
KEY_REGULAR_EXPRESSION	패스코드 정규 표현식	NSString
KEY_SHUFFLE_DISABLE	패스코드 숫자키패드 셔플 비활성화 여부	NSNumber(BOOL)

<정규 표현식 예>

영문,숫자,특수문자 각 1 자 이상, 전체 8 자 이상	^(?=.*[A-Za-z])(?=.*₩d)(?=.*[$@$!%*#?&])[A-Za-z₩d$@$!%*#?&]{8,}$
대문자,소문자,숫자 각 1 자	^(?=.*[a-z])(?=.*[A-Z])(?=.*₩d)[a-zA-Z₩d]{8,}$

이상, 전체 8 자 이상	
대문자,소문자,숫자,특수문자 각 1 자 이상, 전체 8 자 이상	^(?=.*[a-z])(?=.*[A-Z])(?=.*₩d)(?=.*[$@$!%*?&])[A-Za- z₩d$@$!%*?&]{8,}
숫자만 입력, 같은 숫자 세자리 연속 방지	^(?!.*(₩d)₩1₩1)[₩d]{0,}$
숫자만 입력, 연속하는 오름차순 숫자 3 자리 방지	^(?!.*(012))(?!.*(123))(?!.*(234))(?!.*(345))(?!.*(456))(?!.*(567))(?!.*(678))(?!.*(789)) [₩d]{0,}$

입력받은 정규표현식을 패스코드 설정값의 제한사항으로 적용하는 기능을 지원하며, 정규표현식의 생성과 유효성 검증은 SDK 를 적용하는 앱개발사의 역영이다. 위의 표는 정규표현식의 예시를 제공한다.
패스코드 인증장치의 설정값(MIN_LENGTH, MAX_LENGTH, USE_NUMBER_KEYPAD, REGULAR_EXPRESSION)은 패스코드의 UI Type 이 FIDO_PASSCODE_NORMAL 인 경우만 지원한다.

+ (void) **setSSLEnable**:(BOOL)enable**;**

SSL 인증서를 검증 여부를 세팅한다. (Default : true)

Parameters

enable 검증 여부
검증 YES / 비검증 NO

Remarks

운영시에는 SSL 인증서를 검증 해야만 한다.

+ (NSString *) **getVersion;**

SDK 버전을 확인할 수 있다.

Return

SDK 버전 스트링

+ (void) **setAuthenticatorResetCallbackEnable**:(BOOL)enable**;**

인증 화면에서 재등록 버튼이 보이도록 설정할 수 있다.

Parameters

enable 설정 여부
설정 YES / 비설정 NO

Remarks

설정이 되어 있을 경우 생체 인증장치를 제외한 인증 화면에서 재등록 버튼이 활성화 되어 버튼 선택 시 결과코드 980 을 리턴한다.

+ (void) **setPasscodeMaxInputAutoNextStepEnable**:(BOOL)enable**;**

패스코드 인증장치 최대길이 입력 시 다음 단계로 진행하도록 설정할 수 있다.

Parameters

enable 설정 여부
설정 YES / 비설정 NO

Remarks

설정이 되어 있을 경우 패스코드 인증장치에서 입력 최대값을 입력 시 자동으로 다음단계를 진행한다.

+ (void)**setPasscodeUIType:**(FIDO_UI_TYPE)type;

패스코드 인증장치의 UI 를 변경할 수 있다.

Parameters

type 설정하려는 패스코드 UI 타입
FIDO_PASSCODE_NORMAL : 기본 타입
FIDO_PASSCODE_PIN4 : 4 개의 핀 타입
FIDO_PASSCODE_PIN6 : 6 개의 핀 타입

Remarks

호출하지 않을 경우 기본값은 FIDO_PASSCODE_NORMAL.
UI 타입과는 상관없이 패스코드 인증장치는 하나의 값만을 가진다.
(PIN4 에서 1111 로 등록한 경우, NORMAL 에서도 1111 이며, PIN6 에서는 사용 불가)

+ (void)**setKeypadAccessibilityOnSpeakerEnable:**(BOOL)enable;

보안키패드의 VoiceOver 스피커 모드를 설정할 수 있다. (default : false)

Parameters

enable 설정 여부
설정 YES / 비설정 NO

Remarks

보안키패드의 키 영역은 기본적으로 스피커의 경우 VoiceOver 로 읽어주지 않도록 설정되어 있다.
필요한 경우 YES 로 설정하여 스피커를 통해서도 VoiceOver 로 읽히도록 설정할 수 있다.

+ (void)**setPasscodeIgnoreList:**(NSDictionary *)ignore;

패스코드 인증장치의 값으로 허용하지 않을 값들을 설정할 수 있다.

Parameters

ignore 패스코드 인증장치의 값으로 허용하지 않을 값의 해시맵
- Key : 허용하지 않는 스트링
- Value : 허용하지 않는 스트링을 포함한 경우의 에러 메세지 스트링

Remarks

사용자가 패스코드값 설정 시 ignore 의 key 값으로 전달받은 값을 포함하여 설정할 경우 ingore 의 value 값으로 메세지를 보여주며, 패스코드값 등록을 허용하지 않는다.

기본적으로 지원하는 제한사항도 있으며, 해당 키값과 제한사항은 아래와 같다.

키	내용
KEY_EQUAL_CHARACTER	동일한 입력값 (ex: 111111, aaaaaa)
KEY_ASCENDING_DESCENDING	오름차순, 내림차순 (ex: abcdef, 654321)
KEY_2CHARACTER_REPEAT	두 문자 반복 사용 (ex: 373737)
KEY_2EQUAL_CHARACTER_REPEAT	두 동일문자 연속 조합 (ex: 335577)
KEY_NOT_NUMBER_ENGLISH	숫자와 영어를 제외한 다른 문자 사용 (ex: 비밀번호)

+ (void)**setPasscodeRegularExpressionList:**(NSDictionary *)expression;

패스코드 인증장치값의 제약사항에 대한 정규표현식들을 설정할 수 있다.

Parameters

expression 패스코드 인증장치의 입력값을 체크할 정규표현식의 해시맵
- Key : 정규표현식
- Value : 정규표현식에 허용되지 않을 경우 에러 메세지 스트링

Remarks

사용자가 설정한 패스코드값이 expression 의 key 값으로 전달받은 정규표현식에 허용되지 않을 경우 expression 의 value 값으로 메세지를 보여주며, 패스코드값 등록을 허용하지 않는다.

정규표현식의 생성과 유효성 검증은 SDK 를 적용하는 앱개발사의 역영이다.

3.5 FIDOdelegate

3.5.1 delegate Overview

FIDO 의 등록, 인증, 해지에 대한 수행결과 delegate.

3.5.2 Summery

Public Interface	
void	**onFIDOResultWithResult**:(BOOL)result **WithFidoResult**:(FidoResult *) fidoResult; FIDO 수행 결과에 대한 결과

3.6 FIDOResult Class

3.6.1 Class Overview

실행 결과에 대한 클래스

3.6.2 Summary

Values		
RESULT_SUCCESS	0	성공
RESULT_AUTH_READY_SUCCESS	10	사용자 인증 준비 완료
RESULT_AUTHENTICATOR_RESET	980	인증장치 값 재등록 요청 <사용자 액션>
RESULT_USER_CANCEL	999	사용자 취소
ERROR_INVALID_PARAMETER	1001	전달 인자값이 올바르지 않습니다.
ERROR_SET_CHANNEL_BINDING	1002	TLS 인증서가 올바르지 않습니다.
ERROR_JSON_PARSER	1005	JSON 메세지 분석 중 오류가 발생하였습니다.
ERROR_NOT_MATCH_FIDO_CLIENT	1006	FIDO Client 가 설치되어 있지 않습니다.
ERROR_CANNOT_USE_FIDO	1007	FIDO 를 사용 할 수 없는 단말기 입니다.
ERROR_ALREADY_REG_USER	1008	이미 등록된 사용자입니다. <등록 진행을 위해 해지(Deregistration 의 startDeregistration) 후, 재등록 로직 수행.>
ERROR_NOT_REG_USER	1009	인증하려는 아이디의 데이터가 존재하지 않습니다. <인증을 하려면 해지(Deregistration 의 startDeregistration) 후, 재등록 로직 수행>
ERROR_INVALID_USERNAME	1011	username 에는 ' '(공백) 문자 또는 한글을 지원하지 않음
ERROR_INVALID_AUTHENTICATOR_TYPE	1012	유요하지 않은 인증장치 타입입니다.

ERROR_AUTH_REQUEST_NOT_EXIST	1100	인증 요청이 없습니다.
ERROR_UNTRUSTED_FACET_ID	2001	인증되지 않은 RP Client 입니다.
ERROR_NO_SUITABLE_AUTHENTICATOR	2002	적합한 인증장치가 설치되어 있지 않습니다. <등록시 발생 : 이미 등록된 아이디일 경우> <인증시 발생 : 등록안된 아이디일 경우>
ERROR_UNSUPPORTED_VERSION	2003	지원하지 않는 FIDO 프로토콜 버전입니다.
Authenticator_State_Not_Support	3005	인증장치를 지원하지 않음
Authenticator_State_Available	3006	인증장치 사용 가능
Authenticator_State_Not_Registed	3007	인증장치의 값이 등록되어 있지 않음
Authenticator_State_Locked	3008	인증장치가 잠겨 있음
LOCAL_FAIL_CHANGE	3010	인증장치 값 변경 실패
LOCAL_FAIL_RESET	3011	인증장치 값 리셋 실패
LOCAL_FAIL_VERIFY	3012	인증장치 인증 실패
LOCAL_FAIL_VERIFY_LOCKED	3013	인증장치 잠김
ERROR_MAX_LOCK_COUNT_OVER	3014	잠김 허용 횟수 초과
IGNORE_LOCK_OPTION_SET	3015	인증장치 잠금 관련 설정 거부 (한번이라도 다른값을 입력하여 틀린 후 설정값을 변경하려고 할 경우 차단, 바른값을 입력 한 후에는 변경 가능)
ERROR_NETWORK_STATE	4001	네트워크 오류 발생 Excpetion 내용
Biometry_State_Not_Changed	7000	생체인증 상태가 변하지 않았습니다.
Biometry_State_Changed	7001	생체인증 상태가 변했습니다.
Biometry_State_Not_Support	7002	생체인증 상태 확인은 iOS9 이상부터 가능합니다.
Biometry_State_Error	7003	FIDO 등록 후, 생체인증 상태 확인이 가능합니다.
Biometry_State_Auth_Fail	7004	생체인증에 실패했습니다.
Biometry_State_Lock	7005	생체인증 입력 횟수 초과로 인증에 실패했습니다.
Biometry_State_Not_Registed	7006	생체인증 값이 등록되어 있지 않습니다.
Biometry_State_Not_Reg_Passcode	7007	생체인증 암호가 등록되어 있지 않습니다.
Biometry_Not_Support	7008	생체인증을 지원하지 않는 단말입니다.

Biometry_User_Fallback	7009	사용자가 Fallback 선택
Biometry_App_Cancel	7010	어플리케이션이 취소시킴
Biometry_System_Cancel	7011	시스템이 취소시킴
Biometry_Invalid_Context	7012	생체인증이 무효화 됨
Biometry_Not_Available	7013	생체인증을 사용할 수 없는 단말기
Biometry_Not_Interactive	7014	사용자 인터페이스를 노출하는 것이 금지 됨
Biometry_Unknown	7015	알 수 없는 에러(생체인증)
Authenticator_Reset_Success	7100	삭제 성공
Authenticator_Reset_Fail	7101	삭제 실패
Authenticator_Reset_None	7102	삭제 할 인증장치의 값이 없음
Authenticator_Invalid_Type	7103	잘못 된 인증장치 타입
Authenticator_Change_Success	7110	변경 성공
Authenticator_Change_Fail	7111	변경 실패
Authenticator_Change_User_Cancel	7112	변경 사용자 취소
Authenticator_Change_Invalid	7113	유효하지 않은 인증장치
ERROR_FORBIDDEN_FIDO	9000	일시적인 오류로 FIDO 를 사용 할 수 없습니다. 확인 후 다시 이용해 주세요.
ERROR_UNKNOWN	9999	알수 없는 오류 입니다.
ERROR_EMPTY_USERNAME	19019	Username 은 필수 입력 해야 함

Public Methods	
NSString	getDescription FIDO 실행결과에 대한 설명을 획득한다.
int	getErrorCode FIDO 실행결과에 대한 결과코드를 획득한다.

3.6.3 Public Methods

- (NSString *) **getDescription**;

FIDO 실행결과에 대한 설명을 획득한다.

Returns

FIDO 결과에 대한 설명

- (int) **getResultCode**;

FIDO 실행결과에 대한 결과코드를 획득한다.

Returns

FIDO 결과에 대한 코드

4 사용 예제

4.1 FIDO 등록 (Registration) 사용 예제

```
@interface FIDORegistration () {
    Registration *_mRegist;
}

@end

@implementation FIDORegistration

- (instancetype)init {
    self = [super init];
    if (self) {
        //Registration init
        _mRegist = [Registration getInstance];

        NSString *serverURL = [ConfigurationViewController serverUrl];
        NSString *requestURL = [ConfigurationViewController reqUrl];
        NSString *reqFullURL = [NSString stringWithFormat:@"%@%@", serverURL,
requestURL];
        NSString *responseURL = [ConfigurationViewController resUrl];
        NSString *resFullURL = [NSString stringWithFormat:@"%@%@", serverURL,
responseURL];

        [_mRegist setReqURL:reqFullURL WithResURL:resFullURL WithTimeOut:15];

    }
    return self;
}

- (void)registFIDOWithUserID:(NSString *)userID {
    [self registFIDO:userID];
}

- (void)registFIDO:(NSString *)userID {
    //RPContext를 위한 세팅.
    NSMutableDictionary *settingValue = [NSMutableDictionary new];

    [settingValue setObject:userID forKey:@"userName"];

    [_mRegist startRegistration:settingValue];
}

@end
```

4.2 FIDO 인증 (Authentication) 사용 예제

```
@interface FIDOAuthentication () {
    Authentication *_mAuth;
}

@end
```

```
@implementation FIDOAuthentication

- (instancetype)init {
    self = [super init];
    if (self) {
        //Registration init
        _mAuth = [Authentication getInstance];

        NSString *serverURL = [ConfigurationViewController serverUrl];
        NSString *requestURL = [ConfigurationViewController reqUrl];
        NSString *reqFullURL = [NSString stringWithFormat:@"%@%@", serverURL,
requestURL];
        NSString *responseURL = [ConfigurationViewController resUrl];
        NSString *resFullURL = [NSString stringWithFormat:@"%@%@", serverURL,
responseURL];

        [_mAuth setReqURL:reqFullURL WithResURL:resFullURL WithTimeOut:15];
    }
    return self;
}

- (void)loginAuthFIDO:(NSString *)userID {
    [self startAuthentication:userID message:nil];
}

- (void)transferAuthFIDO:(NSString *)userID message:(NSString *)transferMessage {
    [self startAuthentication:userID message:transferMessage];
}

- (void)startAuthentication:(NSString *)userID message:(NSString *)transferMessage
{
    //RPContext를 위한 세팅.
    NSMutableDictionary *settingValue = [NSMutableDictionary new];

    [settingValue setObject:userID forKey:@"userName"];

    if(transferMessage != nil && ![transferMessage isEqualToString:@""]){
        [settingValue setObject:transferMessage forKey:@"transactionText"];
    }

    [_mAuth startAuthentication:settingValue];
}

@end
```

4.3 FIDO 해지 (DeRegistration) 사용 예제

```
@interface FIDODeRegistration () {
    Deregistration *_mDeregist;
}

@end
```

```
@implementation FIDODeRegistration

- (instancetype)init {
    self = [super init];
    if (self) {
        //Deregistration init
        _mDeregist = [Deregistration getInstance];

        NSString *serverURL = [ConfigurationViewController serverUrl];
        NSString *requestURL = [ConfigurationViewController reqUrl];
        NSString *reqFullURL = [NSString
stringWithFormat:@"%@%@",serverURL,requestURL];

        //Server Connect Setting.
        [_mDeregist setReqURL:reqFullURL WithTimeout:15];
    }
    return self;
}

- (void)deregistWithUserID:(NSString *)userID{
    //RPContext를 위한 세팅.
    NSMutableDictionary *settingValue = [NSMutableDictionary new];
    [settingValue setObject:userID forKey:@"userName"];

    /*******************************************************
     * 인증장치를 지정하여 해지하고 싶은 경우 설정
     * LOCAL_AUTH_TYPE 은 MagicFidoUtil.h 에 정의되어 있다.

     [settingValue setObject:[NSNumber numberWithInteger:LOCAL_PASSCODE_TYPE]
forKey:@"localType"];

     *******************************************************/

    [_mDeregist startDeregistration:settingValue];
}

- (void)deregistAllWithUserID:(NSString *)userID {
    //RPContext를 위한 세팅.
    NSMutableDictionary *settingValue = [NSMutableDictionary new];
    [settingValue setObject:userID forKey:@"userName"];

    [_mDeregist startDeregistrationAll:settingValue];
}

@end
```

4.4 FIDO 등록 (Registration) delegate 예외 처리 예제

```
- (void)onFIDOResultWithFidoResult:(FidoResult *)fidoResult {
    if ([fidoResult getErrorCode] == FidoResult_RESULT_SUCCESS) {
        // 에러코드 0 : 성공 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_RESULT_USER_CANCEL) {
        // 에러코드 999 : 사용자 취소시, 로직 수행
```

```
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_JSON_PARSER) {
        // 에러코드 1005 : 서버로부터 받은 메세지 파싱하다 에러 발생시, 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_NOT_SUPPORT_DEVICE) {
        // 에러코드 1007 : FIDO를 이용할 수 없는 단말일 시, 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_ALREADY_REG_USER) {
        // 에러코드 1008 : FIDO 등록진행시 서버에는 등록되어 있고, 로컬에는 등록된 사용자
정보가 없을시, 로직 수행
        // 이미 등록된 사용자이므로, 해지(Deregistration의 startDeregistration) 후 재등록
로직 수행.
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_UNTRUSTED_FACET_ID) {
        // 에러코드 2001 : 등록되지 않은 3rd App으로 진행시, 로직 수행(서버에 FacetID등록)
    }
    else if ([fidoResult getErrorCode] ==
FidoResult_ERROR_NO_SUITABLE_AUTHENTICATOR) {
        // 에러코드 2002 : 적합한 인증장치가 없음. 등록 가능한 모든 인증장치로 이미 등록되어
있는 경우.
        // 등록 된 사용자이기 때문에, 다시 등록할 수 없으며, 인증(Authentication의
startAuthentication)을 이용.
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_NETWORK_STATE) {
        // 에러코드 4001 : FIDO 서버와의 통신이 원활하지 않을 시, 로직 수행
        // [fidoResult getDescription] 을 통해 에러메세지 확인.
    }
    else if ([fidoResult getErrorCode] == FidoResult_TouchID_State_Auht_Fail) {
        // 에러코드 7004 : 터치아이디 인증에 실패했을 시, 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_TouchID_State_Lock) {
        // 에러코드 7005 : 터치아이디 상태가 락 상태일시, 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_UNKNOWN) {
        // 에러코드 9999 : FIDO 로직 수행 중 ,알 수 없는 에러 발생시, 로직 수행
    }
}
```

4.5 FIDO 인증 (Authentication) delegate 예외 처리 예제

```
- (void)onFIDOResultWithFidoResult:(FidoResult *)fidoResult {
    if ([fidoResult getErrorCode] == FidoResult_RESULT_SUCCESS) {
        //에러코드 0 : 성공 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_RESULT_USER_CANCEL) {
        //에러코드 999 : 사용자 취소시, 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_JSON_PARSER) {
        //에러코드 1005 : 서버로부터 받은 메세지 파싱하다 에러 발생시, 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_NOT_SUPPORT_DEVICE) {
        //에러코드 1007 : FIDO를 이용할 수 없는 단말일 시, 로직 수행
```

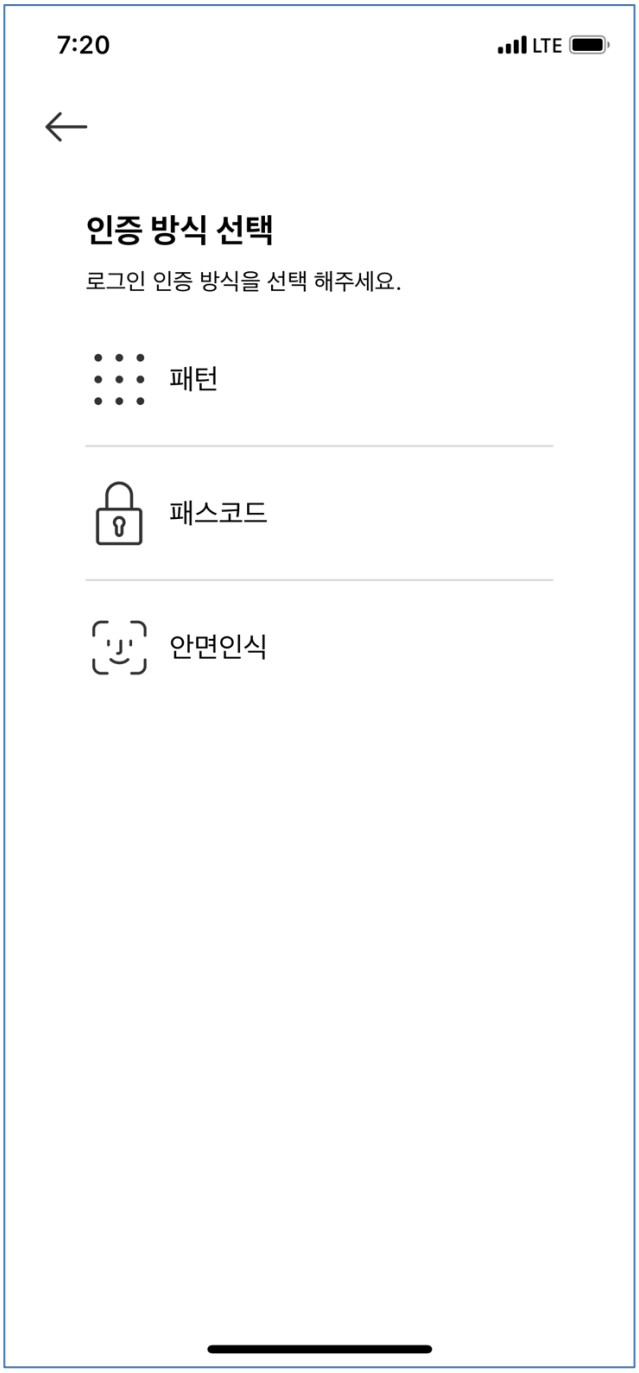
```
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_NOT_REG_USER) {
        //에러코드 1009 : FIDO 인증진행시 서버에는 등록되어 있고, 로컬에는 등록된 사용자
정보가 없을시, 로직 수행
        //해지(Deregistration의 startDeregistration) 후, 재등록 로직 수행.
    }
    else if ([fidoResult getErrorCode] == 13006) {
        //에러코드 13006 : 인증시 등록된 아이디가 없는 경우에 전달되는 서버 에러코드
        //등록(Registration의 startRegistration) 로직 수행.
    }
    else if ([fidoResult getErrorCode] == 19019) {
        //에러코드 19019 : 인증요청시 UserName을 입력하는 않은 경우에 전달되는 서버 에러코드
        //인증 요청시, UserName을 입력했는지 확인.
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_UNTRUSTED_FACET_ID) {
        //에러코드 2001 : 등록되지 않은 3rd App으로 진행시, 로직 수행
    }
    else if ([fidoResult getErrorCode] ==
FidoResult_ERROR_NO_SUITABLE_AUTHENTICATOR) {
        //에러코드 2002 : 적합한 인증장치가 없음. 서버에는 등록 되어 있지만, 단말기에 해당하는
데이터가 없음.
        //해지(Deregistration 의 startDeregistration) 후 재등록(Registration의
startRegistration)을 하여야 함.
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_NETWORK_STATE) {
        //에러코드 4001 : FIDO 서버와의 통신이 원활하지 않을 시, 로직 수행
        //[fidoResult getDescription] 을 통해 에러메세지 확인.
    }
    else if ([fidoResult getErrorCode] == FidoResult_TouchID_State_Auht_Fail) {
        //에러코드 7004 : 터치아이디 인증에 실패했을 시, 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_TouchID_State_Lock) {
        //에러코드 7005 : 터치아이디 상태가 락 일시, 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_UNKNOWN) {
        //에러코드 9999 : FIDO 로직 수행 중 ,알 수 없는 에러 발생시, 로직 수행
    }
}
```

4.6 FIDO 해지 (DerRegistration) delegate 예외 처리 예제

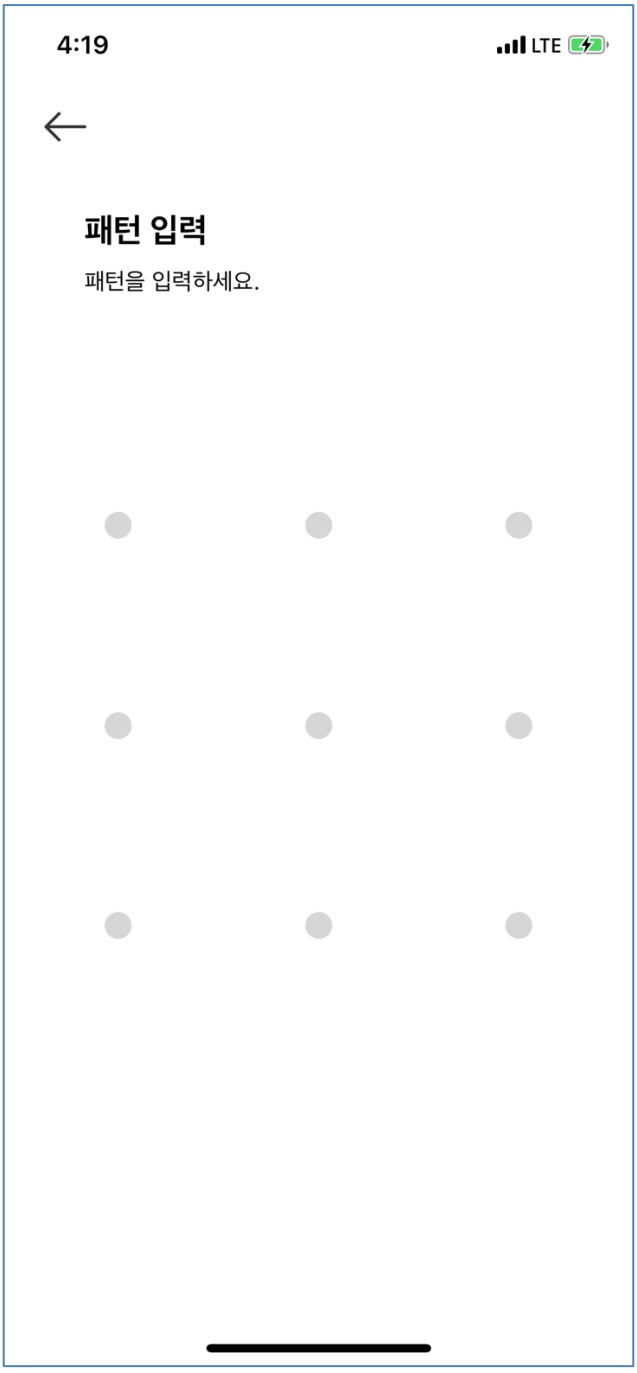
```
- (void)onFIDOResultWithFidoResult:(FidoResult *)fidoResult {
    if ([fidoResult getErrorCode] == FidoResult_RESULT_SUCCESS) {
        //에러코드 0 : 성공 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_JSON_PARSER) {
        //에러코드 1005 : 서버로부터 받은 메세지 파싱하다 에러 발생시, 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_NOT_SUPPORT_DEVICE) {
        //에러코드 1007 : FIDO를 이용할 수 없는 단말일 시, 로직 수행
    }
    else if ([fidoResult getErrorCode] ==
```

```
FidoResult_ERROR_INVALID_AUTHENTICATOR_TYPE) {
        //에러코드 1012 : 유효하지 않은 인증장치로 해지를 요청했을 시, 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_UNTRUSTED_FACET_ID) {
        //에러코드 2001 : 등록되지 않은 3rd App으로 진행시, 로직 수행
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_NETWORK_STATE) {
        //에러코드 4001 : FIDO 서버와의 통신이 원활하지 않을 시, 로직 수행
        //[fidoResult getDescription] 을 통해 에러메세지 확인.
    }
    else if ([fidoResult getErrorCode] == 15007) {
        //에러코드 15007 : FIDO 해지진행시 등록되지 않은 아이디로 요청시, 로직 수행
        // 등록된 아이디가 없기 때문에 등록(Registration의 startRegistration) 로직 수행.
    }
    else if ([fidoResult getErrorCode] == FidoResult_ERROR_UNKNOWN) {
        //에러코드 9999 : FIDO 로직 수행 중 ,알 수 없는 에러 발생시, 로직 수행
    }
}
```

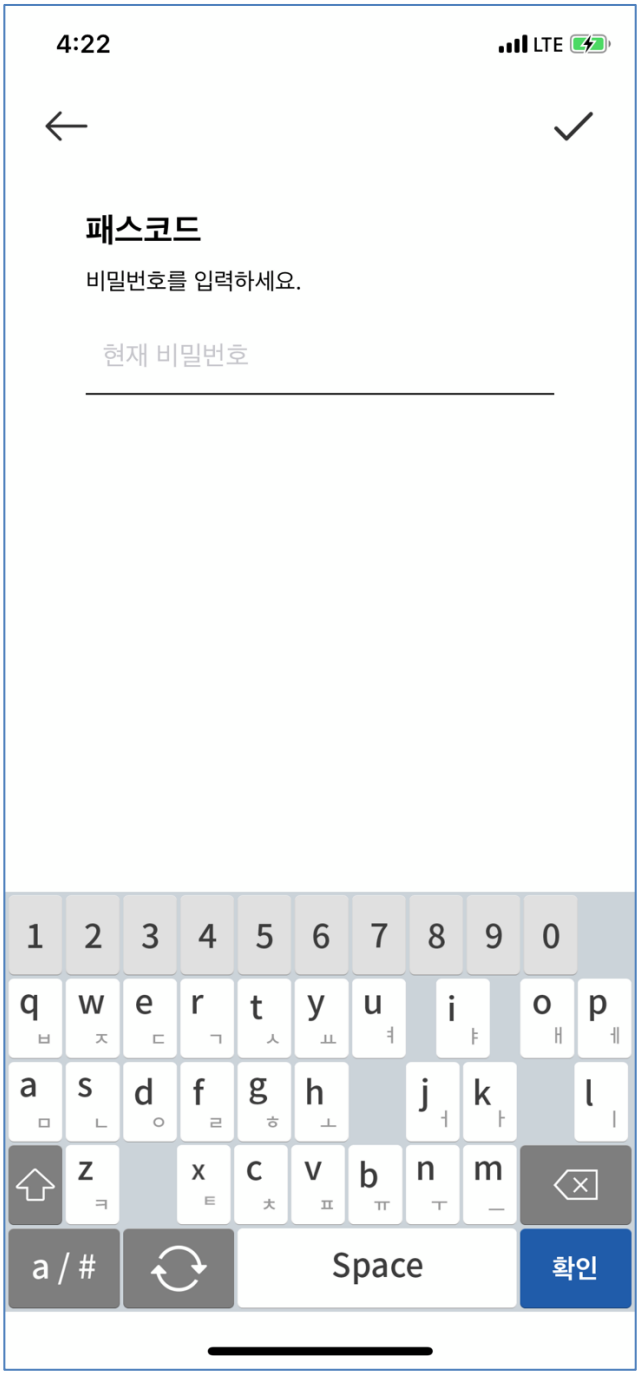
5 인증장치 화면 UI



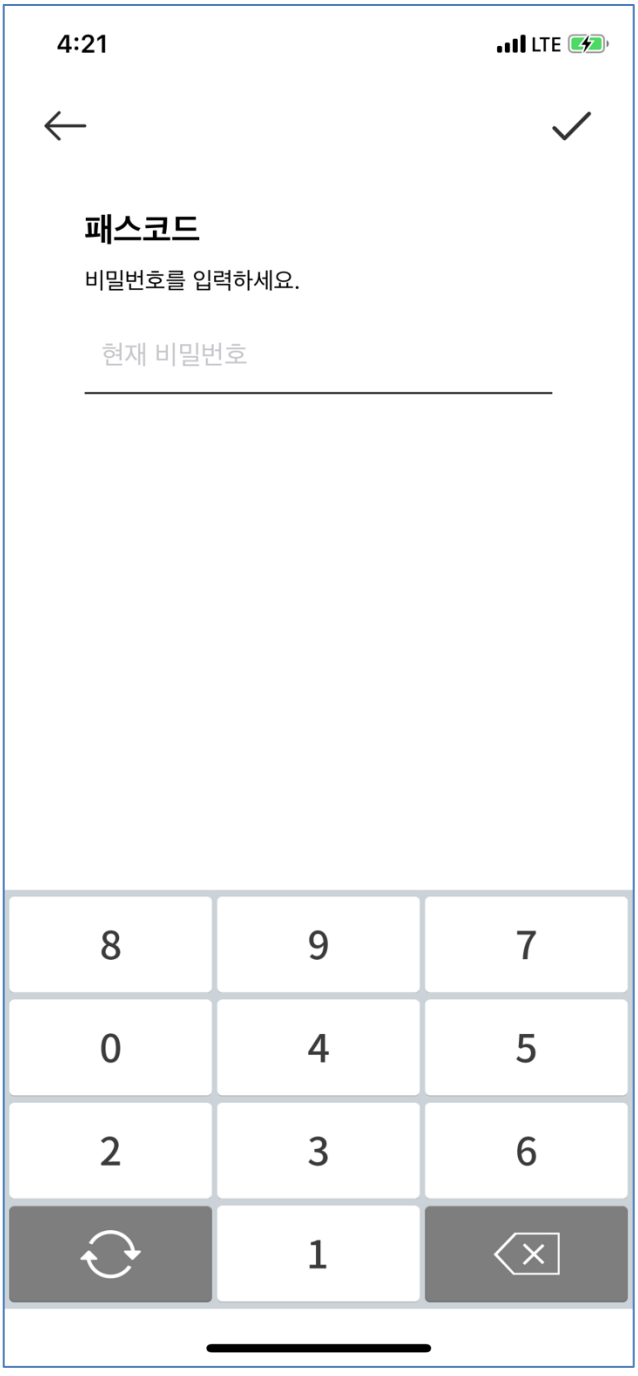
< 인증장치 선택 >



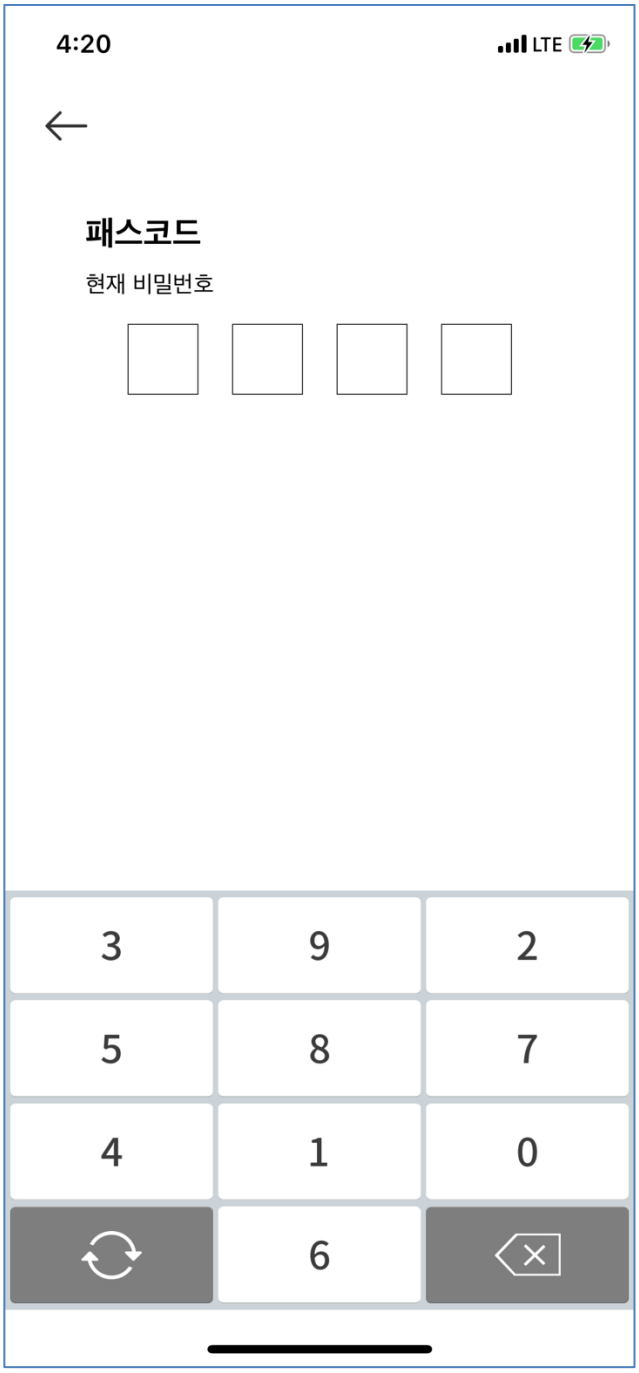
< 패턴 인증장치 >



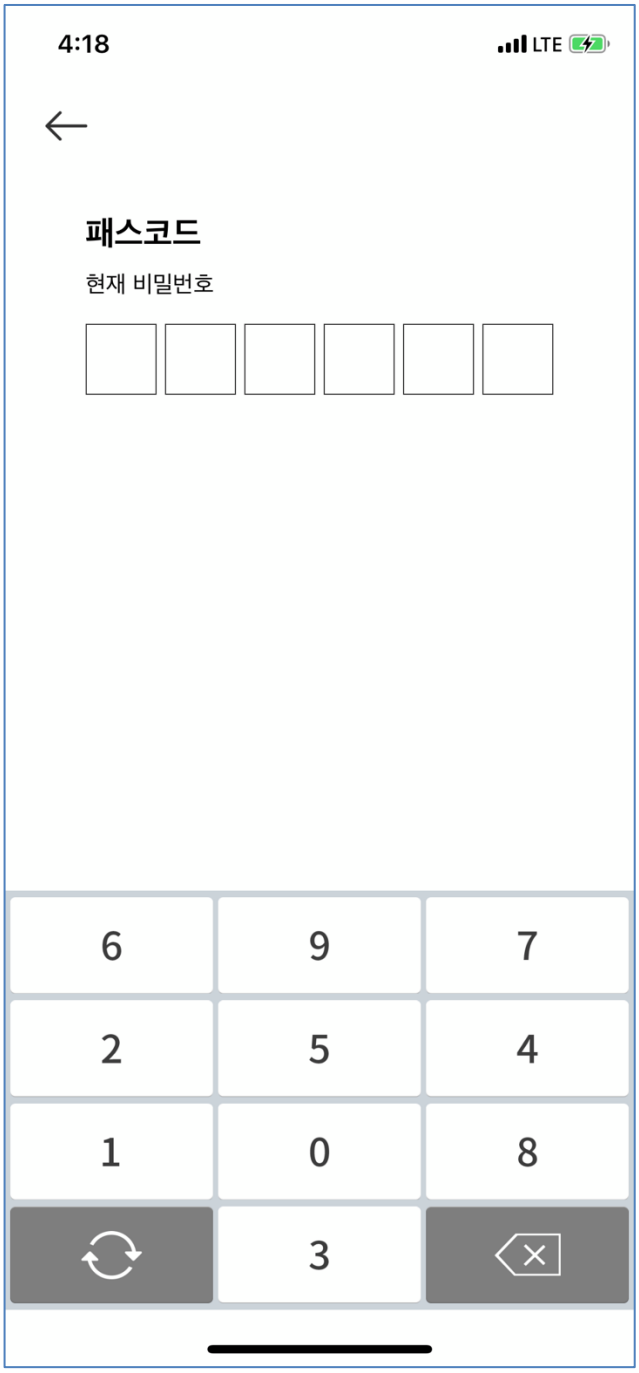
< 패스코드 인증장치(문자) >



< 패스코드 인증장치(숫자) >

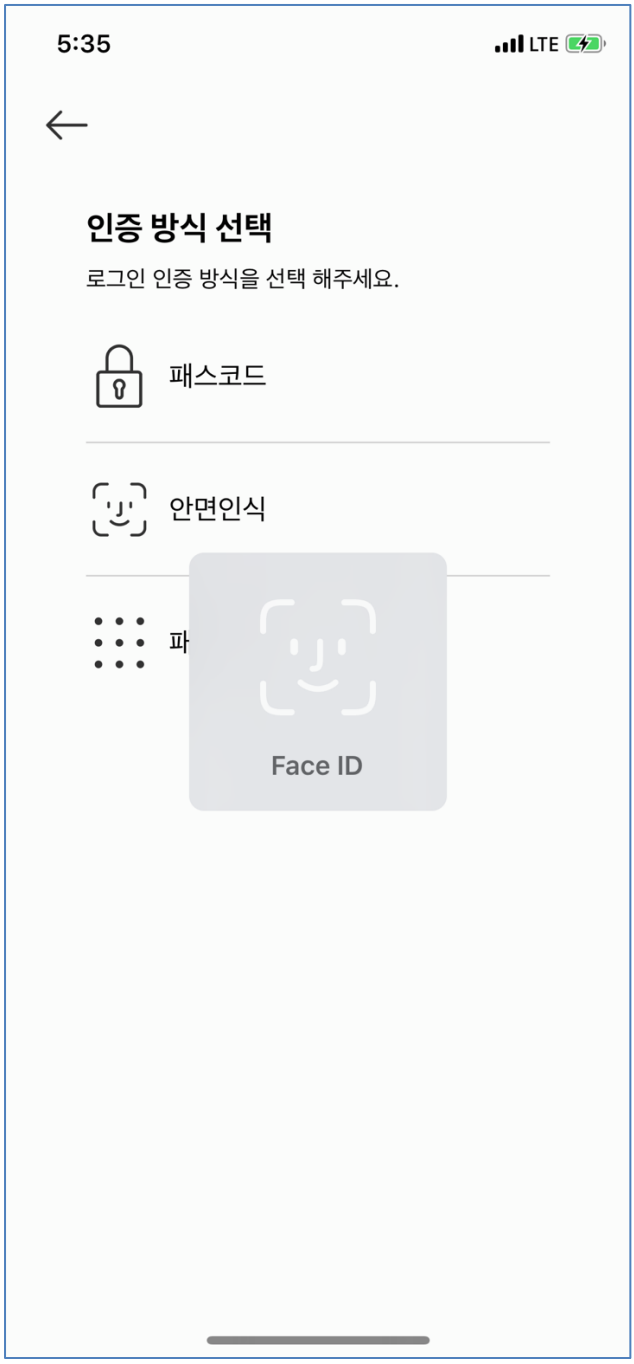


< 패스코드 인증장치(핀 4) >



< 패스코드 인증장치(핀 6) >

< 지문 인증장치 >



< 안면 인증장치 >