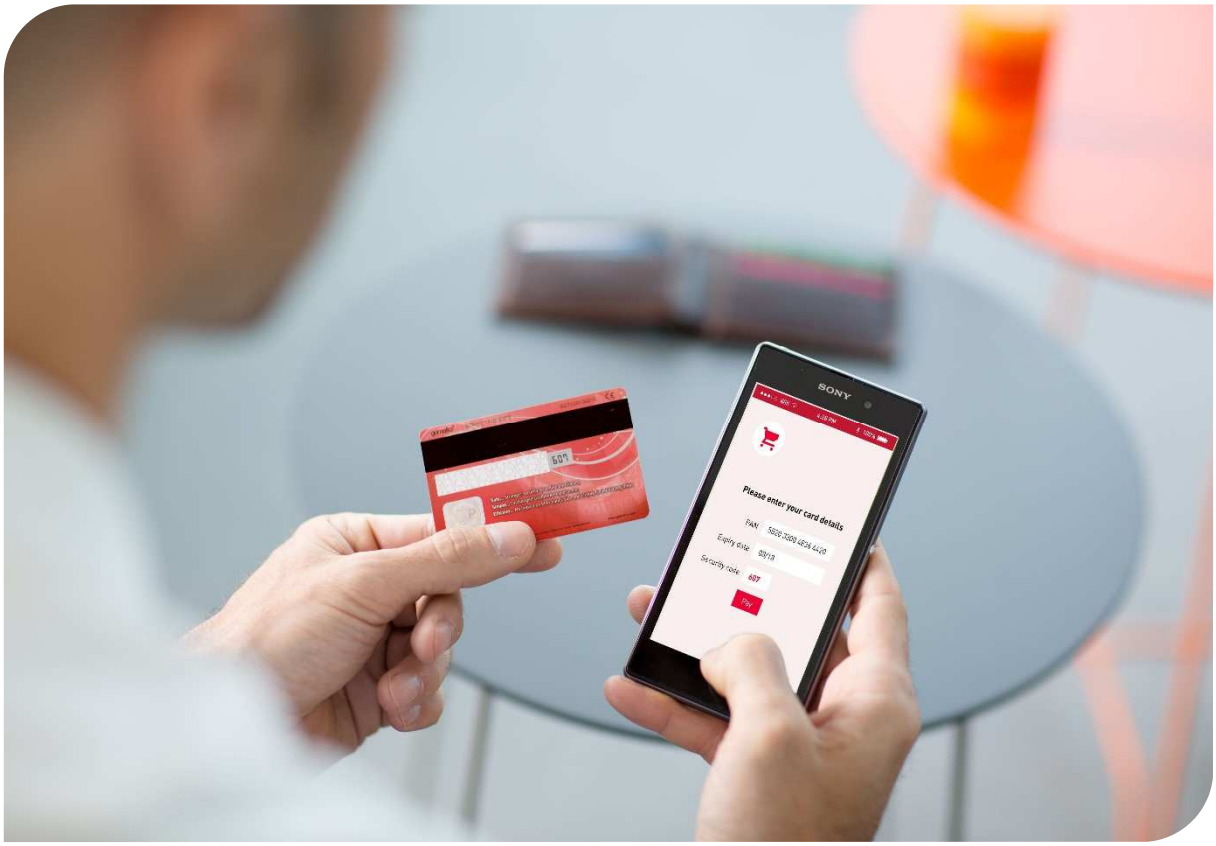
THALES

gemalto
a Thales company



Mobile SDK Library 1.7: Technical Integration Documentation – Revision A05

Card Companion SDK

Document Releases

Release	Date	Author	Modifications
A01	20/07/18	L. Szpieg	Initial revision
A02	30/07/18	L. Szpieg	Add SignatureCertAlias in CB token
A03	17/08/18	L. Szpieg	Change GetPinToken return object
A04	04/04/19	L. Szpieg	Add non CB token
A05	31/07/19	T. DOTHUONG	Add free field value to provisioning, rename methods for harmonization

Table of content

1 Introduction 4
1.1 Purpose 4
1.2 Lexical 4
1.3 Technical Overview 5
1.4 What's new 5
2 Use cases 6
2.1 PIN Distribution 6
2.1.1 PIN Distribution - Captcha with AC 6
2.1.2 PIN Distribution - Captcha with Token 7
2.2 PIN Definition 8
2.2.1 PIN Definition – HMAC 8
2.2.2 PIN Definition – Token 9
2.2.3 PIN Definition – Token CB 10
3 Integration with IDE 11
3.1 Android Studio Integration 11
3.2 XCode Integration 13
4 API References 14
4.1 PIN Distribution - Captcha with AC 14
4.1.1 Android 14
4.1.2 iOS 15
4.2 PIN Distribution - Captcha with Token 17
4.2.1 Android 17
4.2.2 iOS 18
4.3 PIN Definition – HMAC 20
4.3.1 Android 20
4.3.2 iOS 20
4.4 PIN Definition – Token 22
4.4.1 Android 22
4.4.2 iOS 22
4.5 PIN Definition – Token CB 24
4.5.1 Android 24
4.5.2 iOS 25
5 Appendix 27
5.1 PIN Distribution - Error Management 27
5.2 PIN Distribution Notification – Status Value 28
5.3 PIN Definition – Error Management 28
5.4 Token and Signature 29
5.4.1 Token Format 29
5.4.2 Token and Signature CB compliant 29
5.4.3 Token and Signature PCI-CPP compliant 29
5.5 PIN Definition – PIN selection using Keypad 30
5.6 HMAC Computation 31
5.7 Deprecated methods 31

1 Introduction

1.1 Purpose

This document describes the Card Companion SDK, and is the starting point of learning how to embed Card Companion features into a custom **Android** and **iOS** app.
The Card Companion SDK provided by Gemalto (a Thales company) provides two different services on most recent smartphones with iOS and Android: **PIN distribution** and **PIN definition**.

For PIN distribution schemes, the PIN code will be delivered either as a captcha image or as digit images, all images being scrambled with a specific algorithm (depending on customer configuration in the PIN Distribution system). These two methods require the cardholder to be authenticated. Therefore, the cardholder can provide a password (also called authentication code or AC in this document), known by himself/herself only, or a token has to be built and sent as a proof of the cardholder authenticity.

For PIN definition, the customer is free to choose his/her own PIN code. The desired PIN will be ciphered and sent to Thales's backend to be stored.

The document assumes that the reader has a basic understanding of the Card Companion solution.

1.2 Lexical

AC	Authentication Code
CB	French Banking
ECDH	Elliptic-Curve Diffie-Hellman
HMAC	Keyed-Hash Message Authentication Code
JSON	JavaScript Object Notation
PIN	Personal Identification Number
RSA	Rivest–Shamir–Adleman, public key cryptosystem
SDK	Software Development Kit
UI	User Interface

1.3 Technical Overview

Card Companion project is designed to provide multiple services with high added-value to cardholders through the e-banking mobile application.
Card Companion requires internet connectivity as all the services require communication with back-ends are hosted by Thales.

A simple representation of Card Companion SDK product could be as follow:

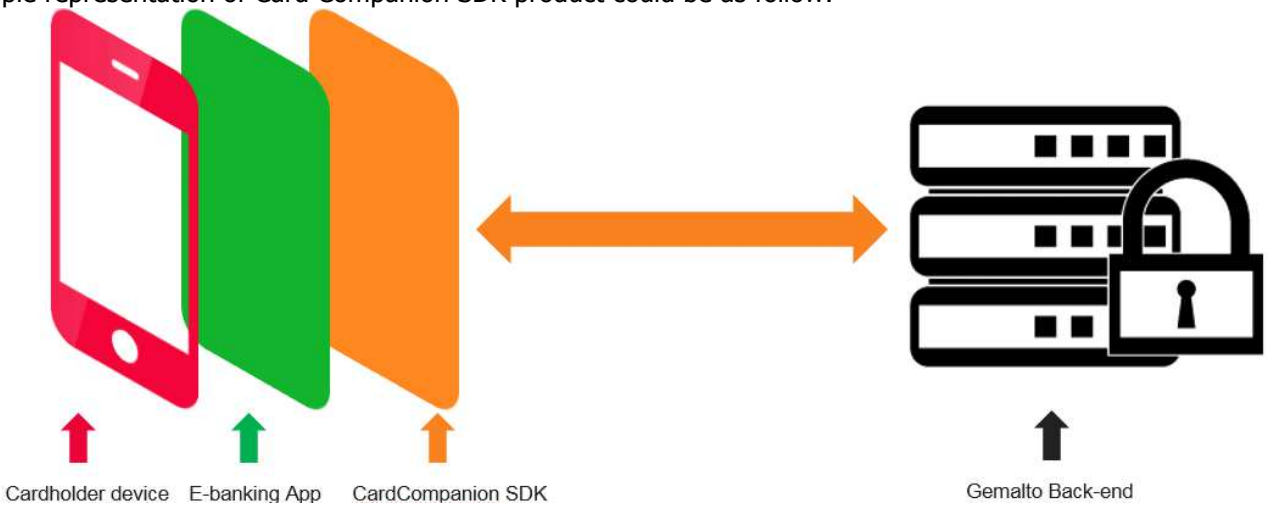


Figure 1 - Project Overview

The library provides API related to the card's PIN management such as definition of PIN code or distribution of the PIN. However, it doesn't manage any GUI (Graphical User Interface) component, which is in scope of the e-banking mobile application.

1.4 What's new

NEW

Card Companion SDK version 1.7 brings the possibility to optionally provide an additional free field along with the PIN captcha.
This feature allows the bank to bring custom messages to the card holder during PIN distribution.

Note: Get in touch with Thales project manager for the configuration about the activation and format of the free field value.

With version 1.7.0, some methods have been set to deprecated and replaced with new ones. These methods still remain in the library and are backward compatible but may be removed in future versions.

You may refer to section §5.7 Deprecated methods for the list of deprecated methods.

2 Use cases

To summarize, Card Companion SDK offers two services: **PIN Definition** and **PIN Distribution**, and each service has multiple variant such as:

- **PIN Distribution**
  - Cardholder authentication: managed by Thales using password (Authentication Code aka VISA/MC scenario) or managed by the customer using proof of authentication (token+signature aka. GIE CB scenario)
- **PIN Definition**
  - Cardholder authentication: using hmac (exchange of symmetric key necessary), using token+signature.
  - PIN format: using shared keypad coordinates or using PIN directly.

2.1 PIN Distribution

2.1.1 PIN Distribution - Captcha with AC

In this PIN Distribution scenario, the cardholder will input a password which has been previously provided by the bank. Then, the handset/device will display a captcha image containing the cardholder's PIN.

The cardholder authentication is managed by Thales Back-end thus an **authentication code must be provided to the Card Companion SDK**.

In the sequence flow below the authentication code is not shared to the customer and is directly retrieved by the e-banking mobile application from a customer's back-end. This process is called **implicit authentication**. But for sure, it could be possible to request cardholder to enter it in the mobile application then we talk about **explicit authentication**.

In any case, the cardholder's authentication code must be shared by the Bank to Thales during a provisioning (enrollment) step.

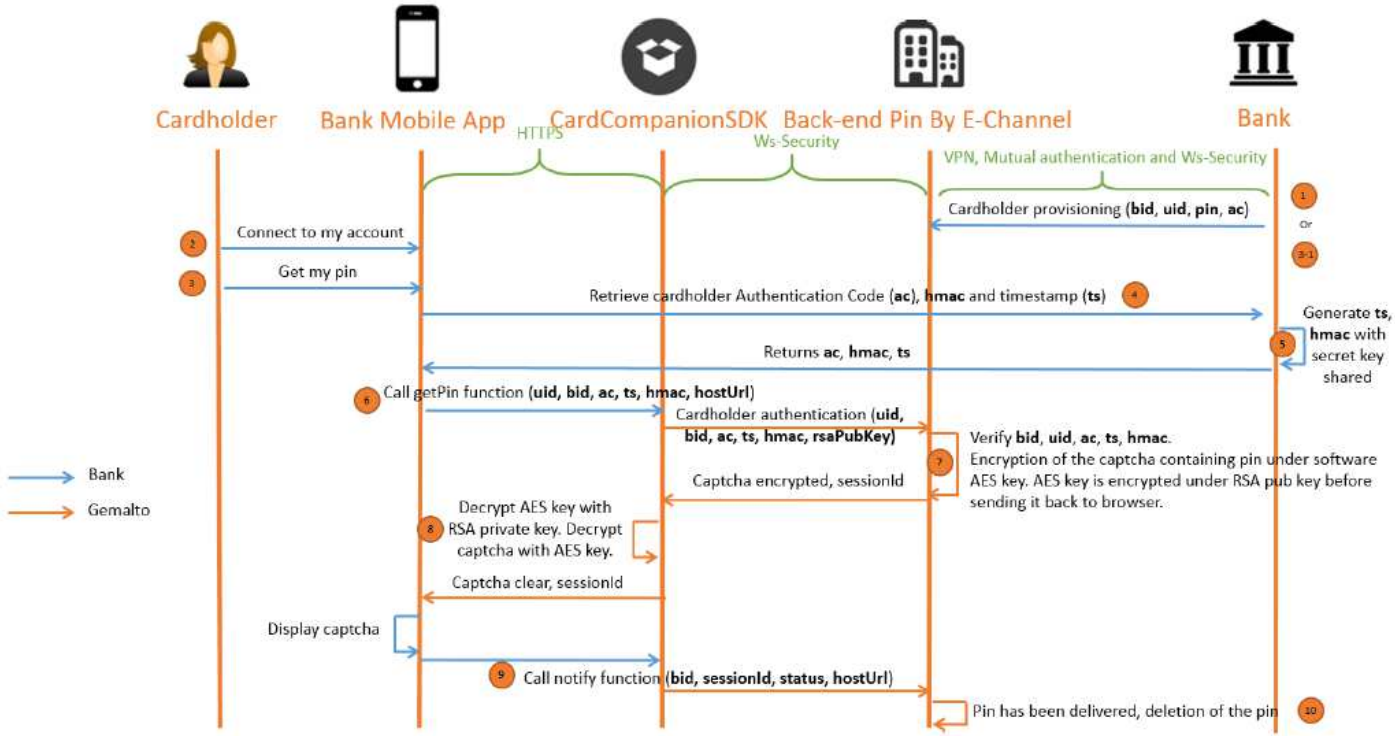


Figure 2 - PIN Distribution with AC

2.1.2 PIN Distribution - Captcha with Token

In this PIN Distribution scenario, the cardholder authentication is managed by the customer. The handset/device will display a captcha image containing the cardholder's PIN.

Thus, instead of an authentication code, the Card Companion SDK must receive a **token and a RSA signature** (aka proof of authentication). As the signature generation requires the use of the RSA private key, it must be generated by a back-end and not inside the app.

We also recommend to **generate the token in the back-end computing the signature** and not in the mobile application, to avoid timestamp issue due to mobile with incorrect time.

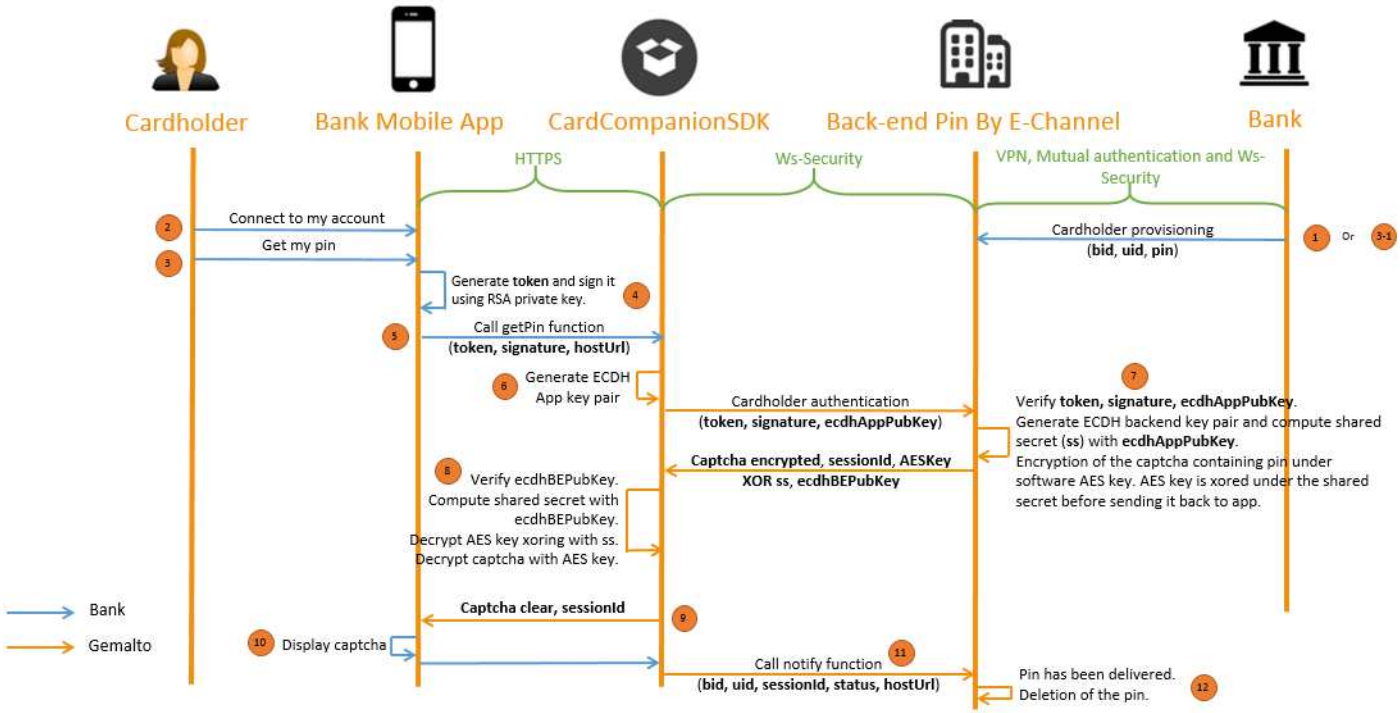


Figure 3 - PIN Distribution with Token

2.2 PIN Definition

2.2.1 PIN Definition – HMAC

In this PIN Definition scenario, a HMAC must be generated by a customer back-end and provided to Card Companion SDK. The HMAC will act as a proof of authentication. Because the HMAC generation requires the use of the shared symmetric key it must be generated by a back-end and not inside the app.

We also recommend to **generate the timestamp in the back-end computing the HMAC** and not in the mobile application, to avoid timestamp issue due to mobile with incorrect time.

In addition the PIN format shared with the SDK must be in clear (ex: 1234).

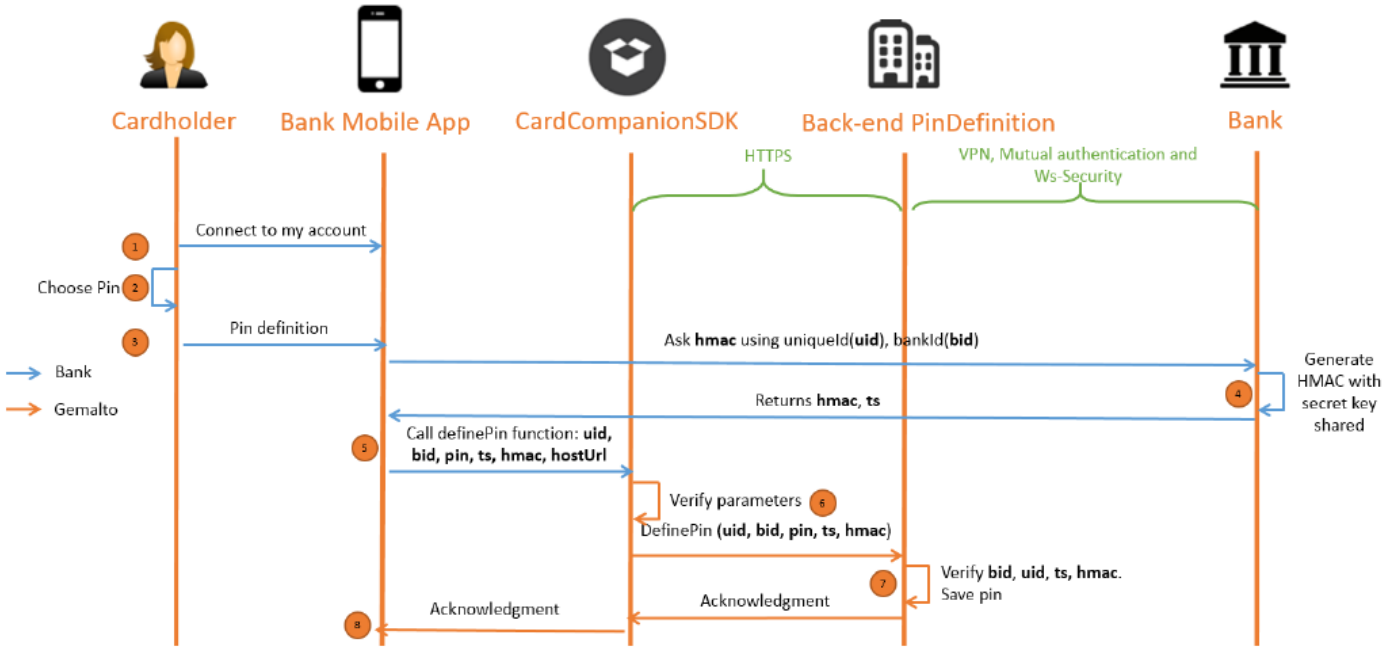


Figure 4 - PIN Definition HMAC

2.2.2 PIN Definition – Token

In this PIN Definition scenario, a token and signature must be generated by a customer back-end and provided to Card Companion SDK. The signature will act as a proof of authentication. Because the signature generation requires the use of the RSA private key it must be generated by a back-end and not inside the app.

We also recommend to **generate the token in the back-end computing the hmac** and not in the mobile application, to avoid timestamp issue due to mobile with incorrect time.

In addition the PIN format shared with the SDK must be in clear (ex: 1234).

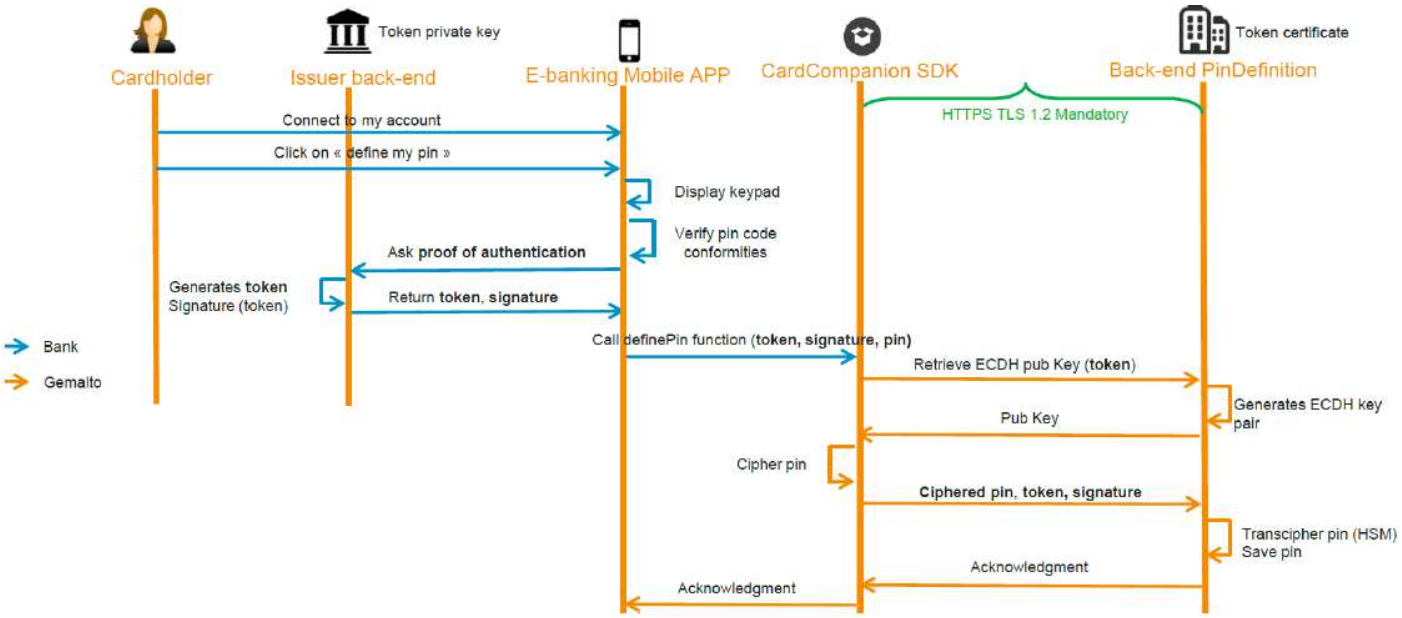


Figure 5 - PIN Definition with Token

2.2.3 PIN Definition – Token CB

In this PIN Definition scenario, a token and signature must be generated by a customer back-end and provided to Card Companion SDK. The signature will act as a proof of authentication. Because the signature generation requires the use of the RSA private key it must be generated by a back-end and not inside the app.

We also recommend to **generate the token in the back-end computing the signature** and not in the mobile application, to avoid timestamp issue due to mobile with incorrect time. In addition, because GIE CB doesn't allow Smartphones to send the PIN (even ciphered) to the PINDefinition backend the **PIN format** send by the mobile must be the **position in the keypad coordinates table shared by Thales' s PINDefinition back-end**.

Example: keypad coordinates [9,5,1,3,5,7,4,0,8,6,2] shared by back-end and PIN selected by cardholder 1234. Coordinates provided to Card Companion SDK must be 2936 – index in the table -).

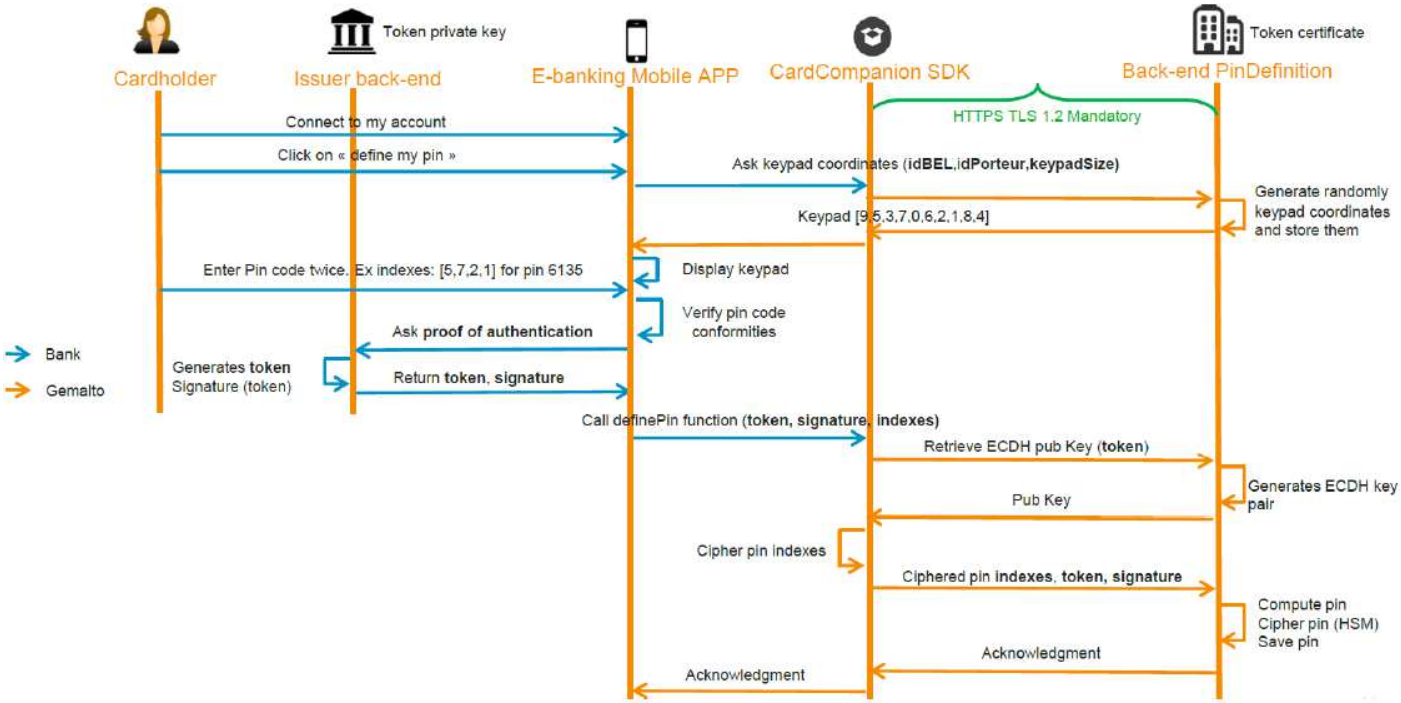


Figure 6 - PIN Definition with Token CB

3 Integration with IDE

Card Companion SDK library is provided for both iOS and Android. They will answer to the use cases described with some specificities. Depending on the platform you are developing, please refer to the relevant chapter.

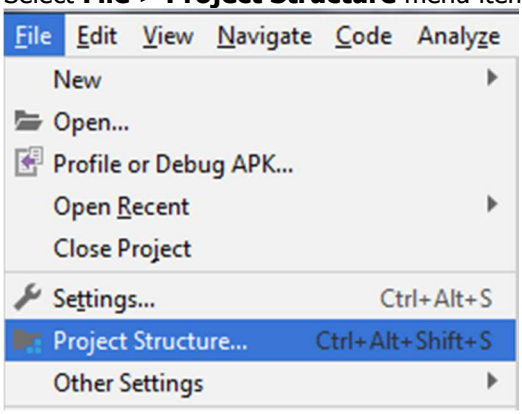
For Android, a library **.aar** will be provided, whereas for iOS the delivery will contains a library in **.a** extension with the header files associated. For each of them you will find a sample application calling all the SDK function for each use cases.

These samples project will be shared by Thales's project manager upon request.

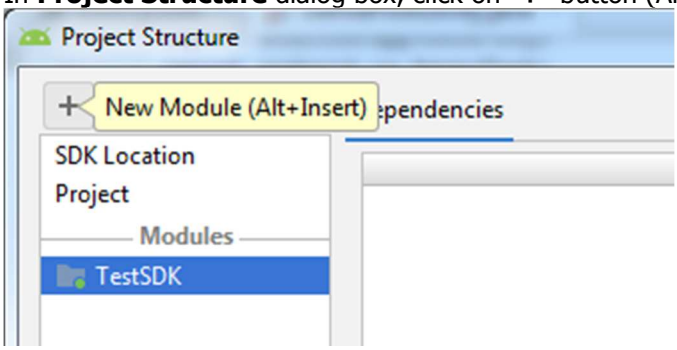
3.1 Android Studio Integration

Using Android Studio as development IDE, please follow these steps to import Card Companion SDK library in your project:

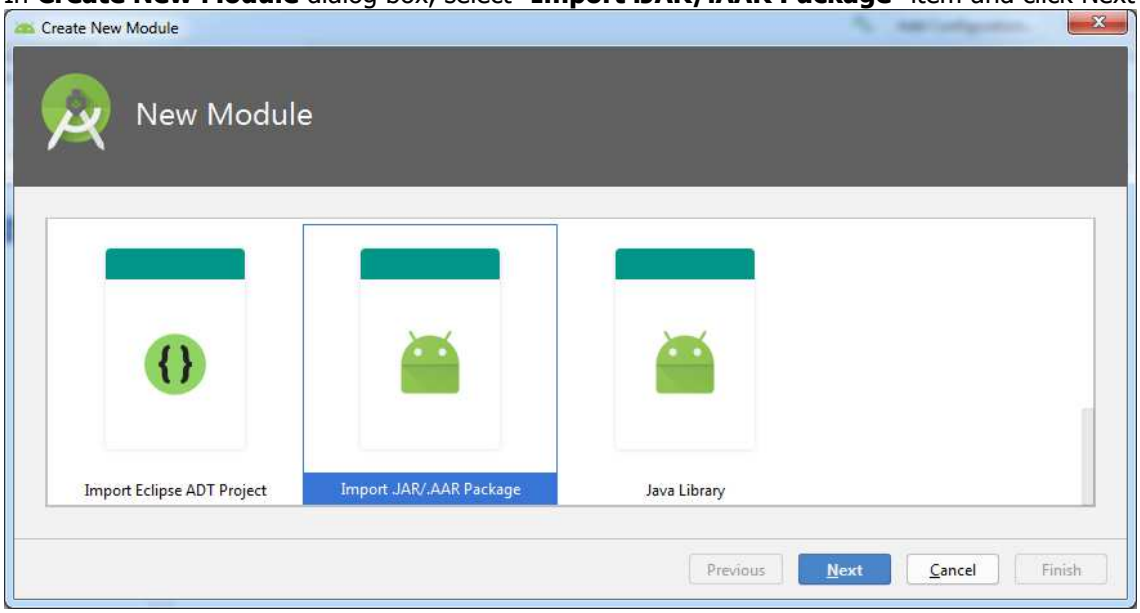
1. Select **File** > **Project Structure** menu item (Ctrl + Alt + Shift +S shortcut).



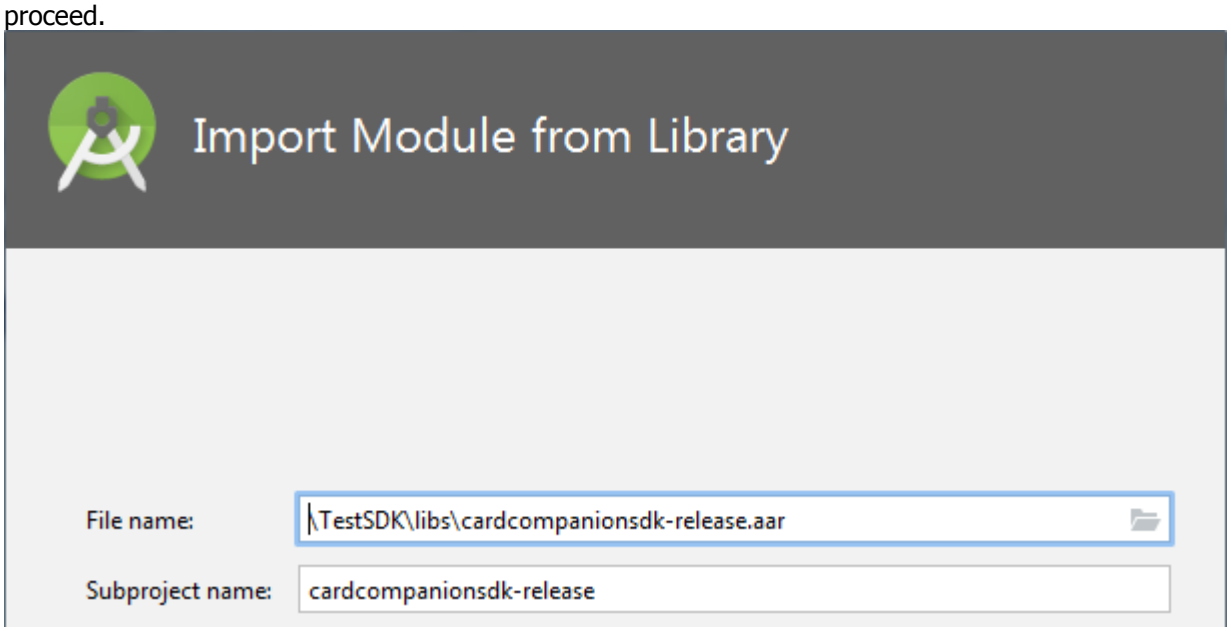
2. In **Project Structure** dialog box, click on "**+**" button (Alt + Insert shortcut).



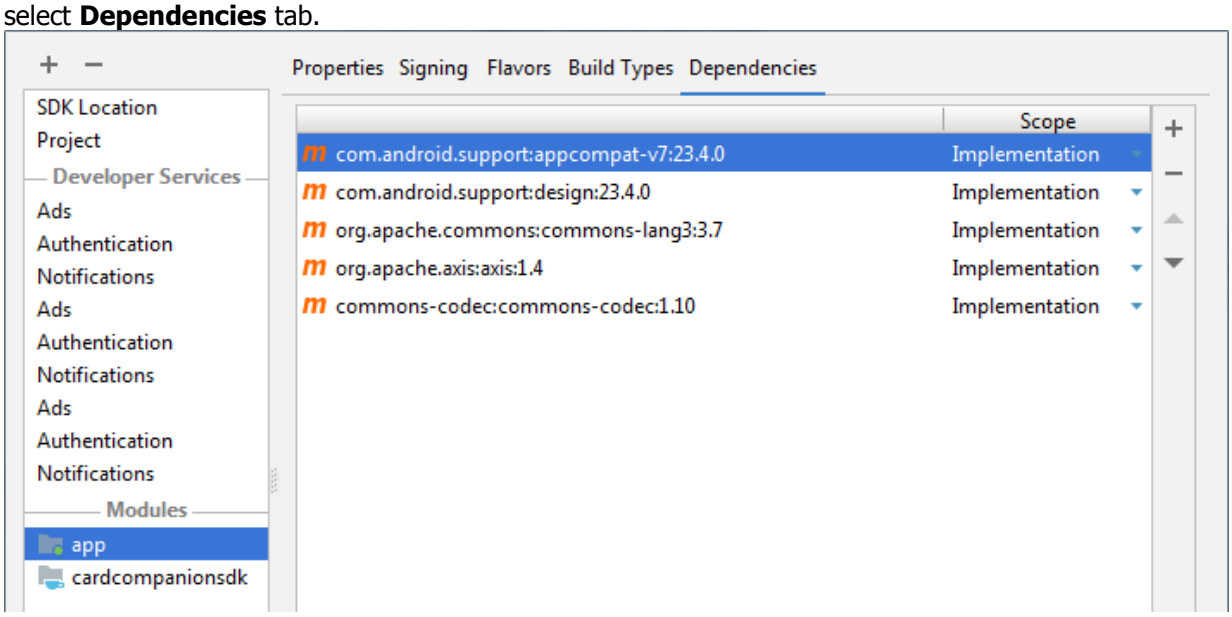
3. In **Create New Module** dialog box, select "**Import .JAR/.AAR Package**" item and click Next to continue.



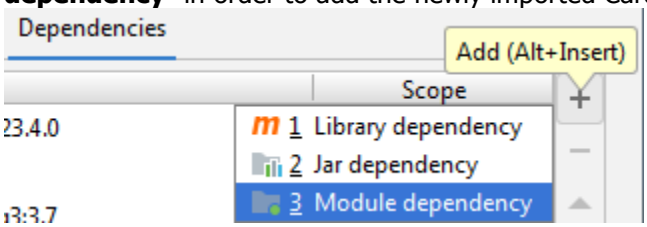
4. Browse for Card Companion SDK aar file, set subproject name (e.g. cardcompanionsdk) and click **Finish** to proceed.



5. The **Project Structure** dialog box will be updated accordingly. Select **app** node in **Modules** section, and select **Dependencies** tab.



6. Click on "+" button (Alt + Insert shortcut) in the under the Dependencies tab and select "**Module dependency**" in order to add the newly imported CardCompanionSDK module as dependency.



7. Click OK to proceed.

8. Add these dependencies to your project in your **build.gradle** file:

```
depencencies {
    implementation 'com.google.code.gson:gson:2.6.2'
    implementation 'commons-codec:commons-codec:1.10'
    implementation 'com.android.support:appcompat-v7:23.4.0'
    implementation 'org.apache.commons:commons-lang3:3.4'
}
```

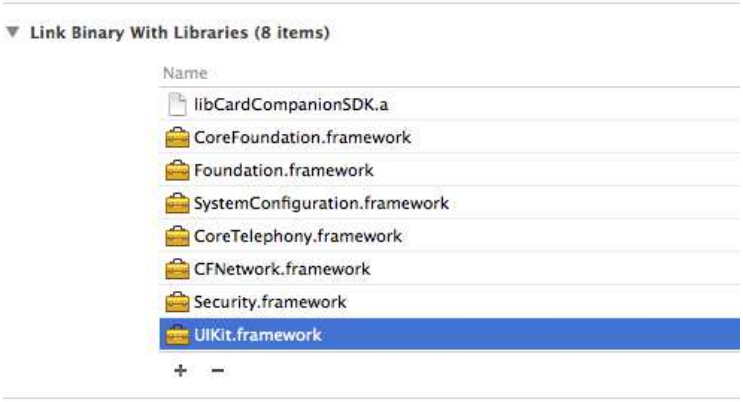
3.2 XCode Integration

In iOS, there are multiple ways to import a library aar inside an xCode project. One of the solution is:

1. Add Lib (.a) and dependency framework to your build target

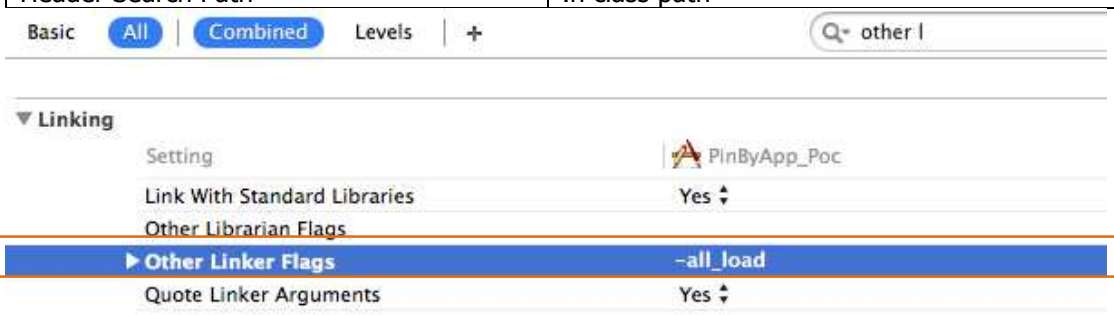
Build Phases	Value
Link Binary With Libraries	Add libCard CompanionLib.a

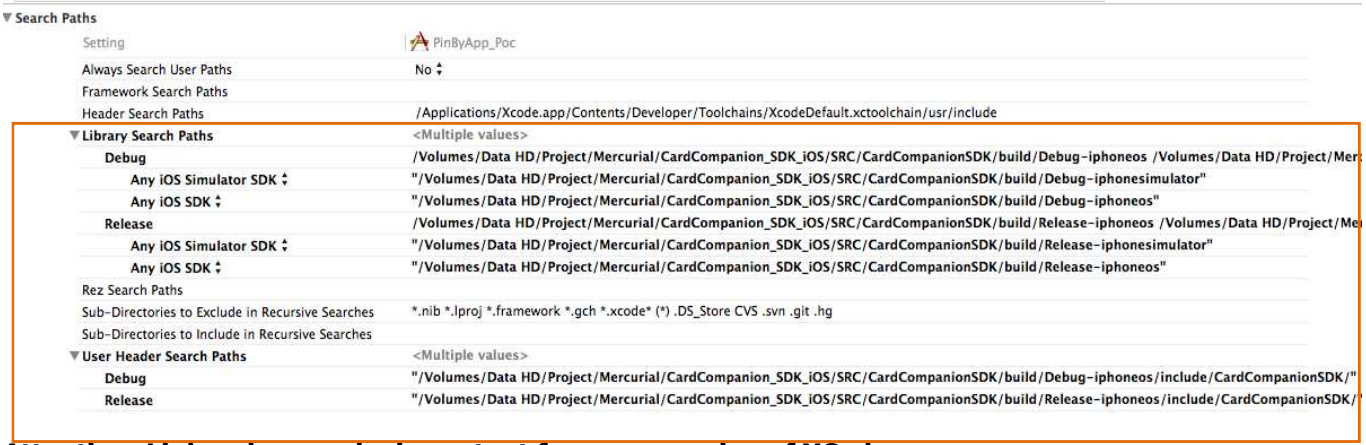
All the libraries below are mandatory for the project.



2. Configuration :

Build Settings	Value
other link flag	-all_load
Library Search Path	Library file path
Header Search Path	.h class path





Attention: Link order may be important for some version of XCode.

3. Import the SDK header from your prefix header to make Card Companion APIs available

```
#import "Card CompanionSDK.h"
OR
#import "PINCodeRequest.h"
```

4 API References

4.1 PIN Distribution - Captcha with AC

From app point of view, the PIN distribution is managed with the following commands:

- Request for PIN Captcha
- Notification to the server

Complete scenario description can be found in section: §2.1.1 PIN Distribution - Captcha with AC.

4.1.1 Android

In order to complete this scenario, you will need to call the following functions:

- `PINDistribution.postGetPINCaptchaRequest`
  - Send a request to PINDistribution Thales platform to retrieve the cardholder's PIN as a captcha image (png).
- `PINDistribution.postNotifyRequest`
  - Send a request to PINDistribution Thales platform to notify the captcha image has been successfully displayed to cardholder.

postGetPINCaptchaRequest – Input parameters			
Name	**Type**	**Attribute**	**Detail**
parameters	ProvisioningParameters	bankId	Name of the bank in the PINDistribution system. This name should be provided by Thales Project Manager
		uniqueId	Cardholder's unique identifier. Will be used by PINDistribution system to identify cardholder request
		authCode	Cardholder's password. This will be used by PINDistribution system to authenticate cardholder request
		timestamp	Timestamp in EPOCH format (ex. 1475255010)
		hashMac	Result of SHA_256 on String bankId+uniqueId+timestamp. A symmetric key needs to be shared between customer and Thales. This key should be provided by Thales Project Manager. Algorithm implementation can be found:
hostUrl	String		PINDistribution system URL. This url should be provided by Thales Project Manager
context	Context		Android application context

postGetPINCaptchaRequest – Output parameters		
Type	**Attribute**	**Detail**
ProvisioningResponse	errorCode	Request process status. Error code table can be found at: §5.1 PIN Distribution - Error Management
	pin	PIN captcha image in Base64 string (png image).
	sessionId	Unique request identifier. Must be used in next function call: postNotifyRequest.
	freeField	Optional additional information provided with the PIN.

Code sample:

```
//Build parameters
ProvisioningParameters provisioningParameters = new ProvisioningParameters(bankId, cardholderId,
authCode, timestamp, hmac);

//Send Request
ProvisioningResponse provisioningResponse =
PINDistribution.postGetPINCaptchaRequest(getApplicationContext(), provisioningParameters,
backEndUrl);
```

postNotifyRequest – Input parameters			
Name	**Type**	**Attribute**	**Detail**
parameters	NotifyParameters	bankId	Name of the bank in the PINDistribution system. This name should be provided by Thales Project Manager
		uniqueId	Cardholder's unique identifier. Will be used by PINDistribution system to identify cardholder request
		sessionId	Value returned in ProvisioningResponse.sessionId
		status	Each status value has a specific meaning. If PIN captcha image has been successfully displayed, this value must be set to 0. Complete status list can be found:
hostUrl	String		PINDistribution system URL. This url should be provided by Thales Project Manager
context	Context		Android application context

postNotifyRequest – Output parameters		
Type	**Attribute**	**Detail**
NotifyResponse	errorCode	Notification process status. Error code table can be found at: §5.2 PIN Distribution Notification – Status Value
	errorMessage	Notification process message.

Code sample:

```
//Build parameters
NotifyParameters notifyParameters = new NotifyParameters(sessionId, bankId, cardholderId,
status);

//Notify that Pin was received properly
NotifyResponse notifyResponse = PINDistribution.postNotifyRequest(getApplicationContext(),
notifyParameters, backEndUrl);
```

4.1.2 iOS

In order to complete this scenario, you will need to call the following functions:

- `[PinCodeRequest postGetPINRequest]`
  - Send a request to PINDistribution Thales platform to retrieve the cardholder's PIN as a captcha image (png).
- `[PinCodeRequest postNotifyRequest]`
  - Send a request to PINDistribution Thales platform to notify the captcha image has been successfully displayed to cardholder.

postGetPINRequest – Input parameters		
Name	**Type**	**Detail**
bankId	NSString	Name of the bank in the PINDistribution system. This name should be provided by Thales Project Manager
uniqueId	NSString	Cardholder's unique identifier. Will be used by PINDistribution system to identify cardholder request
ac	NSString	Cardholder's password. Will be used by PINDistribution system to authenticate cardholder request
timestamp	NSString	Timestamp in EPOCH format (ex. 1475255010)
hashMac	NSString	Result of SHA_256 on String bankId+uniqueId+timestamp. A symmetric key needs to be shared between customer and Thales. This key should be provided by Thales Project Manager. Algorithm implementation can be found:
hostURL	NSString	PINDistribution system URL. This url should be provided by Thales Project Manager

postGetPINRequest – Output parameters		
Type	**Attribute**	**Detail**
PINCodeProvisioningJResponse	errorCode	Request process status. Error code table can be found at: §5.1 PIN Distribution - Error Management
	imageData	PIN captcha image in Base64 string (png image).
	sessionId	Unique request identifier. Must be used in next function call: postNotifyRequest.
	freeField	Optional additional information provided with the PIN.

Code sample:

```
//Get Time Stamp
NSNumber *ts = [CardCompanionUtil getTimestamp];

//Compute HMAC
NSString *hexStr = [CardCompanionUtil stringToHex:[NSString stringWithFormat:@"%@%@%@",
bank,unique,ts]];
NSString *hmac = [CardCompanionUtil calculHashMacValue:hashMac_key str:hexStr];

//Send Request
PinCodeProvisioningJResponse *jresponse = [PinCodeRequest postGetPinRequest:bank uniqueId:unique
ac:pass timestamp:ts hmac:hmac hostURL:pathUrl];
```

postNotifyRequest – Input parameters		
Name	**Type**	**Detail**
bankId	NSString	Name of the bank in the PINDistribution system. This name should be provided by Thales Project Manager
uniqueId	NSString	Cardholder's unique identifier. Will be used by PINDistribution system to identify cardholder request
sessionId	NSString	Value returned in ProvisioningResponse.sessionId
status	GETPIN_STATUS	Each status value has a specific meaning. If PIN captcha image has been successfully displayed, this value must be set to 0. Complete status list can be found:
hostURL	NSString	PINDistribution system URL. This url should be provided by Thales Project Manager

postNotifyRequest – Output parameters		
Type	**Attribute**	**Detail**
JResponse	errorCode	Request process status. Error code table can be found at: §5.2 PIN Distribution Notification – Status Value
	errorMessage	Request process message.

Code sample:

```
//Notify that Pin was received properly
JResponse *j = [PinCodeRequest postNotifyRequest:_getPinBid.text uniqueId:_getPinUid.text
sessionId:_getPinSessionID.text status: 0 hostURL:_getPinURLNotify.text];
```

4.2 PIN Distribution - Captcha with Token

From app point of view, the PIN distribution is managed with the following commands:

- Request for PIN Captcha
- Notification to the server

Complete scenario description can be found in the previous section: §2.1.2 PIN Distribution - Captcha with Token.

Refer to §5.4 Token and Signature for Token format

4.2.1 Android

In order to complete this scenario, you will need to call the following functions:

- `PINDistribution.postGetPINCaptchaTokenRequest`
  - Send a request to PINDistribution Thales platform to retrieve the cardholder's PIN as a captcha image (png).
- `PINDistribution.postNotifyTokenRequest`
  - Send a request to PINDistribution Thales platform to notify the captcha image has been successfully displayed to cardholder.

postGetPINCaptchaTokenRequest – Input parameters			
Name	**Type**	**Attribute**	**Detail**
parameters	TokenProvisioningParameters	token	Token in JSON format as described in the section: §5.4.1 Token Format
		signature	Signature of the token provided. Generation is explained in the following section of the document:
hostUrl	String		PINDistribution system URL. This URL should be provided by Thales Project Manager
context	Context		Android application context

postGetPINCaptchaTokenRequest – Output parameters		
Type	**Attribute**	**Detail**
ProvisioningResponse	errorCode	Request process status. Error code table can be found at: §5.1 PIN Distribution - Error Management
	PIN	PIN captcha image in Base64 string (png image).
	sessionId	Unique request identifier. Must be used in next function call: postNotifyTokenRequest.
	freeField	Optional additional information provided with the PIN.

Code sample:

```
//Build parameters
TokenProvisioningParameters provisioningParameters = new TokenProvisioningParameters(token,
signature);

//Send Request
GetPINTokenResponse getPINTokenResponse =
PINDistribution.postGetPINCaptchaTokenRequest(getApplicationContext(),
TokenProvisioningParameters, hostUrl);
```

postNotifyTokenRequest – Input parameters			
Name	**Type**	**Attribute**	**Detail**
parameters	TokenNotifyParameters	token	Token in JSON format as described in the section: §5.4.1 Token Format
		sessionId	Value returned in ProvisioningResponse.sessionId
		status	Each status value has a specific meaning. If PIN captcha image has been successfully displayed, this value must be set to 0. Complete status list can be found: §5.2 PIN Distribution Notification – Status Value

hostUrl	String		PINDistribution system URL. This url should be provided by Thales Project Manager
context	Context		Android application context

postNotifyTokenRequest – Output parameters		
Type	**Attribute**	**Detail**
TokenNotifyResponse	errorCode	Request treatment status. Complete error code table can be found:
	errorMessage	Request process message.

Code sample:

```
//Build parameters
TokenNotifyParameters notifyParameters = new TokenNotifyParameters(token, sessionId, status);

//Notify that Pin was received properly
TokenNotifyResponse notifyResponse =
PINDistribution.postNotifyTokenRequest(getApplicationContext(), notifyParameters ,backendUrl);
```

4.2.2 iOS

In order to complete this scenario, you will need to call the following functions:

- `[PinCodeRequest postGetPINCaptchaTokenRequest]`
  - Send a request to PINDistribution Thales platform to retrieve the cardholder's PIN as a captcha image (png).
- `[PinCodeRequest postNotifyTokenRequest]`
  - Send a request to PINDistribution Thales platform to notify the captcha image has been successfully displayed to cardholder.

postGetPINCaptchaTokenRequest – Input parameters		
Name	**Type**	**Detail**
token	NSString	Token in JSON format as described in the section: §5.4.1 Token Format
signature	NSString	Signature of the token provided. Generation is explained in the following section of the document:
hostURL	NSString	PINDistribution system URL. This url should be provided by Thales Project Manager

postGetPINCaptchaTokenRequest – Output parameters		
Type	**Attribute**	**Detail**
PINCodeProvisioningJResponse	errorCode	Request process status. Error code table can be found at: §5.1 PIN Distribution - Error Management
	imageData	PIN captcha image in Base64 string (png image).
	sessionId	Unique request identifier. Must be used in next function call: postNotifyTokenRequest.
	freeField	Optional additional information provided with the PIN.

Code sample:

```
//Get Time Stamp
NSString *timestamp = [[CardCompanionUtil getTimestamp] stringValue];

//Get Transaction ID
NSString *transactionId = [NSString stringWithFormat:@"%d", (1 +
arc4random_uniform(1000000000))];

//Build Token
NSString *token = [NSString
stringWithFormat:@"{\"IdBEL\":\"%@\",\"IdPorteur\":\"%@\",\"IdFournisseur\":\"%@\",\"IdTransacti
on\":\"%@\",\"Timestamp\":\"%@\",\"Type\":\"%@\"}", bankId, uniqueId, idFournisseur,
transactionId, timestamp, type];
NSString *signature = [Utils signWithRSA:token];
```

```
//Send Request
PinCodeProvisioningJResponse * jresponse = [PinCodeRequest postGetPinCaptchaTokenRequest:token
signature:signature hostURL:url];
```

postNotifyTokenRequest – Input parameters		
Name	**Type**	**Detail**
bankId	NSString	Name of the bank in the PINDistribution system. This name should be provided by Thales Project Manager
uniqueId	NSString	Cardholder's unique identifier. Will be used by PINDistribution system to identify cardholder request
sessionId	NSString	Value returned in ProvisioningResponse.sessionId
status	GETPIN_STATUS	Each status value has a specific meaning. If PIN captcha image has been successfully displayed, this value must be set to 0. Complete status list can be found:
hostURL	NSString	PINDistribution system URL. This url should be provided by Thales Project Manager

postNotifyTokenRequest – Output parameters		
Type	**Attribute**	**Detail**
JResponse	errorCode	Request process status. Error code table can be found at: §5.2 PIN Distribution Notification – Status Value
	errorMessage	Request process message.

Code sample:

```
//Notify that Pin was received properly
JResponse *jresponse = [PinCodeRequest postNotifyTokenRequest:token
sessionId:_getPinTokenSessionID.text status:status hostURL:hostURL];
```

4.3 PIN Definition – HMAC

From app point of view, the PIN definition is managed with the following command:

- Request for PIN Definition

Complete scenario description can be found in the previous section: §2.2.1 PIN Definition – HMAC.

4.3.1 Android

In order to complete this scenario, you will need to call the following function:

- `PINDefinition.postDefinePINRequest`
  - Send the PIN to the Thales's PINDefinition system

postDefinePINRequest – Input parameters			
Name	**Type**	**Attribute**	**Detail**
parameters	DefinitionParameters	bankId	Name of the bank in the PINDefinition system. This name should be provided by Thales Project Manager
		uniqueId	Cardholder's unique identifier. Will be used by PINDefinition system to identify cardholder request
		timestamp	Timestamp in EPOCH format (ex. 1475255010)
		hashMac	Result of SHA_256 on String bankId+uniqueId+timestamp. A symmetric key needs to be shared between customer and Thales. This key should be provided by Thales Project Manager. Algorithm implementation can be found:
		pin	PIN in clear String format (ex: 1234)
hostUrl	String		PINDefinition system URL. This url should be provided by Thales Project Manager
context	Context		Android application context

postDefinePINRequest – Output parameters		
Type	**Attribute**	**Detail**
DefinitionResponse	errorCode	Request process status. Error code table can be found at: §5.3 PIN Definition – Error Management
	errorMessage	Request process message.

Code sample:

```
//Build Parameters
DefinitionParameters definitionParameters = new DefinitionParameters(bankId, uid, timestamp,
hmac, pin);

//Send Request
DefinitionResponse definePINResponse =
PINDefinition.postDefinePINRequest(getApplicationContext(),definitionParameters,hostUrl);
```

4.3.2 iOS

In order to complete this scenario, you will need to call the following functions:

- `[PinCodeRequest postDefinePinRequest]`
  - Send the PIN to Thales's PINDefinition system.

postDefinePINRequest – Input parameters		
Name	**Type**	**Detail**
bankId	NSString	Name of the bank in the PINDistribution system. This name should be provided by Thales Project Manager
uniqueId	NSString	Cardholder's unique identifier. Will be used by PINDefinition system to identify cardholder request
timestamp	NSString	Timestamp in EPOCH format (ex. 1475255010)

hashMac	NSString	Result of SHA_256 on String bankId+uniqueId+timestamp. A symmetric key needs to be shared between customer and Gemalto. This key should be provided by Gemalto Project Manager. Algorithm implementation can be found: Hmac samples
PIN	NSString	PIN in clear String format (ex: 1234)
hostURL	NSString	PINDefinition system URL. This url should be provided by Thales Project Manager

postDefinePINRequest – Output parameters		
Type	**Attribute**	**Detail**
JResponse	errorCode	Request process status. Error code table can be found at: §5.3 PIN Definition – Error Management
	errorMessage	Request process message.

Code sample:

```
//Send Request
JResponse * definePinJResponse = [PinCodeRequest postDefinePinRequest: bankId uniqueId: uniqueId
timestamp: timestamp hmac: hmac hostURL: hostURL];
```

4.4 PIN Definition – Token

From app point of view, the PIN definition is managed with the following command:

- Request for PIN Definition

Complete scenario description can be found in the previous section: §2.2.2 PIN Definition – Token.

4.4.1 Android

In order to complete this scenario, you will need to call the following function:

- `PINDefinition.postDefinePINECDHRequest`
  - Send the PIN to the Thales's PINDefinition system

postDefinePINECDHRequest – Input parameters			
Name	**Type**	**Attribute**	**Detail**
parameters	DefinitionTokenParameters	token	Token in JSON format as described in the section: §5.4.1 Token Format
		signature	Signature of the token provided. Generation is explained in the following section of the document:
		pin	PIN in clear String format (ex: 1234)
hostUrl	String		PINDefinition system URL. This url should be provided by Thales Project Manager
context	Context		Android application context

postDefinePINECDHRequest – Output parameters		
Type	**Attribute**	**Detail**
DefinitionResponse	errorCode	Request process status. Error code table can be found at: §5.3 PIN Definition – Error Management
	errorMessage	Request process message.

Code sample:

```
//Build Parameters
DefinitionTokenParameters definitionTokenParameters = new DefinitionTokenParameters(token,
signature, pin);

//Send Request
GetPINTokenResponse definePINResponse =
PINDefinition.postDefinePINECDHRequest(getApplicationContext(),definitionTokenParameters,
hostUrl);
```

4.4.2 iOS

In order to complete this scenario, you will need to call the following functions:

- `[PinCodeRequest postDefinePinRequestECDH]`
  - Send the pin to the Gemalto's PinDefinintion system

postDefinePinRequestECDH – Input parameters		
Name	**Type**	**Detail**
token	NSString	Token in JSON format as described in the section: §5.4.1 Token Format
signature	NSString	Signature of the token provided. Generation is explained in the following section of the document
pin	NSString	PIN in clear format (ex: 1234)
hostURL	NSString	PINDefinition system URL. This url should be provided by Thales Project Manager

postDefinePinRequestECDH – Output parameters		
Type	**Attribute**	**Detail**
DefinePinTokenRequest	errorCode	Request process status. Error code table can be found at: §5.3 PIN Definition – Error Management
	errorMessage	Request process message.

Code sample:

```
//Send Request
DefinePinTokenRequest * definePinJResponse = [PinCodeRequest postDefinePinRequestECDH: token
signature: signature pin: pin hostURL: hostUrl]
```

4.5 PIN Definition – Token CB

From app point of view, the PIN definition is managed with the following commands:

- Get KeyPad coordinates
- Request for PIN Definition

Complete scenario description can be found in the previous section: §2.2.3 PIN Definition – Token CB.

4.5.1 Android

In order to complete this scenario, you will need to call the following functions:

- `PINDefinition. postGetKeyPadCoordinatesRequest`
  - Request shared keypad coordinates from PINDefinition Back-end.
- `PINDefinition.postDefinePINECDHCBRequest`
  - Send the PIN indexes to the Thales's PINDefinintion system

postGetKeyPadCoordinatesRequest – Input parameters			
Name	**Type**	**Attribute**	**Detail**
Parameters	KeyPadCoordinatesParameters	bankId	Name of the bank in the PINDefinition system. This name should be provided by Thales Project Manager. It corresponds to *IdBEL* parameter.
		uniqueId	Cardholder's unique identifier. Will be used by PINDefinition system to identify cardholder request. It corresponds to *IdPorteur* parameter.
		keypadSize	Size of the keypad coordinates table required.
hostUrl	String		PINDefinition system URL. This url should be provided by Thales Project Manager
context	Context		Android application context

postGetKeyPadCoordinatesRequest – Output parameters		
Type	**Attribute**	**Detail**
KeyPadCoordinatesResponse	errorCode	Request process status. Error code table can be found at: §5.3 PIN Definition – Error Management
	keypad	List<String> containing the digits. If keypadSize was over 10 some of the item in the list will contain empty string.
	errorMessage	Request process message.

Code sample:

```
//Build Parameters
KeyPadCoordinatesParameters keyPadCoordinatesParameters = new
KeyPadCoordinatesParameters(bankId, uniqueId, 10);

//Get KeyPad coordinates
KeyPadCoordinatesResponse keyPadCoordinatesResponse = PINDefinition.
postGetKeyPadCoordinatesRequest(getApplicationContext(),keyPadCoordinatesParameters, hostUrl);
```

postDefinePINECDHCBRequest – Input parameters			
Name	**Type**	**Attribute**	**Detail**
parameters	DefinitionTokenParameters	token	Token in JSON format as described in the section: §5.4.1 Token Format
		signature	Signature of the token provided. Generation is explained in the following section of the document:
		indexes	PIN coordinates in table provided by PINDefinition back-end in postGetKeyPadCoordinatesRequest response.
hostUrl	String		PINDefinition system URL. This url should be provided by Thales Project Manager
context	Context		Android application context

postDefinePINECDHCBRequest – Output parameters		
Type	**Attribute**	**Detail**
DefinitionResponse	errorCode	Request process status. Error code table can be found at: §5.3 PIN Definition – Error Management
	errorMessage	Request process message.

Code sample:

```
//Build Parameters
DefinitionTokenParameters definitionTokenParameters = new DefinitionTokenParameters(token,
signature, PINIndexesChosen.toString());

//Send Request
DefinitionResponse definePINResponse =
PINDefinition.postDefinePINECDHCBRequest(getApplicationContext(),definitionTokenParameters,
hostUrl);
```

4.5.2 iOS

In order to complete this scenario, you will need to call the following functions:

- `[PinCodeRequest postGetCoordinateKeypad]`
  - Request shared keypad coordinates from PinDefinition Back-end.
- `[PinCodeRequest postDefineTokenRequestECDH_CB]`
  - Send the pin indexes to the Gemalto's PinDefinintion system.

postGetCoordinateKeypad – Input parameters		
Name	**Type**	**Detail**
bankId	NSString	Name of the bank in the PINDefinition system. This name should be provided by Thales Project Manager. It corresponds to *IdBEL* parameter.
uniqueId	NSString	Cardholder's unique identifier. Will be used by PINDefinition system to identify cardholder request. It corresponds to *IdPorteur* parameter.
keypadSize	NSString	Size of the keypad coordinates table required.
hostURL	NSString	PINDefinition system URL. This url should be provided by Thales Project Manager

postGetCoordinateKeypad – Output parameters		
Type	**Attribute**	**Detail**
GetKeypadCoordJResponse	errorCode	Request process status. Error code table can be found at: §5.3 PIN Definition – Error Management
	keyPad	String containing the digits. If keypadSize was over 10 some of the item in the list will contain empty string.
	errorMessage	Request process message.

Code sample:

```
//Get KeyPad coordinates
GetKeypadCoordJResponse * keypadJResponse = [PinCodeRequest postGetCoodinateKeypad: bankId
uniqueId: uniqueId hostURL: hostURL keypadSize: keypadSize];
```

postDefinePinRequestECDH_CB – Input parameters		
Name	**Type**	**Detail**
token	NSString	Token in JSON format as described in the section: §5.4.1 Token Format
signature	NSString	Signature of the token provided. Generation is explained in the following section of the document: Token and Signature.

indexes	NSString	Pin coordinates in table provided by PinDefinition back-end in GetKeypadCoordJResponse response
hostURL	NSString	PinDefinition system URL. This url should be provided by Gemalto Project Manager

postDefinePinRequestECDH_CB – Output parameters		
Type	**Attribute**	**Detail**
JResponse	errorCode	Request process status. Error code table can be found at: §5.3 PIN Definition – Error Management
	errorMessage	Request process message.

Code sample:

```
//Send Request
DefinePinTokenRequest * definePinJResponse = [PinCodeRequest postDefinePinRequestECDH_CB: token
pinIndex: indexes signature: signature hostURL: hostURL];
```

5 Appendix

5.1 PIN Distribution - Error Management

Error Code	Error Description
0	Success
Error from Server	
100	Missing bankId parameter
101	Missing uniqueId parameter
102	Missing Authentication Code parameter
103	Missing Timestamp parameter
104	Missing Hash parameter
105	Missing status parameter
106	Missing RSA Public Key
107	Missing ECDH Public Key
108	Missing Hashmac parameter
109	Missing Transaction ID
110	Missing Signature
111	Missing ID Fournisseur
112	Missing Type
113	Missing Token
200	Invalid Timestamp
201	Bad Hash value
202	Invalid RSA Public Key
203	Wrong Authentication Code
204	Bad Authentication Code format
205	Bad Hashmac value
206	Invalid Signature
207	Invalid ECDH Public Key
208	Invalid Token JSON Format
209	Invalid Token Type
210	Given transaction ID has already been treated
300	Bank not found
301	Cardholder not found
302	Cardholder Status not satisfied for PIN delivery
303	PIN not available
304	Internal Key not found
305	Unexpected crypto error
306	Unknown Status code
307	Retry Limit Reached
308	Incompatible bankId and uniqueId
309	Incompatible Channel for Cardholder
312	Given parameter has bad value
500	Unexpected error
Error from Library	
90000	Null/Invalid input parameters
90001	No Network
90002	Null received data
90003	Received data has bad format
90004	Generating/recovering RSA Keys problem
90005	Data decryption or data decoding problem
90006	Request failed
90007	PIN bad length (4 to 6 digits) or bad format
90008	Generating/recovering ECDH Keys problem
90009	ECDH key doesn't belong to the elliptic curve – maybe an imposter
90010	Shared secret computation failed
90011	PIN parsing failed
90012	Failed to xor the AES key and get it back
90013	The received PIN seems to be incomplete
90014	Failed to convert ECDH key into public key
90015	Server's response not OK, request is not successful

Table 1 – PIN Distribution Error Codes

5.2 PIN Distribution Notification – Status Value

Value	Description
0	SUCCESS
1	SERVER_ERROR
2	APP_ERROR

Table 2 – PIN Distribution Notification Error Codes

5.3 PIN Definition – Error Management

Error Code	Error Description
Error from Server	
100	Missing bankId parameter
101	Invalid bankId (Bank not found in the system)
107	Missing ECDH public key parameter
110	Missing uniqueId parameter
111	Cardholder doesn't exist in the system
114	Cardholder Status not satisfied for PIN definition
115	Missing IdFournisseur in token
116	Missing IdTransaction in token
117	Missing IdType in token
118	Missing Token parameter
120	Missing TimeStamp
121	Invalid Timestamp
130	Missing PIN
131	PIN format is invalid
140	Missing HashMac
141	Invalid HashMac
200	Crypto error
201	AES key is not present in request
206	Signature is invalid
207	ECDH client public key is invalid
208	Token format is invalid
209	Field type in token is invalid (must be equals to 2 or 02)
210	Missing signature parameter
211	TransactionId in token has been already used during the last 5 minutes
212	Keypad size parameter is invalid
213	Missing keypad size parameter
214	Keypad coordinates are not in DB. Make sure getKeypad coordinates called has been made before calling define pin function.
300	Internal error
Error from Library	
90000	Null/Invalid input parameters
90001	No Network
90002	Null received data
90003	Received data has bad format
90004	Generating/recovering RSA Keys problem
90005	Data decryption or data decoding problem
90006	Request failed
90007	PIN bad length (4 to 6 digits) or bad format
90015	Server's response not OK, request is not successful

Table 3 - PIN Definition Error Codes

5.4 Token and Signature

The Token in JSON format must be signed using a 2048 bits RSA private key and SHA-256.

Thales's backend will verify the signature using the public key. Our PIN distribution system will verify the signature using the public key (certificate). In case of multiple certificates, the parameter SignatureCertAlias allows us to select the appropriate certificate (previously inserted inside our system) to verify the signature.

If this field is empty, the default certificate will be used (default certificate must be defined with the customer). Before being shared with Thales, this certificate must be signed by a trust CA.

5.4.1 Token Format

The token shall follow this format:

```
{"IdBEL":"IdBEL","IdPorteur":"IdPorteur","IdFournisseur":"IdFournisseur,"IdTran
saction":"IdTransaction","Timestamp":"Timestamp","Type":"Type","SignatureCertAl
ias":"SignatureCertAlias"}
```

Besides, the token MUST NOT be modified at all after it is signed. Otherwise, the signature verification may fail.

Example:

```
{"IdBEL":"BANK_ID","IdPorteur":"102456","IdFournisseur":"ServiceProvider,"IdTransaction":"456789
7","Timestamp":"1562320989","Type":"01","SignatureCertAlias":"CertificateDefault"}
```

It involves the creation of a RSA key pair on the authentication service provider/customer. The private key must be used to sign the JSON message, the public key must be sent to Thales in order to verify the signature.

5.4.2 Token and Signature CB compliant

The token must be formatted as a JSON message and needs to contain the following information:

- *IdBEL*: Customer unique Identifier.
- *IdFournisseur*: Unique identifier of the service provider used during authentication.
- *IdPorteur*: Unique ID (uid) of the cardholder retrieving his PIN. Same unique id used during provisioning step.
- *IdTransaction*: Unique identifier of the transaction. All request following will be rejected if they use the same idTransaction.
- *Timestamp*: Timestamp in UNIX format (unit second) generated for each transaction.
- *Type*: Transaction type: 00 for PIN delivery, 01 for PIN reminder and 02 for PIN definition
- *SignatureCertAlias*: alias of the certificate used to sign the token – Optional –

5.4.3 Token and Signature PCI-CPP compliant

For PCI-CPP (Visa/Mastercard) an authentication password must be added in the token. Thus token must contain the following information:

- *IdBEL*: Customer unique Identifier.
- *IdFournisseur*: Unique identifier of the service provider used during authentication.
- *IdPorteur*: Unique ID (uid) of the cardholder retrieving his PIN. Same unique id used during provisioning step.
- *IdTransaction*: Unique identifier of the transaction. All request following will be rejected if they use the same idTransaction.
- *Timestamp*: Timestamp in UNIX format (unit second) generated for each transaction.
- *Type*: Transaction type: 00 for PIN delivery, 01 for PIN reminder and 02 for PIN definition
- *SignatureCertAlias*: alias of the certificate used to sign the token – Optional
- *AuthenticationCode*: password of the cardholder retrieving his PIN. Same password used during provisioning step.

5.5 PIN Definition – PIN selection using Keypad

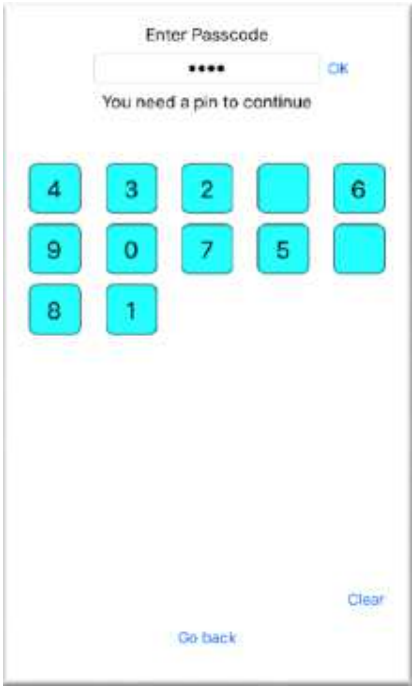
Example 1:

Call postGetCoordinateKeypad or getKeyPadCoordinates method with keypadSize = 12

Response: keypad ="[4, 3, 2, null, 6, 9, 0, 7, 5, null, 8, 1]";
Tap pin: 4343

Keypad	4	3	2	""	6	9	0	7	5	""	8	1
index	0	1	2	3	4	5	6	7	8	9	10	11

Get indexes of pin within keypad received: 0,1,0,1



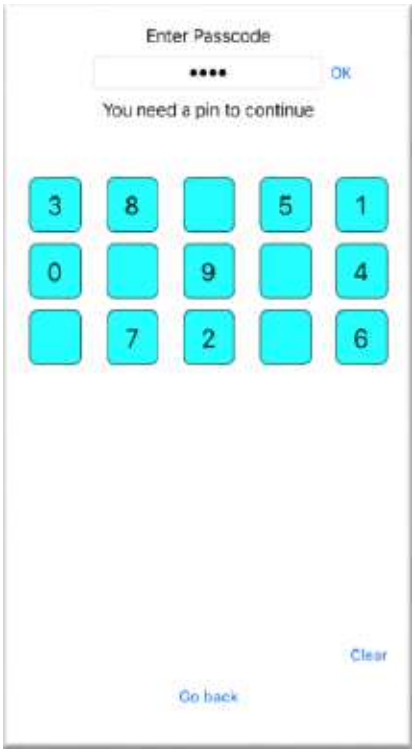
Example 2:

Call postGetCoordinateKeypad or getKeyPadCoordinates method with keypadSize = 15

Response: keypad = "[3, 8, null, 5, 1, 0, null, 9, null, 4, null, 7, 2, null, 6]";
Tap pin: 5959

Keypad	3	8	""	5	1	0	""	9	""	4	""	7	2	""	6
index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Get indexes of pin within keypad received: 3,7,3,7



5.6 HMAC Computation

The CryptoUtil class provides methods to perform cryptographic operations.

- `CryptoUtil.computeHMAC`

computeHMAC – Input parameters		
Name	**Type**	**Detail**
key	String	Key for HMAC256 computation
data	String	Data to compute

computeHMAC – Output parameters	
Type	**Detail**
String	Computed HMAC

- `[CardCompanionUtil calculHashMacValue]`

calculHashMacValue – Input parameters		
Name	**Type**	**Detail**
key	NSString	Key for HMAC256 computation
str	NSString	Data to compute

calculHashMacValue – Output parameters	
Type	**Detail**
NSString	Computed HMAC

5.7 Deprecated methods

Since Card Companion SDK version 1.7.0, some methods have been set to deprecated and replaced with new ones. These methods still remain in the library and are backward compatible.
However, please note that they may be removed in a further version, therefore it is recommended to update your app.

The following table indicates the deprecated methods and their replacement:

Deprecated Method	Replacement Method
`ProvisioningRequest.postDataForGetPin`	`PINDistribution.postGetPINCaptchaRequest`
`GetPinToken.getPinTokenCaptcha`	`PINDistribution.postGetPINCaptchaTokenRequest`
`NotifyRequest.postDataForNotifyRequest`	`PINDistribution.postNotifyRequest`
`TokenNotifyRequest.notifyPinToken`	`PINDistribution.postNotifyTokenRequest`
`DefinitionRequest.definePin`	`PINDefinition.postDefinePINRequest`
`DefinitionTokenRequest.definePinToken`	`PINDefinition.postDefinePINECDHRequest`
`DefinitionTokenRequest.getKeyPadCoordinates`	`PINDefinition.postGetKeyPadCoordinatesRequest`
`DefinitionTokenRequest.definePinTokenCB`	`PINDefinition.postDefinePINECDHCBRequest`

Table 4 - Deprecated Methods

- END OF DOCUMENT -


THALES
The people we all rely on
to make the world go round,
they rely on Thales