Ezio Mobile SDK V4.9

Overview

gemalto
security to be free

All information herein is either public information or is the property of and owned solely by Gemalto NV. and/or its subsidiaries who shall have and keep the sole right to file patent applications or any other kind of intellectual property protection in connection with such information.

Nothing herein shall be construed as implying or granting to you any rights, by license, grant or otherwise, under any intellectual and/or industrial property rights of or concerning any of Gemalto's information.

This document can be used for informational, non-commercial, internal and personal use only provided that:

- The copyright notice below, the confidentiality and proprietary legend and this full warning notice appear in all copies.
- This document shall not be posted on any network computer or broadcast in any media and no modification of any part of this document shall be made.

Use for any other purpose is expressly prohibited and may result in severe civil and criminal liabilities.

The information contained in this document is provided "AS IS" without any warranty of any kind. Unless otherwise expressly agreed in writing, Gemalto makes no warranty as to the value or accuracy of information contained herein.

The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, Gemalto reserves the right to make any change or improvement in the specifications data, information, and the like described herein, at any time.

Gemalto hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Gemalto be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Gemalto does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms' resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Gemalto be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Gemalto products. Gemalto disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service or loss of privacy.

May 10, 2019

Contents

Preface **4**

About This Guide 4
Who Should Read this Guide 4
For More Information 4
Contact Us 5

Getting Started **6**

About Ezio Mobile SDK 6
Supported Platforms 7
Package Contents 7
Documentation 7
Libraries 7
Examples 8
Tools 8

General Description **9**

System Overview 9
Ezio Mobile SDK Services 10
Core Services 11
Authentication Mode 13
OTP Services 14
Secure Storage 19
Out-of-band (OOB) 19
Msp Parser0 19
Secure Pin Pad 20

Terminology **21**

Abbreviations 21
Glossary of Terms 22

Preface

About This Guide

This document serves as an introduction to the Ezio Mobile SDK product. It includes information on the package contents, an overview of the Ezio Mobile system, and a description of the features and services provided by the SDK.

Who Should Read this Guide

The document is intended for application developers who are developing on and integrating applications with the Ezio Mobile SDK. This document uses the term "developer" and "application developer" to refer to the people implementing or customizing Ezio Mobile SDK. "User" refers to the end-user of the Ezio Mobile SDK solution on their mobile devices.

For More Information

The following table contains the complete list of documents for Ezio Mobile SDK V4.9.

Document	Description
Ezio Mobile SDK V4.9 Overview	This serves as an introduction to the Ezio Mobile SDK product. It includes information on the package contents, an overview of the system, and a description of the features and services provided by the SDK.
Ezio Mobile SDK V4.9 Release Notes	This contains detailed release information such as new features, issues fixed, known issues, devices and OS versions tested.
Ezio Mobile SDK V4.9 Migration Guides	This set of documents contains the necessary steps when migrating from an earlier versions to *Ezio Mobile SDK V4.9*.
Ezio Mobile SDK V4.9 Security Guidelines	This contains best and recommended security practices that an application developer should follow.
Ezio Mobile SDK V4.9 API Documentation	This document details the interfaces provided by the Ezio Mobile SDK libraries on Android and iOS platforms
Ezio Mobile SDK V4.9 Programmer's Guide	This guide details the usage of the Ezio Mobile SDK when extending the features, functionalities, and interfaces of Ezio Mobile solution.

You may also refer to the following list of links and documents:

- *Chip Authentication Program - Functional Architecture 2007*(MasterCard)
- Android PRNG Fix
- HOTP RFC
- TOTP RFC
- OCRA RFC

- *Gemalto Dynamic Signature (DS) Specification 1.8*
- *Gemalto Generic SWYS Specification 2.2*
- Android, Invalid Key Exception

Contact Us

For contractual customers, further help is provided in the Gemalto Self Support portal at http://support.gemalto.com or you can contact your Gemalto representative.
Gemalto makes every effort to prevent errors in its documentation. However, if you discover any errors or inaccuracies in this document, please inform your Gemalto representative.

Getting Started

About Ezio Mobile SDK

Ezio Mobile is an enterprise authentication solution for e-banking and e-commerce functions such as one-time password (OTP) management, challenge-responses, transaction data signing, data protection, and out-of-band (OOB) communication. It utilizes the user's mobile device as the security platform. The solution consists of mobile applications based on the Ezio Mobile SDK (software development kit).
For the detailed product overview, refer to *Ezio Mobile SDK V4.9 Overview*.

Supported Platforms

The Ezio Mobile SDK supports the following platforms and versions:

- Android
  Default Package
  Versions: 4.4 to 8.x.
  Architecture: armeabi-v7a, arm64-v8a, x86, x86_64
  FaceID Package
  Versions: 4.4 to 8.x
  Architectures: armeabi-v7a, arm64-v8a
- iOS
  Versions: 9.0 to 11.x.
  Architecture: ARMv7, ARM64

Package Contents

This section lists the main contents of the SDK package.

Documentation

Ezio Mobile SDK provides the following documentation:

- Overview (This document)
- Release Notes
  Contains detailed release information such as new features, issues fixed, known issues, tested devices and OS versions.
- Programmer's Guide
  Contains details on using Ezio Mobile SDK to extend the features, functionalities, and interfaces of the Ezio Mobile solution.
- Security Guidelines
  Contains the recommended security practices that developers must apply.
- API documentation
  Describes the interfaces provided by the Ezio Mobile SDK libraries on Android and iOS platforms.

Libraries

Ezio Mobile SDK provides the following libraries:

- `android/debug/libidpmobile.jar`
  A debug version of the Android Java library. It comes with the set of native libraries.
- `android/release/libidpmobile.jar`
  A release version of the Android Java library. It comes with the set of native libraries.
- `ios/Debug/EzioMobile.framework`
  A debug version of the Xcode framework suitable for iOS.
- `ios/Debug_Nocoverage/EzioMobile.framework`
  A debug version without code coverage of the Xcode framework suitable for iOS.
- `ios/Release/EzioMobile.framework`
  A release version of the Xcode framework suitable for iOS.

Note:

The application must not go live with the debug version of the library. The debug version is only for development purposes that may contain debug symbols and introduce certain security issues if used in production. The application must go live using a release version of the library and a properly signed CA Certificate for the server.

Examples

Ezio Mobile SDK provides the following examples to illustrate the usage of Ezio Mobile SDK. Refer to the platform's readme.txt for the details on building and deploying the project and the additional information.

- `android/example/eziomobilesdk_cap_example`
- `android/example/eziomobilesdk_oathtotp_example`
- `android/example/eziomobilesdk_securepinpad_example`
- `android/example/eziomobilesdk_oob_example`
- `android/example/eziomobilesdk_biofp_example`
- `android/example/eziomobilesdk_dskpp_example`
- `android/example/eziomobilesdk_dskpp_swift_example`
- `android/example/eziomobilesdk_facial_example`
- `ios/example/eziomobilesdk_cap_example`
- `ios/example/eziomobilesdk_oathtotp_example`
- `ios/example/eziomobilesdk_oathtotp_swift_example`
- `ios/example/eziomobilesdk_securepinpad_example`
- `ios/example/eziomobilesdk_oob_example`
- `ios/example/eziomobilesdk_oob_example_swift_example`
- `ios/example/eziomobilesdk_biofp_example`
- `ios/example/eziomobilesdk_dskpp_example`
- `ios/example/eziomobilesdk_faceid_example`

Note:

FaceID example projects are only available in Ezio Mobile SDK Full release package.

Tools

Ezio Mobile SDK provides the Token Builder tool (tools/eps-tokenbuilder) for application development. The Token Builder tool enables the application developer to provision their device without a live EPS server. It generates two types of files:

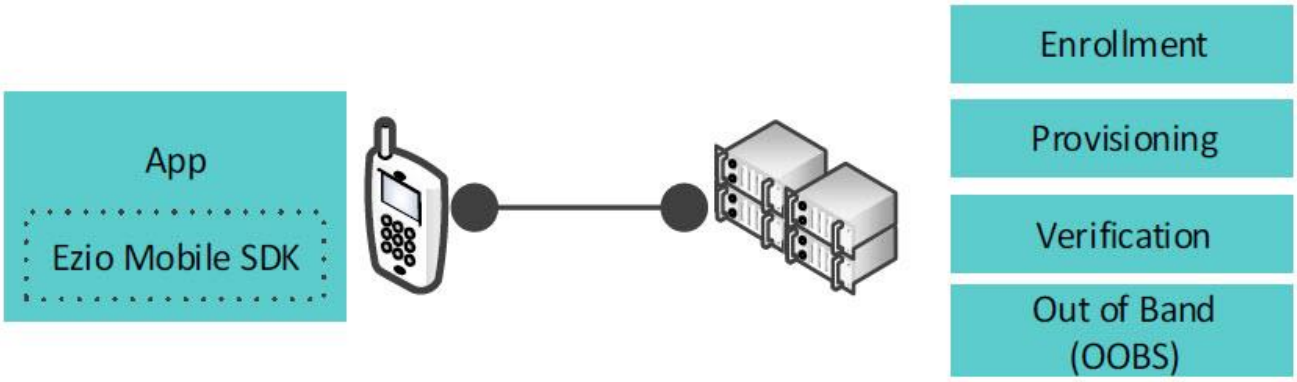
- a file having contents that the Ezio Mobile SDK requires.
- a file having contents that the verification server needs.

For further details, refer to the documentation within the tool.

General Description

This chapter provides an overview of the Ezio Mobile SDK from a technical perspective. Ezio Mobile SDK supports handsets running Android or iOS.
The Ezio Mobile SDK's location in the system is shown in Figure 1.

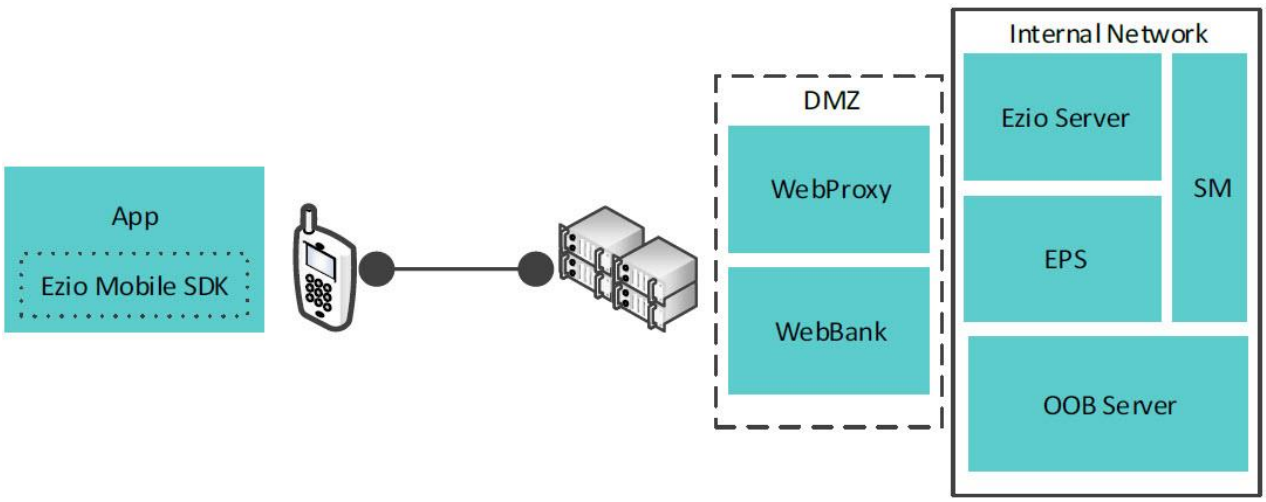
Figure 1 System Overview



System Overview

This section gives a description of the ecosystem of Ezio Mobile SDK and a brief description of each component.
The system components can be divided into groups, based on where they are deployed. There are three principal locations—the handset, front-end and back-end. The front-end components are typically located in a De-Militarized Zone (DMZ) and the back-end components are located on the internal network.

Figure 2 Detailed System Overview



The system comprises of the following components:

- Ezio Mobile SDK
  Provides functionality to create a soft token and provides services to the handset application to manage it

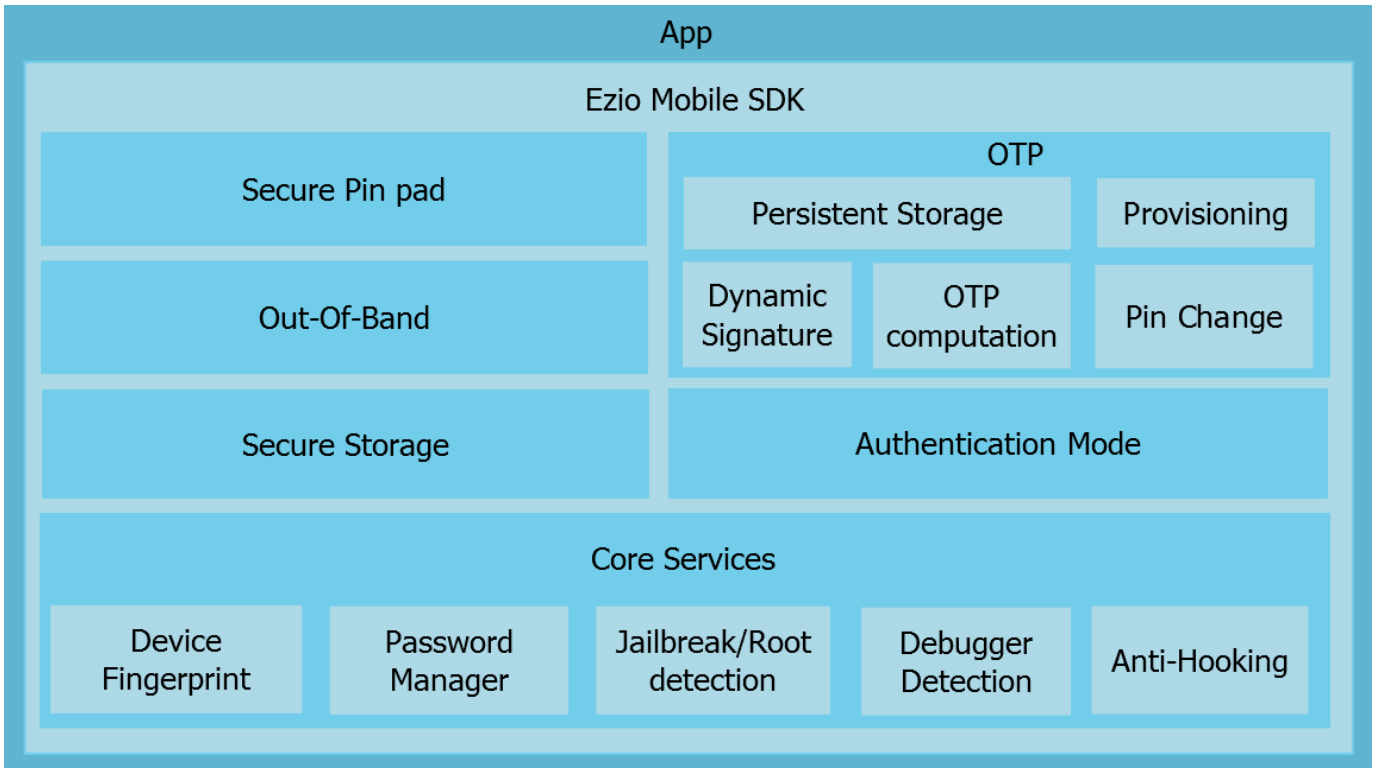
(provisioning, storage device sealing) for OTP computation. It also provides a storage to store sensitive data and Out-of-band communication.

- App - Mobile Application
  The application running on a handset enables the end user to generate one-time passwords, electronic signatures, store sensitive data, and making Out-of-band communications. For this purpose, it uses services provided by the Ezio Mobile SDK.
- WebBank
  The Web Bank is a component in the bank´s infrastructure. It plays a role in the enrollment process. A user visits the web bank to sign up for the mobile application or the mobile application service by entering information requested by the bank. This component is bank specific and will only be described further regarding specific use cases in the section *Provisioning of User Credentials*.
- WebProxy
  A generic HTTP(S) reverse-proxy service that receives handset provisioning requests and forwards these unmodified to the EPS. This is part of the bank's infrastructure and is not described further.
- Ezio Server
  The Ezio Server authenticates one-time passwords generated by the end users. All operations on sensitive data are performed inside an SM.
- EPS
  The Enrollment Provisioning Server (EPS) is the central component in the enrollment/provisioning process. It serves the front end services with credentials for provisioning. It also serves the Ezio Server with enrollment information and keeps track of each user's progress in the enrollment/provisioning process. All operations on sensitive data are performed inside an SM.
- SM
  The Security Module (SM) is a secure environment in which cryptography operations on sensitive data takes place. All backend components including the Ezio Server and EPS that handle sensitive data have an SM at their disposal. The SM guarantees that no keys or sensitive data are available in clear text on the backend network except for the SM. Customers that require high internal security should choose a Hardware Security Module (HSM). Other customers can choose a Software Security Module (SSM).
- OOB Server
  The Out-of-band (OOB) Server is a server component in the out of band service. It is used in conjunction with Ezio Mobile SDK to provide a secure messaging channel between the backend and the mobile client.

Ezio Mobile SDK Services

The following figure shows the services that Ezio Mobile SDK provides to the mobile application to manage credentials and generate OTPs.

Figure 3 Services Provided to the Application



Core Services

The core services provided are as follows:

- Password Manager
  Password Manager provides a mechanism to protect features of the Ezio Mobile SDK with the use of a password. Depending on the customer needs, a password can be set globally or a default password will be used and managed by the SDK. The features protected by Password Manager is OTP Computation (Android Only), Secure Storage, and OOB.
- Device Fingerprint
  Ezio Mobile SDK provides services to seal the user's credentials with information retrieved from the device. The fingerprint is used by OTP computation, Secure Storage, and OOB service. The application does not need to provide this information but can request to use it instead.
  - For Android:
    - Device information. The sealing process uses information linked to the device itself (not the operating system running on the device). If this is used and the user tries to restore the application and its data on another device, then all OTPs on the other device will be invalid. Since Ezio Mobile SDK V4.9.2, due to the Android Q Privacy Policy enforcement, this fingerprint source is based on an auto generated file. On the previous version, this device fingerprint source uses the IMEI/MEID/ESN information on Android. Since Ezio Mobile SDK V4.4, it can be disabled globally from the Core configuration via a flag `DeviceSourceBinding`.
    - Soft information. The sealing process uses information linked to the software part of the device. Updating the operating system has no impact on it. On Android, this device fingerprint source will use **Settings.Secure.ANDROID_ID**.
    - Service information. The sealing process will use information (if available) linked to the carrier account. If the SIM card is changed, then all OTPs will be invalid. Since Ezio Mobile SDK V4.9.2, this information is deprecated and is ignored due to Android Q

Privacy Policy enforcement. However, this information is still required in Data Migration Process. On the versions before V4.9.2, this information uses the IMSI information

- For iOS:
  - Device information. The sealing process uses information linked to the device itself. If a user tries to backup and restore the application and its data on another device, then all OTPs on the other device will be invalid. On iOS, this device fingerprint source uses identifierForVendorinformation provided by the platform. Since version 3.x, this is enforced by default and cannot be disabled. This value is stored into the platform's keychain, which not only prevents the data from being backed up and decrypted on another device via iTunes or iCloud, but also because of the Apple App Store bug, which occurred sometime in May 2015 – July 2015 wherein app update from app store generated different values of **identifierForVendor**.
  - Soft information. The sealing process will use information linked to the software part of the device. Updating the operating system has no impact on it. On iOS, this device fingerprint source will use a random value generated at first application run and stored into the keychain items. This information is backup-able.

In addition to those predefined types, the Ezio Mobile SDK also supports using custom fingerprint data. The application can provide a custom value of any length. It is good practice to include the application package name or bundle identifier at a minimum. This ensures domain separation of user data from another application using the Ezio Mobile SDK. Ezio Mobile SDK never stores the Custom Data provided by the application. If used, application needs to provide back this Custom Data each time it is required by the security functions.

For further details regarding fingerprints, their constraints and their usage, refer to the *Ezio Mobile SDK 4.1 Programmer's Guide*.

Note:

Ezio Mobile SDK never stores the fingerprint data nor does it store a hash of the fingerprint data. Ezio Mobile SDK temporarily uses this data to encrypt/decrypt the token credentials saved in Token Persistent Storage. It is also used to encrypt/decrypt data stored in Property Persistent Storage. On the other hand, on application choice, Ezio Mobile SDK can store a two-byte checksum of all the device fingerprint values. This enables the application developer to choose between two behaviors—detect the cloning of data or silently generate an invalid OTP.

- Root/Jailbreak Detection
  Ezio Mobile SDK provides services to detect if the application is running on a rooted or jailbroken device. The primary service is a jailbreak policy that changes the behavior of Ezio Mobile SDK when creating or retrieving tokens for OTP. The following policies may be set:
  - Ignore – Does nothing.
  - Fail (default) – Throws an exception or returns an error.
  - Remove – For OTP token, removes a particular token that is being retrieved and throws an error. For OOB, it silently unregisters the client, removes all associated data and throws an error.
  - Remove All – Removes all tokens then throw an exception or return an error.

  When using OOB features, the following policies may be set:
  - Ignore – Does nothing.
  - Fail (default) – Throws an exception or returns an error.

- Unregister – Unregisters the client. A request is sent to the server to unregister the client and related credentials on the device are wiped out.

Additionally, the application may use a secondary service that simply returns the jailbreak status of the device. The status may then be used by the bank to assess the risk of executing on such a device.

- Anti-Hooking
  This feature prevents an attacker to hook critical method calls made by Ezio Mobile SDK. This feature is only available on Android platform from Ezio Mobile SDK V4.0. From Ezio Mobile SDK V4.0 to V4.3, anti-hooking is enabled by default and cannot be disabled. When an attacker loads a hooking module on device and tries to hook into the SDK calls, SDK will directly throw a runtime exception in such scenario. Since Ezio Mobile SDK V4.4, the runtime exception is replaced by a normal exception, so application developer is able to catch the exception to avoid the application from crashing. In addition, the SDK has provided a callback functionality in which the application maker can choose to continue or to abort when hooking is detected. Without setting the listener, or if the application decides to abort the operation from the callback, Ezio Mobile SDK aborts and throws the exception.
- Debugger Detection
  Debugger detection in Ezio Mobile SDK automatically detects a debugger attached to the application. This feature is enabled by default in release mode and cannot be disabled.
- Anti-tampering
  This feature prevents an attacker from tampering native shared libraries, that is. the `.so` files of Ezio Mobile SDK. This feature is only available on Android platform from Ezio Mobile SDK V4.9. Anti-tampering is enabled by default and cannot be disabled. When an attacker tampers the native shared libraries, Ezio Mobile SDK will detect this and cause the application to crash or hang.
- Emulator Detection

  This feature is only available on Android since Ezio Mobile SDK V4.8. Ezio Mobile SDK will crash the app when it detects the application runs on emulator.
- Overlay Apps Detection
  This feature informs the application with a list of app names which may potentially do overlay attacks. This feature is only available on Android platform from Ezio Mobile SDK 4.9.

Authentication Mode

The authentication mode services provide multiple ways to authenticate users before accessing encrypted user credentials (for example, generating an OTP). Any activated authentication mode can authenticate a user. The service can only be used after it is enabled. Currently the supported authentication modes are:

- PIN (mandatory)
  This uses what the user knows (knowledge) to protect the credential. This mode is mandatory to ensure there is always one useable authentication mode. The security of the solution relies on the property that using an invalid PIN generates an incorrect OTP value that is indistinguishable from a good one. Attackers cannot validate a PIN without submitting the OTP to an authentication server, which limits the number of false attempts and can block the account if required.
- Biometric Fingerprint (optional)
  This uses what the user is (inherence) to protect the credential. It can only be activated when the app is running on supported hardware and when the user has configured his or her fingerprint. See the API documentation for further details and restrictions.
- FaceID Authentication (optional)
  This uses what the user is (inherence) to protect the credential. It can only be activated when the app is

running on supported hardware and when the user has enrolled his or her face data into the system. See the API documentation for further details and restrictions.

Note:

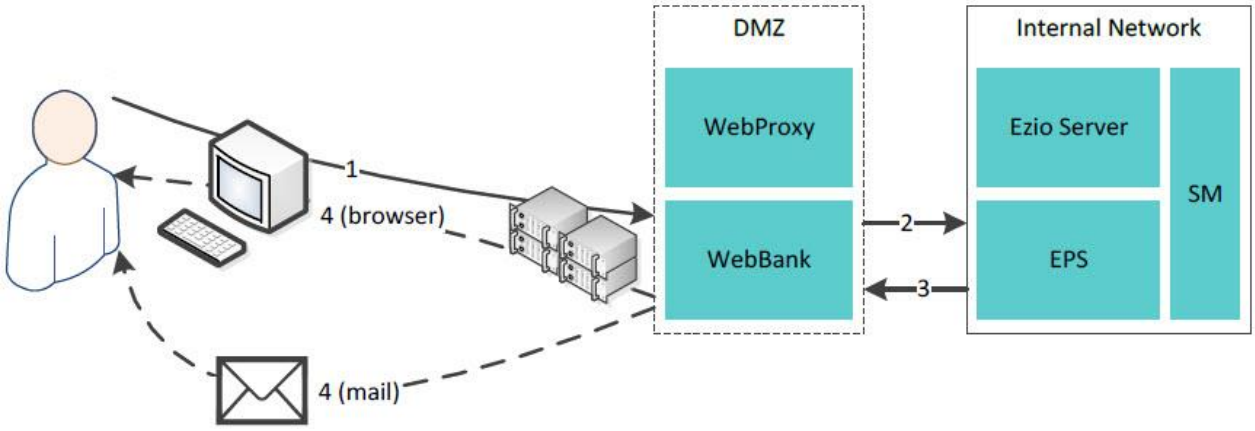
FaceID authentication mode is only available in Ezio Mobile SDK Full release package.

OTP Services

Ezio Mobile SDK provides the following OTP services:

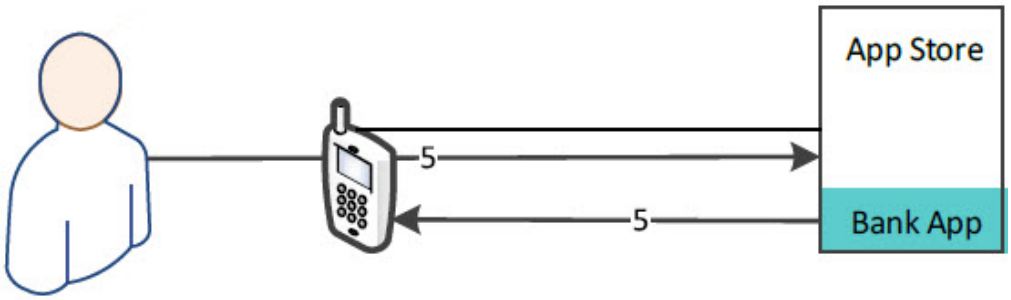
- Provisioning of User Credentials
  Provisioning is the process of securely importing user credentials from the EPS to the mobile device so that an application can provide new services. This process is configured by the application, but is executed entirely by Ezio Mobile SDK for simplicity and security of use. After a successful execution, the PIN wrapped token key and related data are stored transparently by Ezio Mobile SDK in device persistent memory. At this point, the application is capable of generating OTPs from the credentials. Provisioning can be executed several times in order to import one or more credentials for one or more users.
  The provisioning process can be configured to define the URL and protocol version of the provisioning service as well a variety of security configurations. The security of the provisioning process is composed of the transport layer security (TLS) protocol that wraps a proprietary protocol between the EPS and Ezio Mobile SDK. The former can be optionally reduced or ignored for testing and debugging purposes while the latter is always enforced. However, for production releases the application shall use the default security level.
  The Ezio Mobile SDK supports a debug mode for development purpose that allows user to do provisioning without a TLS tunnel, TLS with self-signed certificate, or a connection with HTTP hostname mismatch.
  In case there is no EPS server during development, the offline token builder tool that shipped along with SDK is available. This allows application developers to begin testing without having access to a working EPS. For further instructions regarding how to use this feature, refer to *EzioMobile SDK – Programmer's Guide*.
  OATH dual-seed token is supported in provisioning. The OTP computation could be done from one of two selectable keys in a single token. The dual-seed token unlocked more possibilities to user to choose the key for different purposes or scenarios. The usage of two keys could be pre-defined by the application and then selected from the token instance.
  Offline Provisioning is available to replace the real provisioning for development purpose when EPS is not available. The provisioning response and session key is generated by an offline provisioning tool that is delivered along with Ezio Mobile SDK binaries. The process after offline provisioning is similar with the actual provisioning.
  Besides offline provisioning, Ezio Mobile SDK also provides Plain Text Secret Import feature to import a secret that is usually obtained during provisioning. The secret itself is encrypted by using a PIN inside Ezio Mobile SDK.
  Ezio Mobile SDK also supports HTTP headers customization during provisioning. This customization becomes a part of EPS configuration and there is no limitation of the number of customized HTTP request headers to be added.
  Ezio Mobile SDK communicates directly with other components in the system to perform the provisioning. This step is triggered by the handset application that must provide the registration code as well as the EPS' address and public key for the communication and the fingerprint configuration and credential label for storage.
  The following set of diagrams describes the flow of the enrollment process:

Figure 4 Enrollment (Steps 1 to 4)



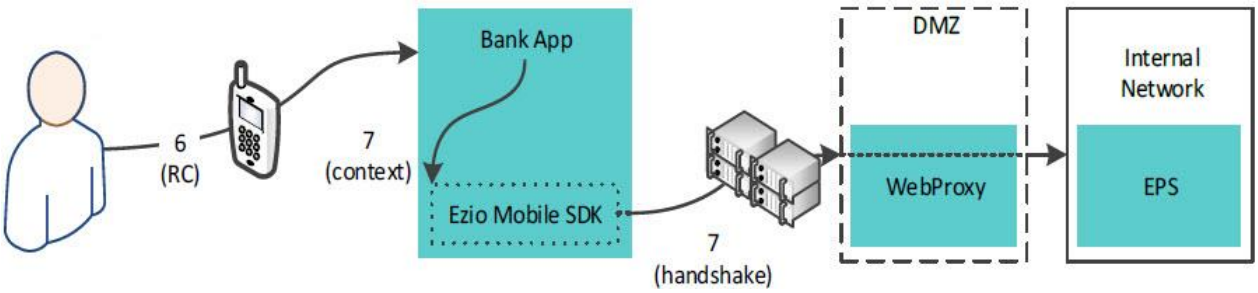
1. The end user logs into the web bank and registers for the mobile application.
2. The web bank service contacts the EPS.
3. The EPS allocates an entry and assigns it a RC. The RC is used by the handset application in the provisioning process to identify itself to the EPS. The PIN is used by the end user to generate OTPs. RC and PIN are returned to the web bank.
4. The web bank distributes the RC and PIN to the end user via different channels like the web browser, PIN letter, push notification or some other channels.

Figure 5 Bank App Download (Step 5)



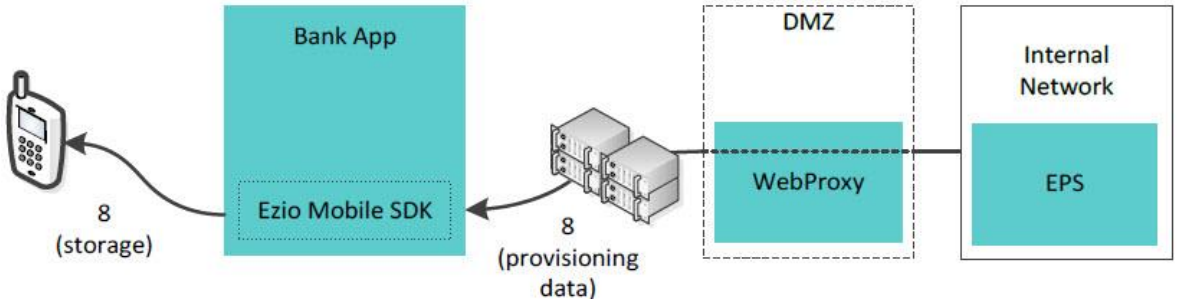
5. The end user downloads the mobile application. The application can be hosted by the bank, or on a public market like Apple's App Store or Google Play.

Figure 6 Start of Mobile Provisioning (Steps 6 - 7)



6. The end user starts the mobile application and enters the registration code (RC) received earlier.
7. The mobile application provides Ezio Mobile SDK with the provisioning context (registration code, EPS address and key, fingerprint configuration, credential label) and starts the operation. Ezio Mobile SDK will perform the provisioning by a handshake protocol with EPS.

Figure 7 Provisioning Completed (Step 8)



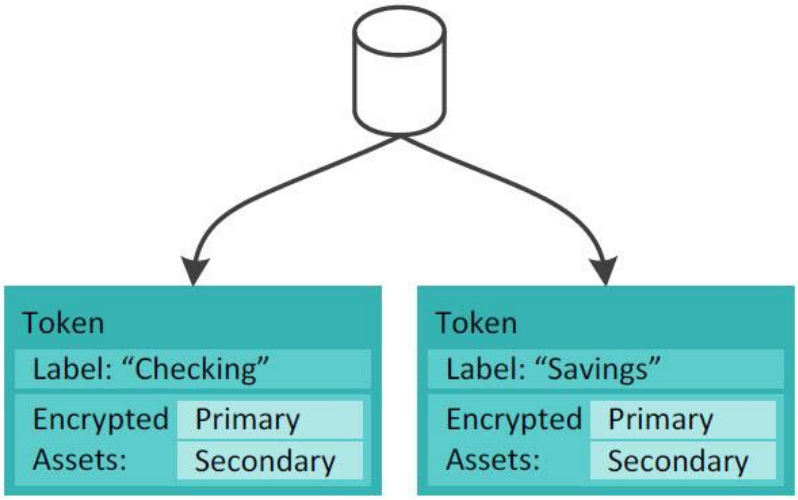
8. The EPS looks up the earlier allocated entry with the registration code. The EPS fetches the user's PIN encrypted key and some other secondary assets. Then the EPS packages the provisioning data and sends it to Ezio Mobile SDK for storage on the device.

- Token Persistent Storage
  Ezio Mobile SDK contains all the resources to manage user credentials stored in persistent memory. Each is identified by a unique label the application provided during provisioning. Each label is linked to a "token" inside Ezio Mobile SDK. At any time, the application can retrieve the list of available tokens, remove a token, or use a token to generate OTPs. A token's credentials can be decomposed into the following asset levels:

  – Primary Asset
    The token's secret key used as part of an OTP calculation. This is the only data that completely prevents an attacker from forging an attack against a user's credentials.

  – Secondary Asset
    All data involved in generating an OTP. This data does not allow an attacker to forge an attack against a user's credentials.

Figure 8 Token Persistent Storage



Tokens use the platform's resources for securing the data. Therefore, applications on different platforms may enforce different security schemes and risk managements to leverage the features of a particular platform. In the storage layer, Ezio Mobile SDK applies two schemes for security.
The first scheme uses a PIN to protect the user's primary asset. The application cannot influence this scheme, as it is the most important one. The security of the solution relies on the property that using a wrong PIN will generate an incorrect OTP value that is indistinguishable from a good one. Attackers cannot validate a PIN without submitting the OTP to the Ezio Server, which limits the number of false attempts and blocks the account if required.
The second scheme protects both the primary and secondary assets. Ezio Mobile SDK encrypts them

in order to ensure they are never in plaintext in persistent memory or outside of the device (such as back ups). Security of these values cannot depend on the user's PIN because this would introduce patterns that may be used to distinguish a good PIN from a bad one. The primary security of the solution remains on the first scheme's PIN-protected key.
A rising barrier of protection that increases the complexity of an attack is the design driver. The security level will be different depending on whether the application accepts impacts on the user experience or not. Indeed Android platform does not provide management of secure containers by the system (like keychain for example on iOS). Ezio Mobile SDK provides two possibilities the application can decide to use.
The first option, used by default, lets the Ezio Mobile SDK apply its built-in mechanisms to protect secondary assets. In this case, data will be encrypted by keys managed by the Ezio Mobile SDK. Since this version of Ezio Mobile SDK for Android is protected by a software only mechanism, it is not the preferred one.
The second option uses a password chosen by the user (something the user knows) that provided to Password Manager Service to protect the secondary assets. The password needs to be provided each time the application is started. Ezio Mobile SDK will use this value to cryptographically derive a key for protection. The password is never stored in persistent memory by Ezio Mobile SDK. This second option is the only reliable way to efficiently protect secondary assets with certain cost of usability reduction.
If the second option is chosen, special care must be taken regarding the password selection. As explained before, the protection of secondary assets introduces patterns that can be used to forge a brute force attack. Therefore the application must enforce the following rules:

i. A user's PIN used to protect the key (provided by the bank after enrollment) must not be used in place of the password. To do so, the application must implement a filter that prevents any possible re-use of the PIN. For example, if a PIN is 4 to 8 digits (defined by the server enrollment policy), then the application may reject any password that does not have at least one alphabetical character or symbol in order to actively prevent the user from reusing their PIN.
ii. Due to internal mechanisms to derive a protected key from the password, this value should at least have the following properties to offer resistance to any exhaustive search:
    - Minimum of 6 characters in space \[a-z, A-Z, 0-9, *\] is recommended.
    - Avoid 'weak' password based on a dictionary word or the user's name.

Meanwhile in iOS, all secondary assets are encrypted by Ezio Mobile SDK by using a randomly generated key that is securely stored in the iOS's keychain mechanism. This mechanism is the most advanced secure storage technology available on iOS. Therefore, this is the only option for protecting secondary assets on iOS.

- OTP Generation
  OTPs are set of digits, cryptographically linked to a challenge, data to sign, counter or time, and a secret key. They are generated by the handset with the goal of authenticating or signing a transaction. The mobile application does not require network access to generate OTPs. An OTP is checked on the server side to authenticate it.
  Ezio Mobile SDK supports different modes of OTP generation. The first four modes are described by the CAP specification, followed by three HMAC-based modes defined in HOTP, TOTP, OCRA memos. The two last modes are Gemalto proprietary, known as *Dynamic Signatures*. Ezio Mobile SDK also contains a toolbox that provides helpers to manipulate the OTP and challenge values: OTP formatting, OTP scrambling, OTP prefix with token sequence number, challenge check digit, challenge dynamic signature template identifier, and others.
  CAP-based OTP modes that are defined in MasterCard's Chip Authentication Program - Functional Architecture 2007:

- CAP/Mode1 provides an authentication mechanism. Input contains an unpredictable challenge, but may also include a transaction amount with or without the corresponding currency. This data is transformed into an application cryptogram that is used to generate the token.
- CAP/Mode2 provides a simple OTP mechanism. In contrast to the previous mode, no data input is used to generate the OTP.
- CAP/Mode2 with Transaction Data Signing (TDS) is an extension of Mode2. It provides a mechanism in which up to ten data fields can be signed together. The token value is cryptographically linked to these data fields which are freely formatted. For example, a bank account transfer authentication may use these data fields to store the origin account number, destination account number, amount, currency, date, etc.
- CAP/Mode3 can be used to implement challenge-response authentication. This mode uses an unpredictable challenge value only, and cryptographically signs it. This mode functions in exactly the same way as Mode1, where currency and amount values are not used. It has been included to allow issuers to specify amount and currency handling (Mode1) and still have the option to sign a challenge value for other services that do not use amount and currency (Mode3).

HMAC-Based OTP modes defined in HOTP, TOTP, OCRA memos:

- HOTP event-based OTP provides a HMAC-based mechanism to generate OTP values. Ezio Mobile SDK also support unofficial HOTP with SHA-256 digest algorithm.
- TOTP time-based OTP provides a similar mechanism as previous one in which the event counter moving factor is replaced by the time. It provides short-lived OTP values.
- OCRA OTP specify mechanisms that leverage the HMAC-based One-Time Password (HOTP) algorithm and offer one-way and mutual authentication, and electronic signature capabilities.
- Dynamic CVX2 is based on TOTP. The 3-4 digits OTP could be generated with a minor specification change. Dynamic CVX2 changes every 20 minutes. However it is made configurable for the application developer to set the expiry time of DCVX2.

- Gemalto proprietary Dynamic Signature modes defined in *Gemalto Dynamic Signature (DS) Specification 1.8:*
  - Dynamic Signatures can be described as an enhanced Challenge/Response scheme that is adaptable and controlled from the server side. The central parameters of Dynamic Signatures are the challenge, domain, dialogs, template, purpose, and intent. It implements Secure Domain Separation as an option.
  - Dynamic Signatures based on CAP/Mode2-TDS uses the same concept as the previous one but the underlying algorithm is CAP/Mode2-TDS. The advantage of using CAP/Mode2 TDS is that back end support for Dynamic Signatures can be added on any standard CAP compliant authentication server using the Ezio library.
- Dynamic Signatures
  Dynamic Signatures provide dynamic behavior that allows a bank to respond in real-time to threats. There are two modes of operation for Dynamic Signatures:
  - The unconnected mode uses a challenge code generated by the bank to trigger various predefined, but customizable templates from Dynamic Signature (DS) Design Specification in Ezio Mobile SDK. Each template contains dialogs that prompt the user to enter or accept certain information. Once the user has entered all the required information, a signature that includes all values and their dialog identifier (tags) can be computed.

  - The connected mode allows the bank to build arbitrary data to sign. The exact behavior of this mode must be defined by the bank. Further information for this mode is found in *Gemalto Generic SWYS Specification 2.2*.

- PIN Change
  Users can change their PIN at any time after the provisioning process has successfully completed. A single operation is provided by Ezio Mobile SDK to transform the old credentials into new ones that are protected by the new PIN. This operation checks if the new PIN complies with a configurable list of PIN rules, such as a minimum PIN length requirement. However, the complete PIN Change service is a multistep process requiring interactions between the user, the application, the Ezio Mobile SDK, and the Ezio Server. For information on changing PINs, refer to *Ezio Mobile SDK – Programmer's Guide*.

Secure Storage

Secure Storage provides a functionality for the user to store data in the form of key-value pair. It provides protections on the stored property that sealed by a password which is managed by the Password Manager as well as configured fingerprint data.

Out-of-band (OOB)

Out-of-band is a functionality of Ezio Mobile SDK that provides an out-of-band communication channel. The communication between client and OOB Server have extra layer of protection, on top of SSL communication. The OOB solution is added to Ezio Mobile SDK since V3.0 for both identity authentication and transaction verification. In the actual use case, the OOB solution is designed to work with push notification mechanism on mobile platform to provide a highly secure communication channel for notification and message transmission between backend and user's mobile devices.
There are three main categories of APIs that the OOB solution provides to Ezio Mobile SDK:

- Registration/Unregistration
  To register/unregister the client application to the OOB server.
- Notification Profiles Management
  To manage the notification profiles when the back end notifies the end user of a pending message.
- Messaging
  To enable users to fetch a message, acknowledge or reply to a message, send a message, verify a transaction request and sign a transaction request.

Starting from Ezio Mobile SDK V4.0, the application developer is also able to extend the message MIME type and implements customized type of messages and message handling. Custom MIME types can be registered with Ezio Moblile SDK at runtime to be treated during incoming/outgoing message parsing.

Msp Parser0

Mobile Signing Protocol (MSP) is a protocol used to transfer data for generating OTP and signing data. With the help of MSP parser, the application developers will be able to parse the msp frame to generate the OTP. The main features are:

- It supports OATH (HOTP, TOTP, OCRA) and CAP (Mode1, Mode2, Mode2TDS, Mode3) algorithms
- The application developers can configure MSP with the obfuscation and/or signature keys or without the keys.
- The frame data is transformed to OATH or CAP data and the application developers can easily access the data fields for OTP generation.
- In case of partial input, the application developers can easily identify the incompleteness of the field and update the field before using it for OTP generation.

Starting from Ezio Mobile SDK V4.6, the application developers will be able to use the MSP parser with OOB, to parse the frame and to generate the OTP.

Secure Pin Pad

Starting from Ezio Mobile SDK V2.5, the application developer can invoke Secure Pin Pad when a PIN is required from the user. Secure Pin Pad provides a way to manage and manipulate the PIN input process in a secured, controlled, and verified manner. It provides protection against malicious keyloggers, overshoulder attacks, memory dumping, and screen capturing attacks.
From V4.4, More customization APIs were exposed to the developer to improve the look and feel of Secure Pin Pad. Developers are able to customize the top screen and implement the customized ok button behavior.
For information regarding the usage of Secure Pin Pad, refer to *Ezio Mobile SDK - Programmer's Guide*.

Terminology

This section contains abbreviations and terms found in this document or related documents.

Abbreviations

EPS	Enrollment Provisioning Server
ESN	Electronic Serial Number
HOTP	HMAC based One Time Password
HSM	Hardware Security Module
IMEI	International Mobile Equipment Identifier
IMSI	International Mobile Subscriber Identifier
OATH	Open Authentication
OOB	Out Of Band
OCRA	OATH Challenge-Response Algorithm
OTP	One Time Password
PM	Password Manager
PIN	Personal Identification Number
PTC	PIN Try Counter
RC	Registration Code
SM	Security Module (see also HSM and SSM)
SMS	Short Message Service
SSM	Software Security Module
TLS	Transport Layer Security
TOTP	Time-based One Time Password
URL	Universal Resource Locator
VIC	Verify Issuer Code
VICATC	VIC Application Transaction Counter
VICTC	VIC Try Counter
DSKPP	Dynamic Symmetric Key Provisioning Protocol

Glossary of Terms

Provisioning Access Domain is a SIM Toolkit/UICC Toolkit parameter that specifies the identities or access rights (CHV & ADM) granted to an application to access GSM/UICC files and perform actions on these files. For example, if the Access Domain value is "FF" (No Access to the File System), attempts to access a file cause an exception.
Token The Ezio Mobile SDK's representation of a user's credentials.
User The mobile device user.