公的個人認証サービス

利用者クライアントソフト API 仕様書

【カード AP ライブラリ PKCS#11 編】

第 4.6 版

地方公共団体情報システム機構

変更履歴

版数	変更日付	変更内容
1.0 版	平成 16 年 1 月 16 日	新規作成
1.1 版	平成 16 年 10 月 14 日	Windows XP SP2 対応に伴い表 3-1 のプラットフォームを追加
2.0 版	平成 18 年 5 月 2 日	公的個人認証サービス利用者クライアントソフト Ver2.0 のリリースに伴い、動作環境、ソフトウェア構成図を変更
2.1 版	平成 18 年 11 月 1 日	MacOS 対応に伴い、第２章　ドキュメント体系、第３章　動作環境、第４章　第１節　ソフトウェア構成図、第６章　第１節　ライブラリのロード方法　を変更
2.2 版	平成 19 年 4 月 10 日	表 3-1 動作環境　を変更
2.3 版	平成 19 年 10 月 4 日	表 3-1 動作環境 ・ プラットフォームに WindowsVista、MacOS X 10.4.9、MacOS X 10.4.10 を追加
2.4 版	平成 20 年 10 月 10 日	表 3-1 動作環境 ・ プラットフォームに WindowsVista ServicePack1、WindowsXP ServicePack3、MacOS X 10.5.4、MacOS X 10.5.3、MacOS X 10.5.2、MacOS X 10.5.1、MacOS X 10.5、MacOS X 10.4.11 を追加 ・ Web ブラウザに Internet Explorer6.0 ServicePack3 を追加

版数	変更日付	変更内容
2.5 版	平成 23 年 04 月 01 日	図２－１ ドキュメント体系図**エラー! 参照元が見つかりません。**に「JavaDoc JPKICryptJNI(64bit)」を追加 表 ３-１ 動作環境 ・ 表 ３-１ 動作環境(Windows)、表 ３-２ 動作環境(MacOS)、表 ３-３ 動作環境(IC カード) に分割し、マトリクス形式の記述に変更。 表 ３-１ 動作環境(Windows) ・ OS に Windows 7(32/64 bit) , WindowsVista ServicePack2 を追加。 ・ Web ブラウザに Internet Explorer8.0 を追加。 表 ３-２ 動作環境(MacOS) ・ OS に MacOS X 10.6.4, MacOS X 10.5.6, MacOS X 10.5.5 を追加。 ・ Web ブラウザに Safari 3.2, Safari 5.0 を追加。 図 ４-１ ソフトウェア構成図 (Windows 対応版) を変更 第６章 第１節 ライブラリのロード方法に64bit版に関する記述を追加
2.6 版	平成 25 年 12 月 01 日	・第３章 動作環境 表 ３-１ 動作環境(Windows) Windows2000 を削除、Windows8(32/64bit)、Windows8.1(32/64bit)を追加 表 ３-２ 動作環境(MacOS) MacOS X 10.4.X, 10.5.X, 10.6.X を削除、MacOS X 10.7.5, OS X 10.8.4 を追加 ・第５章 第１節 sha-256 に対応する API の記載を追加 ・第５章 第３節 （８）C_GetMechanismList の備考にアルゴリズム CKM_SHA256 を追加。 ・第５章 第３節 （９）C_GetMechanismInfo の備考に type = CKM_SHA256 の場合を追加。 ・第５章 第３節 （２２）C_DigestInit の引数 CK_MECHANISM_PTR に アルゴリズム CKM_SHA256 を追加。 ・第５章 第６節 （６）署名検証処理の C_DigestInit に指定するアルゴリズム CKM_SHA256 を追加。 ・第６章 第１節 ロード情報定義ファイル例のパス ¥JPKI を削除。

版数	変更日付	変更内容
3.0 版	平成 26 年 04 月 01 日	全体　「地方公共団体情報システム機構」への事業承継により、組織名称を変更 全体　「公的個人認証サービス共通基盤事業運用会議」への事業承継により、「公的個人認証サービス都道府県協議会」の組織名称を変更する。
3.1 版	平成 26 年 07 月 01 日	・第３章　表 ３-１ 動作環境(Windows)で Windows XP を削除、Windows7(32/64bit)の Web ブラウザを IE10.0 から IE11.0 に変更、Windows 8(32/64bit)を削除、Windows 8.1 を Windows 8.1 update に変更 ・第３章　表 ３-１ 動作環境(Windows)で Web ブラウザの Safari6.0 を 6.1 に変更、OS X 10.8.4 を OS X 10.8.5 に変更し、Web ブラウザの Safari6.0 を Safari6.1 に変更、OS X 10.9.3(64bit)、Web ブラウザに Safari7.0 を追加 ・第５章　第１節　表 ５-１ サポート API 一覧の注釈に No21 を追加し、SHA256 による署名生成および署名検証について追記 ・第５章　第６節　（４）のタイトルに「((署名対象データを渡すパターン))」を追加 ・第５章　第６節　（４）C_DigestInit の pMechanism に SHA256 についての記載を追加 ・第５章　第６節　（５）署名生成処理(ハッシュ値を渡すパターン)のシーケンスを追加 ・第５章　第６節　（６）のタイトルに「(検証対象データを渡すパターン)」を追加 ・第５章　第６節　（７）に署名検証処理(ハッシュ値を渡すパターン)のシーケンスを追加 ・第５章　第６節　（８）のタイトルに「(署名対象データを渡すパターン)」を追加 ・第５章　第６節　（９）繰り返し署名生成処理(ハッシュ値を渡すパターン)の説明を追加 ・第５章　第６節　（１０）のタイトルに「(検証対象データをわたすパターン)」を追加 ・第５章　第６節　（１１）に繰り返し署名検証処理(ハッシュ値を渡すパターン)を追加

版数	変更日付	変更内容
4.0 版	平成 27 年 6 月 30 日	番号制度対応に伴い、以下を修正。 ・第１章 第１節に用語の定義を追加。 ・第２章のドキュメント体系を修正。 ・第３章の動作環境を修正。 ・第４章の機能仕様を修正。 ・第５章の API 仕様を修正。 ・第４章 第１節のソフトウェア構成図(MacOS 対応版)を修正。
4.0.1 版	平成 28 年 10 月 26 日	・第３章システム概要 動作環境　更新プログラムに係る注釈※3、4 の追加 ・文末　注意事項の利用用途の文言修正 その他、図の整形及び誤記等の文言修正
4.1 版	平成 28 年 11 月 30 日	PC 接続機能追加対応に伴い、以下を修正。 ・第１章　第１節の「用語の定義」に以下を追加。 ➢ PC/SC ➢ IC カードリーダライタ ➢ 挿入 ➢ NFC ➢ Bluetooth ・第２章　ドキュメント体系図に Android 版を追加 ・第３章　表 3-1 および表 3-2 に PC 接続機能対応可否追加。 ・第３章　表 3-1 に Windows 10(32/64bit)を追加。 ・第３章　表 3-3 動作環境(共通)を表 3-3 動作環境(IC カード)、第 1 節 PC/SC 対応 IC カードリーダライタ、第 2 節 Android 端末に分割。 ・第４章 第１節 ソフトウェア構成図に Bluetooth 通信を追加。

版数	変更日付	変更内容
4.2 版	平成 29 年 07 月 31 日	Java9 対応に伴い、以下を修正。 ・第２章 ドキュメント体系図を改訂。 ・第３章 表 ３-１の OS から Windows Vista(32bit)を削除。 ・第３章 表 ３-１の OS から Windows 8(32bit)を削除。 ・第３章 表 ３-１の OS から Windows 8(64bit)を削除。 ・第３章 表 ３-１の JavaVM に JRE9 を追加。 ・第３章 表 ３-２の OS から OS X 10.8, 10.9 を削除。 ・第３章 表 ３-２の OS に OS X 10.11, macOS v10.12 を追加。 ・第３章 表 ３-２の JavaVM に JRE9 を追加。 ・※２に説明文を追記。 ・第３章 表 ３-５の【PC 接続の場合】に Android 6.0.1、7.0 を追加。 ・第３章 表 ３-５の【Android 単体で利用する場合】に Android 6.0.1、7.0 を追加。
4.3 版	平成 31 年 03 月 31 日	・第３章 表 ３-２の OS から OS X 10.11 を削除。 ・第３章 表 ３-２の OS に macOS v10.13 を追加。 ・第３章 表 ３-５の【PC 接続の場合】、【Android 単体で利用する場合】に Android 8.0 を追加。

版数	変更日付	変更内容
4.4 版	令和 2 年 3 月 31 日	MacOS 版における開発言語(Java)変更対応に伴い、以下を修正。 ・第３章　表３-２を修正。 macOS v10.12、macOS v10.13 を削除。 macOS v10.14、macOS v10.15（Web ブラウザは Safari 13）を追加。 ・第４章 第１節 図４-２ ソフトウェア構成図(MacOS 対応版)から個人認証サービス AP C 言語 I/F から個人認証サービス AP JavaI/F の呼び出しを削除。 MacOS 版における住基カードサポート廃止に伴い、以下を修正。 ・第２章 ドキュメント体系図から「API 仕様書　Mac OS X　C 言語インターフェース編」を削除。 ・第３章 表３-２ ※２の注釈を削除。 ・第３章 表３-３ MacOS 版を使用する場合は個人番号カードのみ対応である旨を追加。 ・第４章 第１節 図４-２ ソフトウェア構成図(MacOS 対応版)から Keychain Services 及び CSSM を削除、。 ・第４章 第２節 表４－２から住基カードの記載を削除。 ・第６章 第１節の「path」の指定内容に住基カードは Windows 版のみ使用可能である旨の記述を追加 ・第６章 第１節の MacOS 版の場合のロード情報定義ファイルの例から「path」を削除。 MacOS 版における SHA-1 サポート廃止に伴い、以下を修正。 ・第５章 第２節 （８）C_GetMechanismList の備考の CKM_SHA_1 の記載に Windows 版のみ取得可能である旨の注釈を追加。 ・第５章 第２節 （９）C_GetMechanismInfo の備考に CKM_SHA_1 は Windows 版のみ設定可能である旨の注釈を追加。 ・第５章 第２節 （２２）C_DigestInit の CK_MECHANISM_PTR の指定内容の CKM_SHA_1 に Windows 版のみ指定可能である旨の注釈を追加。

版数	変更日付	変更内容
4.4版	令和2年3月31日	ブラウザ対応版、iOS版のリリースに伴い、以下を修正。 ・第２章　ドキュメント体系　「利用者クライアントソフト 機能概要説明書 ブラウザ対応編」を追加。 ・第２章　ドキュメント体系　「API仕様書 カードAPライブラリ ブラウザインターフェース編」を追加。 ・第２章　ドキュメント体系　「利用者クライアントソフト 機能概要説明書(iOS版)」を追加。 ・第２章　ドキュメント体系　「API 仕様書 iOS Framework編」を追加。 ・第３章　第２節　動作環境にAndroid9.0、10.0を追加。 Windows 7サポート終了に伴い、以下を修正。 ・第３章　表３-１を修正。 Windows 7(32bit)、Windows 7(64bit)および※3を削除。
4.5版	令和3年3月31日	「日本工業規格 X560-1」を「ISO/IEC 8825-1」に名称変更 ・第１章 用語の定義 ブラウザ対応版(Android)のリリースに伴い、以下を修正 ・第２章　ドキュメント体系 ・**エラー！参照元が見つかりません。**　動作環境 パスワードロックまでの残回数取得関数の追加に伴い、以下を修正 ・第５章第２節、第５章第４節を追加 ・第５章第６節を修正、追記 ・第６章第２節を追加
4.5版	令和3年3月31日	PKCS#11 Ver2.2 仕様書の参照先リンク切れに伴い、以下を修正 ・第５章第５節　構造体仕様
4.5.1版	令和3年3月31日	InstallShield2022対応に伴い、第３章　動作環境の記載を修正
4.6版	令和6年4月15日	MacOS版Ver3.7のリリースに伴い、第３章　動作環境の記載を修正

－ 目次 －

第１章　はじめに

公的個人認証サービス 利用者クライアントソフト(以下、JPKI 利用者ソフト)におけるカード AP ライブラリは、以下の機能を実現するための Application Program Interface(以下、API)を提供する。

- 証明書取得機能
- 電子署名生成機能
- 電子署名検証機能

以降、本書ではカード AP ライブラリのうち、PKCS#11 の API 仕様について説明する。

第１節　用語の定義

表 1-1 用語の定義

項番	用語・略号	説明
1	IC カード	以下のカードを指す総称。 ・住基カード ・個人番号カード
2	電子証明書	公開鍵及び発行対象を識別する情報を含むデータに、認証局が発行対象の正当性を保証する電子署名を付与して、発行されるデータをいう。データは、ISO/IEC 8825-1 の識別符号化規則により符号化された形式で利用される。
3	証明書	電子証明書と同義。
4	利用者証明書	公的個人認証サービスで発行した利用者の証明書。 本書では以下の電子証明書を指す。 ・住基カードに格納された署名用電子証明書 ・個人番号カードに格納された署名用電子証明書 ・個人番号カードに格納された利用者証明用電子証明書
5	利用者秘密鍵	公開鍵暗号方式において用いられる鍵ペアの一方。公開鍵に対する、利用者のみが保有する鍵。 本書では以下の秘密鍵を指す。 ・住基カードに格納された署名用利用者秘密鍵 ・個人番号カードに格納された署名用利用者秘密鍵 ・個人番号カードに格納された利用者証明用利用者秘密鍵
6	認証局の自己署名証明書	自認証局の公開鍵に対して、自認証局の秘密鍵で署名した証明書。 本書では以下の電子証明書を指す。 ・住基カードに格納された都道府県知事の自己署名証明書 ・個人番号カードに格納された署名用認証局の自己署名証明書 ・個人番号カードに格納された利用者証明用認証局の自己署名証明書
7	PC/SC	Personal Computer/Smart Card の略。
8	IC カードリーダライタ	以下の機器を指す総称 ・PC/SC 対応 IC カードリーダライタ ・Android 端末
9	挿入	IC カードリーダライタが IC カードを読み込める状態にすること。 具体的には以下の状態にすることを指す。 ・PC/SC 対応 IC カードリーダライタに IC カードをセットすること ・Android 端末に IC カードをセットすること
10	NFC	Near Field Communication　(近距離無線通信)の略。

項番	用語・略号	説明
11	Bluetooth	機器間の近距離無線通信 IEEE 802.15.1 の規格名称。
12	拡張 API	RSA Laboratories が標準提供している PKCS#11 ライブラリの API ではなく、機能拡充を目的として公的個人認証サービスが独自に追加した API のこと。

第2章　ドキュメント体系

JPKI 利用者ソフトのドキュメント体系図を以下に示す。本書は以下の体系図の網掛け部分に該当する。

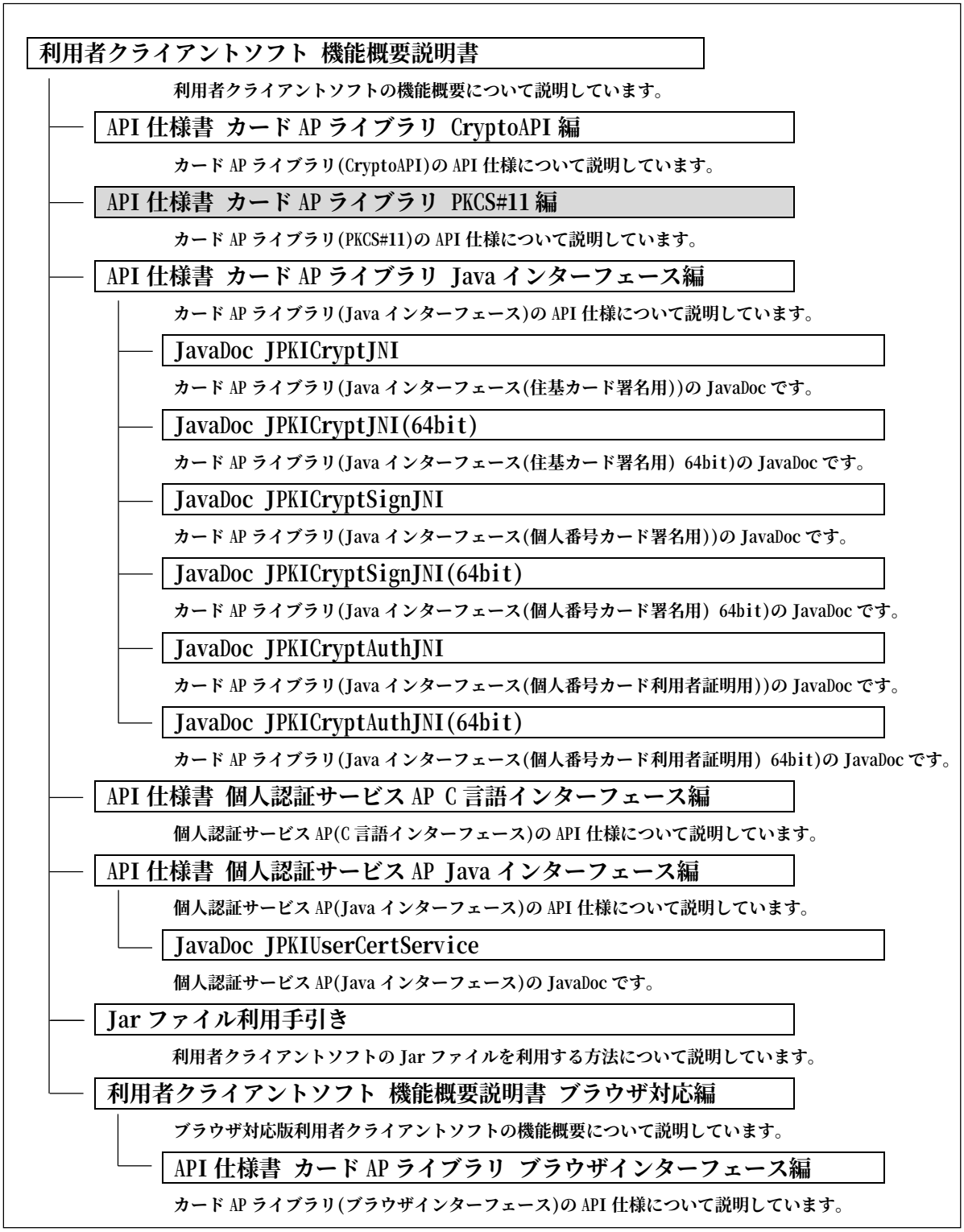


図 2-1 ドキュメント体系図(PC 版)

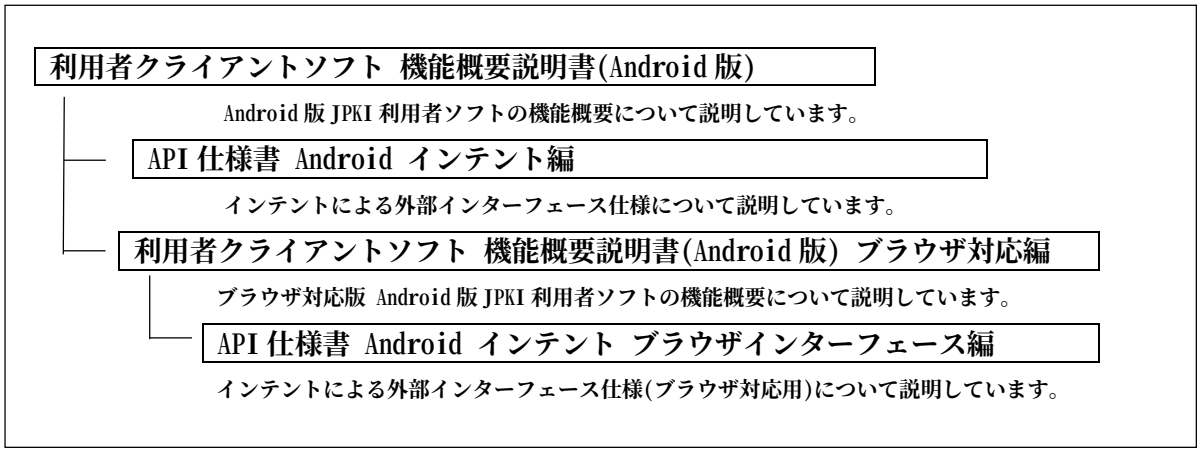


図 2-2 ドキュメント体系図(Android 版)

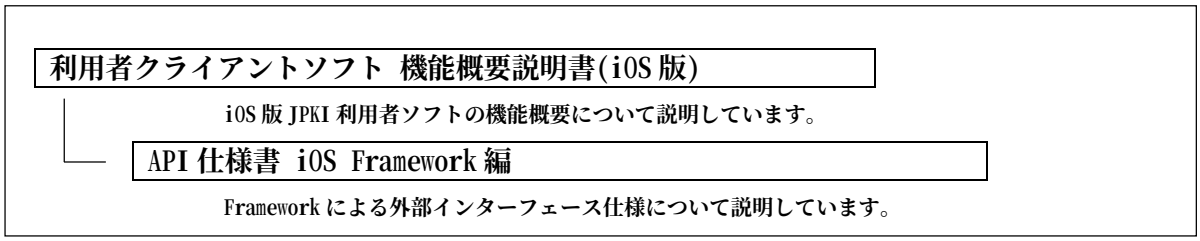


図 2-3 ドキュメント体系図(iOS 版)

第３章　動作環境

カード AP ライブラリ(PKCS#11)の動作環境は以下の通りとする。

表 3-1 動作環境(Windows)

OS(※1)	Web ブラウザ (※1, ※2)	JavaVM(※1) JRE8.0	PC 接続機能対応可否(※3)
Windows 10 Home/Pro (32bit/64bit)	Edge	○	○
Windows 11 Home/Pro	Edge	○	○

※1 本仕様書で定めるバージョンの開発時点の環境。最新の動作環境の情報は、JPKI ポータルサイトに掲載するものとする。
また、最新の動作環境に掲載された OS、Web ブラウザ、JavaVM 以外を使用した場合の不具合等に関するお問い合わせは、サポート対象外とする。

※2 プラットフォームが Windows の場合、暗号機能等の利用のために Microsoft Edge(カード AP ライブラリ（Java インタフェース編）の場合、IE モード)が必要。

※3 PC 接続機能については「利用者クライアントソフト 機能概要説明書 第３章 第３節 PC 接続機能について」を参照。

表 3-2 動作環境(MacOS)

OS(※1)	Web ブラウザ (※1)	JavaVM(※1) JRE8.0	PC 接続機能対応可否 (※2)
macOS 13 Ventura	Safari 16	○	×
macOS 14 Sonoma	Safari 17	○	×

※1 本仕様書で定めるバージョンの開発時点の環境。最新の動作環境の情報は、JPKI ポータルサイトに掲載するものとする。
また、最新の動作環境に掲載された OS、Web ブラウザ、JavaVM 以外を使用した場合の不具合等に関するお問い合わせは、サポート対象外とする。

※2 PC 接続機能については「利用者クライアントソフト 機能概要説明書 第３章 第３節 PC 接続機能について」を参照。

IC カードの動作環境は以下の通りとする。

表 3-3 動作環境(IC カード)

項目	条件
IC カード	住基カードまたは個人番号カードであること。 ※PC 接続機能を使用する場合は個人番号カードのみ対応。 ※MacOS 版を使用する場合は個人番号カードのみ対応。

第１節　PC/SC 対応 IC カードリーダライタ

PC/SC 対応 IC カードリーダライタの動作環境は以下の通りとする。

表 3-4 動作環境(PC/SC 対応 IC カードリーダライタ)

項目	条件
PC/SC 対応 IC カードリーダライタ	以下の条件を満たす PC/SC 対応 IC カードリーダライタとする。(「個人番号カード対応適合性検証済み IC カードリーダーライタ一覧」「住基カード対応適合性検証済み IC カードリーダーライタ一覧」(※1)を参照のこと。) ・ IC カードのインターフェース(非接触型、接触非接触両対応型)に対応していること。 ・ PC/SC 対応 IC カードリーダライタであること。 ・ USB など、パソコンに接続するためのインターフェースを有すること。 ・ PC/SC 対応 IC カードリーダライタと通信するためのドライバソフトウェアが提供されていること。 ・ IC カードの搬送方式が手動挿入/手動排出タイプまたは自動挿入/自動排出タイプであること。 ・ IC カードを挿入するスロットの数は 1 つとし、1 度に挿入できる IC カードは 1 枚であること。

※1 最新の「個人番号カード対応適合性検証済み IC カードリーダーライタ一覧」「住基カード対応適合性検証済み IC カードリーダーライタ一覧」の情報は、JPKI ポータルサイトに掲載するものとする。

第２節　Android 端末

Android 端末の動作環境は以下の通りとする。

表 3-5 動作環境(Android 端末)

項目	条件
Android 端末	以下の条件を満たす Android 端末とする。(「個人番号カード対応適合性検証済み Android 端末一覧」(※)を参照のこと。) ・Android 8.0、9.0、10.0、11.0 または 12.0 を搭載していること。 ・Bluetooth 4.2 を搭載していること。 ・ISO/IEC 14443 Type B に対応している NFC を搭載していること。

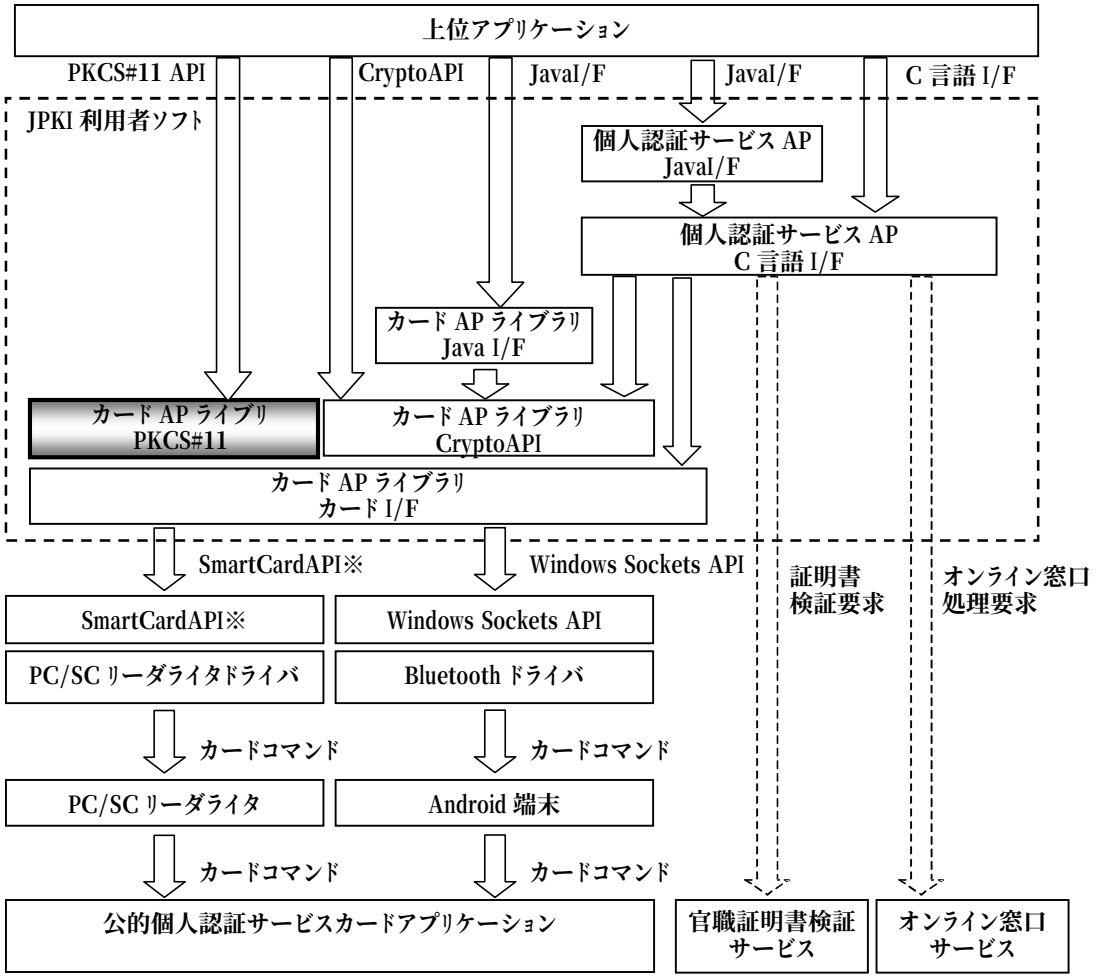
※ 最新の「個人番号カード対応適合性検証済み Android 端末一覧」の情報は、JPKI ポータルサイトに掲載するものとする。
また、最新の動作環境に掲載された OS 以外を使用した場合の不具合等に関するお問い合わせは、サポート対象外とする。

第４章　機能仕様

第１節　ソフトウェア構成図

本仕様書では、JPKI 利用者ソフトのうち、下図の太枠に示すカード AP ライブラリ(PKCS#11)の仕様をまとめる。

＜Windows 対応版＞



※ Smart Card Resource Manager API の略

図 4-1 ソフトウェア構成図（Windows 対応版）

＜MacOS 対応版＞

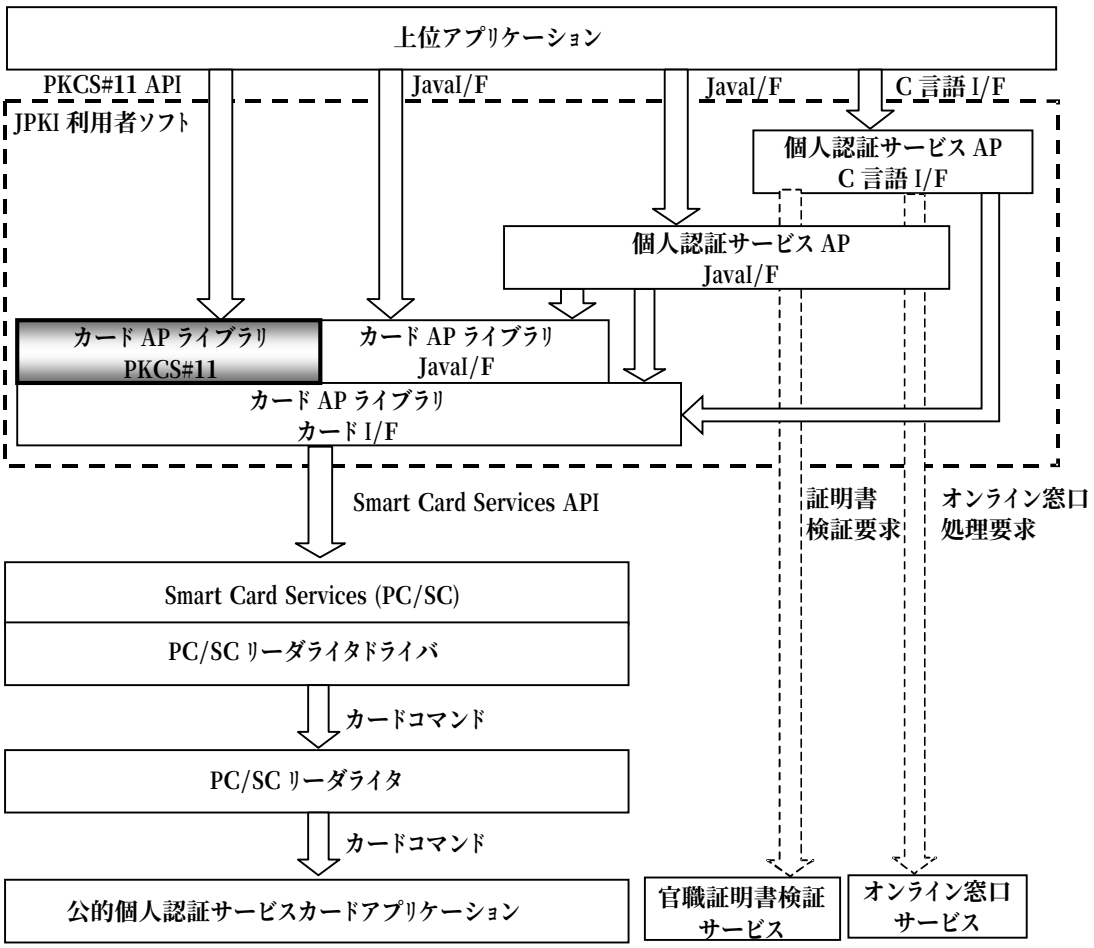


図 4-2 ソフトウェア構成図（MacOS 対応版）

第２節　ライブラリの構成

カード AP ライブラリ(PKCS#11)は、利用する IC カード、証明書の種別によってロードするライブラリを切り替える必要がある。

IC カード、証明書の種別とライブラリの対応を表 ４-１、表 ４-２に示す。

※パス定義項目名の記載内容は、ロード情報定義ファイルでライブラリの絶対パスを定義した項目名を表す。詳細は「第６章　第１節　ライブラリのロード方法」を参照。

※ロードするライブラリは１つのみとすること。利用する IC カードや証明書の種別を切り替える必要がある場合は、それまで使用していたライブラリを解放した上で新たなライブラリをロードすること。

表 ４-１ IC カード、証明書の種別とライブラリの対応（Windows 対応版）

NO	IC カード	証明書	ロードするライブラリ			
			ライブラリ名称		パス定義項目名	
					Windows (32bit)	Windows (64bit)
1	住基カード	署名用	住基カード対応 PKCS#11 ライブラリ		path	path64
2	個人番号カード	署名用	個人番号カード対応 PKCS#11 ライブラリ	個人番号カード署名対応 PKCS#11 ライブラリ	pathSign	pathSign64
3		利用者証明用		個人番号カード利用者証明対応 PKCS#11 ライブラリ	pathAuth	pathAuth64

表 ４-２ IC カード、証明書の種別とライブラリの対応（MacOS 対応版）

NO	IC カード	証明書	ロードするライブラリ		
			ライブラリ名称		パス定義項目名
					MacOS
1	個人番号カード	署名用	個人番号カード対応 PKCS#11 ライブラリ	個人番号カード署名対応 PKCS#11 ライブラリ	pathSign
2		利用者証明用		個人番号カード利用者証明対応 PKCS#11 ライブラリ	pathAuth

第３節　実現可能な機能の一覧

カード AP ライブラリ(PKCS#11)で実現可能な機能の一覧を表 ４-３に示す。

表 ４-３ 実現可能な機能の一覧

NO	機能	概要
1	証明書取得	IC カードに格納された電子証明書(利用者証明書、認証局の自己署名証明書)を取得する。
2	署名生成	署名対象データからハッシュ値を計算し、IC カードに格納された利用者秘密鍵を使用して電子署名を生成する。
3	署名検証	検証対象データからハッシュ値を計算し、ハッシュ値、電子署名、公開鍵を使用して電子署名を検証する。
4	繰り返し署名生成	NO2 の処理を繰り返し実行し、複数の署名対象データに対する電子署名を生成する。
5	繰り返し署名検証	NO3 の処理を繰り返し実行し、複数の電子署名を検証する。

第５章　API 仕様

第１節　サポート API 一覧

カード AP ライブラリ(PKCS#11)のサポート API の一覧を表 ５-１に示す。

表 ５-１ サポート API 一覧

NO	API 名	概要
1	C_GetFunctionList	関数ポインタリストを取得する。
2	C_Initialize	PKCS#11 ライブラリを初期化する。
3	C_Finalize	PKCS#11 ライブラリを終了する。
4	C_GetInfo	ライブラリ情報を取得する。
5	C_GetSlotList	スロットリストを取得する。
6	C_GetSlotInfo	スロット情報を取得する。
7	C_GetTokenInfo	トークン情報を取得する。
8	C_GetMechanismList	サポートメカニズム（アルゴリズム）を取得する。
9	C_GetMechanismInfo	メカニズム（アルゴリズム）情報を返す。
10	C_OpenSession	セッションを確立する。
11	C_CloseSession	セッションを切断する。
12	C_CloseAllSessions	すべてのセッションを切断する。
13	C_GetSessionInfo	セッション状態を取得する。
14	C_Login	トークンをログイン状態にする。
15	C_Logout	トークンをログアウト状態にする。
16	C_FindObjectsInit	オブジェクトの検索を開始する。
17	C_FindObjects	オブジェクトの検索を行う。
18	C_FindObjectsFinal	オブジェクトの検索を終了する。
19	C_GetAttributeValue	オブジェクトの属性値を取得する。
20	C_SignInit	署名処理を初期化する。
21	C_Sign	データに署名を行う。
22	C_DigestInit	ダイジェスト作成を開始する。
23	C_DigestUpdate	ダイジェストを作成する。
24	C_DigestFinal	ダイジェスト作成を終了する。
25	C_VerifyInit	署名検証を開始する。
26	C_Verify	署名値を検証する。
27	C_CreateObject	公開鍵オブジェクトを作成する。
28	C_DestroyObject	公開鍵オブジェクトを破棄する。

※NO.8,9,20,21,22,23,24,25,26 について、sha-256 の利用者証明書による署名生成および署名検証、sha-256 の官職証明書・職責証明書による署名検証に対応済み。

第２節　拡張 API 一覧

カード AP ライブラリ(PKCS#11)の拡張 API の一覧を表 ５-２拡張 API 一覧に示す。

この拡張 API については住基カードは提供対象外となる。

表 ５-２拡張 API 一覧

NO	API 名	概要
1	JPKIGetRemain	署名用または利用者証明用電子証明書のパスワードロックまでの残回数を取得する。

第３節　サポート API 仕様詳細

（1）C_GetFunctionList

API 名	C_GetFunctionList		
概要	関数ポインタリストを取得する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetFunctionList)(　CK_FUNCTION_LIST_PTR_PTR　ppFunctionList);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_FUNCTION_LIST_PTR_PTR	OUT	関数アドレスリストポインタ

（2）C_Initialize

API 名	C_Initialize		
概要	PKCS#11 ライブラリを初期化する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_Initialize)(　CK_VOID_PTR pReserved);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_VOID_PTR	IN	NULL ポインタを指定

（3）C_Finalize

API 名	C_Finalize		
概要	PKCS#11 ライブラリを終了する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_Finalize)(　CK_VOID_PTR pReserved);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_VOID_PTR	IN	NULL ポインタを指定

（4）C_GetInfo

API 名	C_GetInfo
概要	ライブラリ情報を取得する。
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetInfo)(　CK_INFO_PTR pInfo);

戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_INFO_PTR	IN/OUT	ライブラリ情報ポインタ
備考	取得可能なライブラリ情報は以下の通り。 CK_INFO::cryptokiVersion: PKCS11 規格バージョン：2.2 CK_INFO::manufacturerID: ライブラリ製造者名：「JPKI」 CK_INFO::description: ライブラリ記述文：「JPKI PKCS#11」 CK_INFO::libraryVersion: ライブラリバージョン:1.0		

（5）C_GetSlotList

API 名	C_GetSlotList		
概要	スロットリストを取得する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetSlotList)(CK_BBOOL tokenPresent, CK_SLOT_ID_PTR pSlotList, CK_ULONG_PTR pulCount);		
戻り値	CK_RV (CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_BBOOL	IN	TRUE: カード有りのスロットリストを返す FALSE: 接続されているすべてのスロットリストを返す
	CK_SLOT_ID_PTR	IN/OUT	スロット ID リストポインタ
	CK_ULONG_PTR	IN/OUT	スロット ID リスト件数

（6）C_GetSlotInfo

API 名	C_GetSlotInfo
概要	スロット情報を取得する。
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetSlotInfo)(CK_SLOT_ID slotID, CK_SLOT_INFO_PTR pInfo);
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）

	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_SLOT_INFO_PTR	IN/OUT	スロット情報ポインタ
備考	取得可能なスロット情報は以下の通り。 CK_SLOT_INFO::hardwareVersion: スロットハードウェアバージョン: 0.0 CK_SLOT_INFO::firmwareVersion: スロットファームウェアバージョン: 0.0 CK_SLOT_INFO::slotDescription: スロット記述文: IC カードリーダライタ名称（※） ※Android 端末の場合、IC カードリーダライタ名称は Shift_JIS コードで規定されている文字のみ使用できるものとする。 CK_SLOT_INFO::manufacturerID: スロット製造者名: なし（0 バイトの文字列） CK_SLOT_INFO::flags: カード有無		

（7）C_GetTokenInfo

API 名	C_GetTokenInfo		
概要	トークン情報を取得する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetTokenInfo)(CK_SLOT_ID slotID, CK_TOKEN_INFO_PTR pInfo);		
戻り値	CK_RV (CKR_OK: 成功 CKR_TOKEN_NOT_PRESENT: カードが挿入されていないまたはカードが抜かれた CKR_TOKEN_NOT_RECOGNIZED: 不正な IC カードを検出した CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_TOKEN_INFO_PTR	OUT	トークン情報ポインタ

（8）C_GetMechanismList

API 名	C_GetMechanismList		
概要	サポートメカニズム（アルゴリズム）を取得する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetMechanismList)(CK_SLOT_ID slotID, CK_MECHANISM_TYPE_PTR pMechanismList, CK_ULONG_PTR pulCount);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_MECHANISM_TYPE_PTR	OUT	メカニズムタイプポインタ
	CK_ULONG_PTR	IN/OUT	メカニズムタイプ件数
備考	取得可能なサポートメカニズムは以下の通り。 Sign、Verify 用: CKM_RSA_PKCS Digest 用: CKM_SHA_1、CKM_SHA256（※） ※CKM_SHA_1 は Windows 版のみ取得可能。		

（9）C_GetMechanismInfo

API 名	C_GetMechanismInfo		
概要	メカニズム（アルゴリズム）情報を返す。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetMechanismInfo)(CK_SLOT_ID slotID, CK_MECHANISM_TYPE type, CK_MECHANISM_INFO_PTR pInfo);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_MECHANISM_TYPE	IN	メカニズムタイプ
	CK_MECHANISM_INFO_PTR	IN/OUT	メカニズム情報ポインタ

備考	設定する情報は以下の通り。 type = CKM_RSA_PKCS の場合 CK_MECHANISM_INFO::ulMinKeySize: 1024 CK_MECHANISM_INFO::ulMaxKeySize: 2048 CK_MECHANISM_INFO::flags: CKF_VERIFY ¦ CKF_SIGN ¦ CKF_HW type = CKM_SHA_1 の場合（※） CK_MECHANISM_INFO::ulMinKeySize: 0 CK_MECHANISM_INFO::ulMaxKeySize: 0 CK_MECHANISM_INFO::flags: CKF_DIGEST type = CKM_SHA256 の場合 CK_MECHANISM_INFO::ulMinKeySize: 0 CK_MECHANISM_INFO::ulMaxKeySize: 0 CK_MECHANISM_INFO::flags: CKF_DIGEST ※CKM_SHA_1 は Windows 版のみ設定可能。

（１０）C_OpenSession

API 名	C_OpenSession		
概要	セッションを確立する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_OpenSession)(CK_SLOT_ID slotID, CK_FLAGS flags, CK_VOID_PTR pApplication, CK_NOTIFY Notify, CK_SESSION_HANDLE_PTR phSession);		
戻り値	CK_RV (CKR_OK: 成功 CKR_TOKEN_NOT_PRESENT: カードが挿入されていないまたはカードが抜かれた CKR_TOKEN_NOT_RECOGNIZED: 不正な IC カードを検出した CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID
	CK_FLAGS	IN	0 または CKF_SERIAL_SESSION を指定
	CK_VOID_PTR	IN	NULL ポインタを指定

	型	I/O	内容
引数	CK_NOTIFY	IN	NULL ポインタを指定
	CK_SESSION_HANDLE_PTR	IN/OUT	セッションハンドルポインタ

（１１）C_CloseSession

API 名	C_CloseSession		
概要	セッションを切断する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_CloseSession)(CK_SESSION_HANDLE hSession);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル

（１２）C_CloseAllSessions

API 名	C_CloseAllSessions		
概要	すべてのセッションを切断する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_CloseAllSessions)(CK_SLOT_ID slotID);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_SLOT_ID	IN	スロット ID

（１３）C_GetSessionInfo

API 名	C_GetSessionInfo		
概要	セッション状態を取得する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetSessionInfo)(CK_SESSION_HANDLE　hSession, CK_SESSION_INFO_PTR pInfo);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションタイプ
	CK_SESSION_INFO_PTR	OUT	セッション状態ポインタ

備考	以下の状態を返す。 ログインしていないとき: CKS_RO_PUBLIC_SESSION ログインしているとき: CKS_RO_USER_FUNCTIONS

（１４）C_Login

API 名	C_Login		
概要	トークンをログイン状態にする（証明書 DF を活性化する）。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_Login)(CK_SESSION_HANDLE hSession, CK_USER_TYPE userType, CK_CHAR_PTR pPin, CK_ULONG ulPinLen);		
戻り値	CK_RV (CKR_OK: 成功 CKR_DEVICE_REMOVED: カードが挿入されていないまたはカードが抜かれた CKR_PIN_INCORRECT: パスワード指定誤り CKR_PIN_LOCKED: パスワードがロックされている CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_USER_TYPE	IN	ユーザタイプ
	CK_CHAR_PTR	IN	パスワード文字列ポインタ
	CK_ULONG	IN	パスワード文字列長

（１５）C_Logout

API 名	C_Logout		
概要	トークンをログアウト状態にする。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_Logout)(CK_SESSION_HANDLE hSession);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル

（１６）C_FindObjectsInit

API 名	C_FindObjectsInit		
概要	オブジェクトの検索を開始する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_FindObjectsInit)(CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount);		
戻り値	CK_RV (CKR_OK: 成功 CKR_DEVICE_REMOVED: カードが挿入されていないまたはカードが抜かれた CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_ATTRIBUTE_PTR	IN	属性テーブルポインタ
	CK_ULONG	IN	属性テーブル数
備考	以下の属性による検索を行う。 CKA_CLASS CKO_CERTIFICATE または CKO_PRIVATE_KEY CKA_ID Login で読み出したときに生成した番号 CKA_TOKEN 真 CKA_LABEL 証明書の名前 CKA_VALUE 証明書の値 CKA_MODULUS 利用者公開鍵の N CKA_PUBLIC_EXPONENT 利用者公開鍵の E		

（１７）C_FindObjects

API 名	C_FindObjects
概要	オブジェクトの検索を行う。
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_FindObjects)(CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE_PTR phObject, CK_ULONG ulMaxObjectCount, CK_ULONG_PTR pulObjectCount);

戻り値	CK_RV (CKR_OK: 成功 CKR_DEVICE_REMOVED: カードが挿入されていないまたはカードが抜かれた CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_OBJECT_HANDLE_PTR	IN/OUT	オブジェクトハンドルポインタ
	CK_ULONG	IN	最大オブジェクト数
	CK_ULONG_PTR	IN/OUT	オブジェクト数ポインタ

（１８）C_FindObjectsFinal

API 名	C_FindObjectsFinal		
概要	オブジェクトの検索を終了する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_FindObjectsFinal)(CK_SESSION_HANDLE hSession);		
戻り値	CK_RV (CKR_OK: 成功 CKR_DEVICE_REMOVED: カードが挿入されていないまたはカードが抜かれた CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル

（１９）C_GetAttributeValue

API 名	C_GetAttributeValue
概要	オブジェクトの属性値を取得する。
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_GetAttributeValue)(CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount);

<table>
<tr><td>戻り値</td><td colspan="3">CK_RV（
CKR_OK: 成功
CKR_DEVICE_REMOVED:
カードが挿入されていないまたはカードが抜かれた
CKR_FUNCTION_FAILED: 失敗
）</td></tr>
<tr><td></td><td>型</td><td>I/O</td><td>内容</td></tr>
<tr><td>引数</td><td>CK_SESSION_HANDLE</td><td>IN</td><td>セッションハンドル</td></tr>
<tr><td></td><td>CK_OBJECT_HANDLE</td><td>IN</td><td>オブジェクトハンドル</td></tr>
<tr><td></td><td>CK_ATTRIBUTE_PTR</td><td>OUT</td><td>属性テーブルポインタ</td></tr>
<tr><td></td><td>CK_ULONG</td><td>OUT</td><td>属性テーブル数</td></tr>
<tr><td>備考</td><td colspan="3">以下の属性に対する値が取得可能である。
CKA_CLASS CKO_CERTIFICATE または CKO_PRIVATE_KEY
CKA_LABEL 表 5-3オブジェクト情報一覧参照。
CKA_VALUE 表 5-3オブジェクト情報一覧参照。
CKA_ISSUER 表 5-3オブジェクト情報一覧参照。
CKA_SERIAL_NUMBER 表 5-3オブジェクト情報一覧参照。
CKA_SUBJECT 表 5-3オブジェクト情報一覧参照。
CKA_ID 表 5-3オブジェクト情報一覧参照。
CKA_TOKEN True
CKA_PRIVATE True
CKA_CERTFICATE_TYPE CKC_X_509
CKA_MODULUS 表 5-3オブジェクト情報一覧参照。
CKA_MODULUS_BITS 表 5-3オブジェクト情報一覧参照。
CKA_PUBLIC_EXPONENT 表 5-3オブジェクト情報一覧参照。
CKA_KEY_TYPE CKK_RSA
CKA_SENSITIVE True
CKA_SIGN True</td></tr>
</table>

表 5-3 オブジェクト属性一覧

#	オブジェクト	属性名	属性値
1	利用者公開鍵	CKA_LABEL	USERKEY
		CKA_ID	住基カード対応 PKCS#11 ライブラリの場合： N の SHA1 ハッシュ 個人番号カード対応 PKCS#11 ライブラリの場合： N の SHA256 ハッシュ
		CKA_PUBLIC_EXPONENT	利用者証明書から取得した値
		CKA_MODULUS	利用者証明書から取得した値
		CKA_MODULUS_BITS	利用者証明書から取得した値
2	利用者証明書	CKA_LABEL	USERCERT
		CKA_ID	住基カード対応 PKCS#11 ライブラリの場合： N の SHA1 ハッシュ 個人番号カード対応 PKCS#11 ライブラリの場合： N の SHA256 ハッシュ
		CKA_VALUE	証明書自体
		CKA_SUBJECT	利用者証明書から取得した値
		CKA_ISSUER	利用者証明書から取得した値
		CKA_SERIAL_NUMBER	利用者証明書から取得した値
3	認証局の自己署名証明書	CKA_LABEL	CACERT
		CKA_ID	住基カード対応 PKCS#11 ライブラリの場合： N の SHA1 ハッシュ 個人番号カード対応 PKCS#11 ライブラリの場合： N の SHA256 ハッシュ
		CKA_VALUE	証明書自体
		CKA_SUBJECT	認証局の自己署名証明書から取得した値
		CKA_ISSUER	認証局の自己署名証明書から取得した値

#	オブジェクト	属性名	属性値
		CKA_SERIAL_NUMBER	認証局の自己署名証明書から取得した値

（２０）C_SignInit

API 名	C_SignInit		
概要	署名処理を初期化する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_SignInit)(CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey);		
戻り値	CK_RV (CKR_OK: 成功 CKR_DEVICE_REMOVED: カードが挿入されていないまたはカードが抜かれた CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_MECHANISM_PTR	IN	メカニズム情報ポインタ mechanism: CKM_RSA_PKCS のみ指定可
	CK_OBJECT_HANDLE	IN	オブジェクトハンドル

（２１）C_Sign

API 名	C_Sign
概要	データに署名を行う。
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_Sign)(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen);

戻り値	CK_RV (CKR_OK: 成功 CKR_DEVICE_REMOVED: カードが挿入されていないまたはカードが抜かれた CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_BYTE_PTR	IN	データポインタ
	CK_ULONG	IN	データ長
	CK_BYTE_PTR	IN/OUT	署名データポインタ
	CK_ULONG_PTR	IN/OUT	署名データ長ポインタ ※署名データ長はカード種別によって変化する。第６節　コーリングシーケンスに従って署名データ長取得の上、十分なメモリ領域を署名データポインタに指定すること。

（２２）C_DigestInit

API 名	C_DigestInit		
概要	ダイジェスト作成を開始する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_DigestInit)(CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_MECHANISM_PTR	IN	メカニズム情報ポインタ mechanism: CKM_SHA_1 または CKM_SHA256 いずれかを指定可（※） ※CKM_SHA_1 は Windows 版のみ指定可能。

（２３）C_DigestUpdate

API 名	C_DigestUpdate

概要	ダイジェストを作成する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_DigestUpdate)(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_BYTE_PTR	IN	ハッシュするデータポインタ
	CK_ULONG	IN	ハッシュするデータ長

（２４）C_DigestFinal

API 名	C_DigestFinal		
概要	ダイジェスト作成を終了する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_DigestFinal)(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pDigest, CK_ULONG_PTR pulDigestLen);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_BYTE_PTR	IN/OUT	ダイジェストデータポインタ
	CK_ULONG_PTR	IN/OUT	ダイジェストデータ長ポインタ ※ダイジェストデータ長は C_DigestInit で指定したメカニズム情報によって変化する。第６節　コーリングシーケンスに従ってダイジェストデータ長取得の上、十分なメモリ領域をダイジェストデータポインタに指定すること。

（２５）C_VerifyInit

API 名	C_VerifyInit
概要	署名検証を開始する。

関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_VerifyInit)(CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism CK_OBJECT_HANDLE hKey);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_MECHANISM_PTR	IN	メカニズム情報ポインタ mechanism: CKM_RSA_PKCS のみ指定可
	CK_OBJECT_HANDLE	IN	RSA 公開鍵オブジェクト

（２６）C_Verify

API 名	C_Verify		
概要	署名値を検証する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_Verify)(CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen);		
戻り値	CK_RV (CKR_OK: 成功 CKR_SIGNATURE_INVALID: 署名データ不正 CKR_FUNCTION_FAILED: 失敗)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_BYTE_PTR	IN	検証するデータポインタ
	CK_ULONG	IN	検証するデータ長
	CK_BYTE_PTR	IN	署名データポインタ
	CK_ULONG	IN	署名データ長

備考	pData と pSigneture の OID の関係は以下の通り。 署名データに OID をつける場合は、上位 UP にて検証するデータに OID を付加したデータを入力しなければならない。

		検証するデータ(pData)	
		OIDあり	OIDなし
署名データ (pSigneture)	OIDあり	OK	NG
	OIDなし	NG	OK

（２７）C_CreateObject

API 名	C_CreateObject		
概要	公開鍵オブジェクトを作成する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_CreateObject)(CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phObject);		
戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_ATTRIBUTE_PTR	IN	属性テーブルポインタ
	CK_ULONG	IN	属性テーブル数
	CK_OBJECT_HANDLE_PTR	IN/OUT	オブジェクトハンドルポインタ
備考	本 API では RSA 公開鍵セッションオブジェクトのみ作成できる。 以下の属性を pTemplate で指定する。 CK_OBJECT_CLASS: CKO_PUBLIC_KEY（※１） CKA_PUBLIC_EXPONENT CKA_MODULUS ※１ :CK_OBJECT_CLASS の値は CKO_PUBLIC_KEY 固定とする。 その他の属性は使用不可とする。		

（２８）C_DestroyObject

API 名	C_DestroyObject
概要	公開鍵オブジェクトを破棄する。
関数インターフェース	CK_DEFINE_FUNCTION(CK_RV, C_DestroyObject)(CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject);

戻り値	CK_RV（CKR_OK: 成功　CKR_FUNCTION_FAILED: 失敗）		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_OBJECT_HANDLE	IN	オブジェクトハンドル
備考	本 API では RSA 公開鍵セッションオブジェクトのみ破棄できる。		

第４節　拡張 API 仕様一覧

（1）JPKIGetRemain

API 名	JPKIGetRemain		
概要	署名用または利用者証明用電子証明書のパスワードロックまでの残回数を取得する。		
関数インターフェース	CK_DEFINE_FUNCTION(CK_LONG,JPKIGetRemain)(CK_SESSION_HANDLE hSession, CK_USER_TYPE userType);		
戻り値	CK_LONG (JPKI_GETREMAIN_LOCKED：パスワードロックまでの残回数 0 （ロック状態） JPKI_GETREMAIN_MIN (=1)以上 JPKI_GETREMAIN_MAX (=15)以下： パスワードロックまでの残回数 1～15 （戻り値の数値が、残回数を示す 1～15 の値となる） JPKI_GETREMAIN_UNLIMITED：パスワードロックまでの残回数制限なし JPKI_GETREMAIN_DEVICE_REMOVED： カードまたはリーダライタが認識できなくなった JPKI_GETREMAIN_NG：その他エラー)		
	型	I/O	内容
引数	CK_SESSION_HANDLE	IN	セッションハンドル
	CK_USER_TYPE	IN	ユーザタイプ
備考	マイナンバーカードでは、署名用電子証明書のパスワードは５回、利用者証明用電子証明書のパスワードは３回をそれぞれ連続で誤った場合パスワードロックする仕様である。		

第５節　構造体仕様

利用者クライアントソフトは、PKCS#11 Ver2.2 の仕様書の定義/構造体に準拠しています。
構造体は下記 URL にある PKCS#11 Ver2.2 の資料を参考にしてください。

PKCS#11: Cryptographic Token Interface Standard

「https://www.cryptsoft.com/pkcs11doc/STANDARD/pkcs-11v2-20.pdf」

表 ５-４に本ライブラリで使用する構造体の一覧を示す。

表 ５-４ PKCS#11 構造体一覧

NO	構造体名	概要
1	CK_INFO CK_INFO_PTR	PKCS ライブラリ情報
2	CK_SLOT_ID CK_SLOT_ID_PTR	スロット ID
3	CK_SLOT_INFO CK_SLOT_INFO_PTR	スロット情報
4	CK_TOKEN_INFO CK_TOKEN_INFO_PTR	トークン情報
5	CK_SESSION_HANDLE CK_SESSION_HANDLE_PTR	セッションハンドル
6	CK_USER_TYPE	ユーザタイプ
7	CK_SESSION_INFO CK_SESSION_INFO_PTR	セッション情報
8	CK_OBJECT_HANDLE CK_OBJECT_HANDLE_PTR	オブジェクトハンドル
9	CK_OBJECT_CLASS CK_OBJECT_CLASS_PTR	オブジェクトクラス
10	CK_ATTRIBUTE CK_ATTRIBUTE_PTR	属性タイプ、値、長さを含む構造体
11	CK_MECHANISM_TYPE CK_MECHANISM_TYPE_PTR	メカニズムタイプ
12	CK_MECHANISM CK_MECHANISM_PTR	メカニズムタイプを含む、メカニズムを示す構造体
13	CK_MECHANISM_INFO	メカニズム情報

NO	構造体名	概要
	CK_MECHANISM_INFO_PTR	
14	CK_RV	ライブラリの戻り値
15	CK_NOTIFY	コールバック情報
16	CK_FUNCTION_LIST CK_FUNCTION_LIST_PTR CK_FUNCTION_LIST_PTR_PTR	PKCS ライブラリ関数

第６節　コーリングシーケンス

「第４章　第３節　実現可能な機能の一覧」を実現するためのコーリングシーケンスを以下に示す。上位アプリケーションは、このコーリングシーケンスに沿って実装すること。

注意事項

・複数のコーリングシーケンスを並行して実行しないこと。
　１つのシーケンスを開始したなら、そのシーケンスを完了してから次のシーケンスを開始してください。
・JPKIGetRemain 関数の呼び出しはパスワードロックまでの残回数取得を行う場合に実施すること。

※任意関数であり、残回数取得が不要の場合は省略してよい。

（1）初期処理

C_GetFunctionList	関数ポインタリストの取得 ppFunctionList: API アドレスポインタ
↓	
C_Initialize	PKCS#11 ライブラリの初期化 pReserved: NULL
↓	
C_GetSlotList	スロットリスト数の取得 tokenPresent: TRUE pSlotList: NULL pulCount: スロット数格納アドレス
↓	
C_GetSlotList	スロットリストの取得 tokenPresent: TRUE pSlotList: スロットリスト格納アドレス pulCount: スロット数格納アドレス
↓	
C_GetSlotInfo	スロット情報の取得 slotID: スロット ID pInfo: スロット情報格納アドレス
↓	
C_GetTokenInfo	トークン情報の取得 slotID: スロット ID pInfo: トークン情報格納アドレス

↓

C_OpenSession	アプリケーションとトークン間のセッションの確立 slotID: スロット ID flags: 0 pApplication: NULL Notify: NULL phSession: セッションハンドル格納アドレス

↓

残回数取得を行う場合に実施（任意）

JPKIGetRemain	パスワードロックまでの残回数取得 hSession: C_OpenSession で取得したハンドル userType: CKU_USER

↓

C_Login	トークンへのログイン hSession: C_OpenSession で取得したハンドル userType: CKU_USER pPin: パスワード文字列 ulPinLen: パスワード文字列長

↓

更新された残回数取得を行う場合に実施（任意）

JPKIGetRemain	パスワードロックまでの残回数取得 hSession: C_OpenSession で取得したハンドル userType: CKU_USER

（2）終了処理

C_Logout	トークンからのログアウト hSession: C_OpenSession で取得したハンドル

↓

C_CloseSession もしくは、 C_CloseAllSessions	セッションの切断 C_CloseSession の場合 hSession: C_OpenSession で取得したハンドル C_CloseAllSessions の場合 slotID: C_OpenSession に指定したスロット ID

↓

C_Finalize	PKCS#11 ライブラリの終了処理 pReseved: NULL

（3）証明書取得処理

初期化処理	（1）初期処理参照

↓

C_FindObjectsInit	証明書検索操作の初期設定 hSession: C_OpenSession で取得したハンドル pTemplate: 以下の属性を指定 (1) type = CKA_CLASS value = CKO_CERTIFICATE (2) type = CKA_TOKEN value = TRUE ulCount: 設定する属性数:2

↓

C_FindObjects	証明書の検索 hSession: C_OpenSession で取得したハンドル phObject: オブジェクトハンドル格納アドレス ulMaxObjectCount: オブジェクトハンドル格納領域数:2 pulObjectCount: 発見したオブジェクト数格納アドレス

↓

①（pulObjectCount – 1）回分ループ
（カウンタを i、初期値を 0 とする）

↓

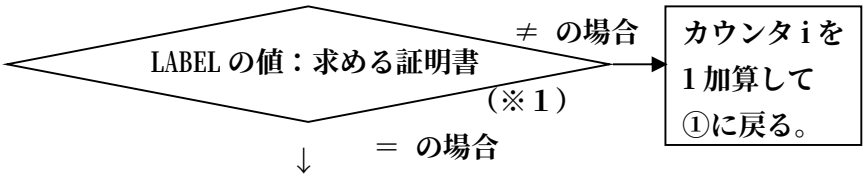
C_GetAttributeValue	証明書サイズの取得 hSession: C_OpenSession で取得したハンドル hObject: C_FindObjects で取得したオブジェクトハンドルリストの i 番目(phObject[i]) pTemplate: 以下の属性を指定 (1) type = CKA_LABEL（※1） value = NULL ulValueLen = サイズ格納領域アドレス (2) type = CKA_VALUE value = NULL ulValueLen = サイズ格納領域アドレス ulCount: 設定する属性数:2

↓

C_GetAttributeValue	証明書の取得 hSession: C_OpenSession で取得したハンドル

	hObject: C_FindObjects で取得したオブジェクトハンドルリストの i 番目(phObject[i]) pTemplate: 以下の属性を指定 (1) type = CKA_LABEL（※１） value = ラベル格納アドレス ulValueLen =格納領域サイズ (2) type = CKA_VALUE value = 証明書格納アドレス ulValueLen = 格納領域サイズ ulCount: 設定する属性数:2

↓



↓

C_FindObjectsFinal	証明書検索操作の終了処理 hSession: C_OpenSession で取得したハンドル

↓

終了処理	（２）終了処理参照

（４）署名生成処理(署名対象データを渡すパターン)[*1]

初期化処理	（１）初期処理参照

↓

C_FindObjectsInit	証明書検索操作の初期設定 hSession: C_OpenSession で取得したハンドル pTemplate: 以下の属性を指定 (1) type = CKA_CLASS value = CKO_CERTIFICATE (2) type = CKA_TOKEN value = TRUE ulCount: 設定する属性数:2

↓

C_FindObjects	証明書の検索 hSession: C_OpenSession で取得したハンドル phObject: オブジェクトハンドル格納アドレス ulMaxObjectCount: オブジェクトハンドル格納領域数:2

*1 署名生成処理のコーリングシーケンスを実行することで、IC カード内の利用者証明書、署名対象データに対するハッシュ値および署名値を取得することができる。

	pulObjectCount: 発見したオブジェクト数格納アドレス

↓

①（pulObjectCount – 1）回分ループ
（カウンタを i、初期値を 0 とする）

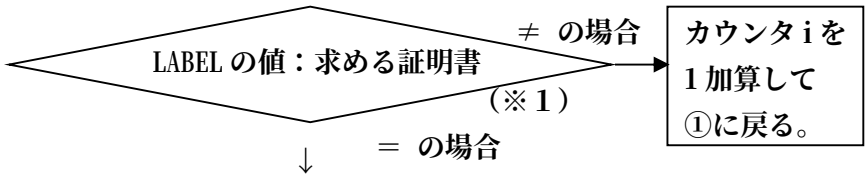
↓

C_GetAttributeValue	証明書サイズの取得 hSession: C_OpenSession で取得したハンドル hObject: C_FindObjects で取得したオブジェクトハンドルリストの i 番目(phObject[i]) pTemplate:以下の属性を指定 (1) type = CKA_LABEL（※１） value = NULL ulValueLen = サイズ格納領域アドレス (2) type = CKA_VALUE value = NULL ulValueLen = サイズ格納領域アドレス ulCount:設定する属性数:2

↓

C_GetAttributeValue	証明書の取得 hSession: C_OpenSession で取得したハンドル hObject: C_FindObjects で取得したオブジェクトハンドルリストの i 番目(phObject[i]) pTemplate:以下の属性を指定 (1) type = CKA_LABEL（※１） value = ラベル格納アドレス ulValueLen =格納領域サイズ (2) type = CKA_VALUE value = 証明書格納アドレス ulValueLen = 格納領域サイズ ulCount:設定する属性数:2

↓



↓

C_FindObjectsFinal	証明書検索操作の終了処理 hSession: C_OpenSession で取得したハンドル

↓

利用者証明書から公開鍵の E と N を取得

↓

C_FindObjectsInit	利用者鍵検索操作の初期設定 hSession: C_OpenSession で取得したハンドル pTemplate: 以下の属性を指定する CKO_PRIVATE_KEY の指定は必須。他は任意。 (1) type = CKA_CLASS value = CKO_PRIVATE_KEY (2) type = CKA_TOKEN value = TRUE (3) type = CKA_MODULUS value = 公開鍵の N (4) type = CKA_PUBLIC_EXPONENT value = 公開鍵の E ulCount: 設定する属性数:4

↓

C_FindObjects	利用者鍵の検索 hSession: C_OpenSession で取得したハンドル phObject: 利用者鍵オブジェクトハンドル ulMaxObjectCount: オブジェクトハンドル格納領域数:1 pulObjectCount: 発見したオブジェクト数格納アドレス

↓

C_FindObjectsFinal	利用者鍵の検索操作の終了処理 hSession: C_OpenSession で取得したハンドル

↓

C_DigestInit	ダイジェスト作成開始 hSession: C_OpenSession で取得したハンドル pMechanism: CKM_SHA_1 または CKM_SHA256 のいずれかを署名方式に合わせて指定（※） ※CKM_SHA_1 は Windows 版のみ指定可能。

↓

C_DigestUpdate	データをハッシュ hSession: C_OpenSession で取得したハンドル pPart: 署名対象データ ulPartLen: 署名対象データ長

↓

C_DigestFinal	ダイジェスト格納領域サイズ取得 hSession: C_OpenSession で取得したハンドル pDigest: NULL

	pulDigestLen: ダイジェストデータ長格納アドレス

↓

C_DigestFinal	ダイジェスト取得 hSession: C_OpenSession で取得したハンドル pDigest: ダイジェスト値格納アドレス pulDigestLen: ダイジェスト値長格納アドレス

↓

C_DigestFinal で取得したダイジェスト値（pDigest）を使用して DigestInfo を作成

↓

C_SignInit	署名処理の初期設定 hSession: C_OpenSession で取得したハンドル pMechanism: CKM_RSA_PKCS hKey: C_FindObjects で取得したオブジェクトハンドル

↓

C_Sign	署名値長取得 hSession: C_OpenSession で取得したハンドル pData: DigestInfo ulDataLen: DigestInfo のデータ長 pSigneture: NULL pulSignetureLen: 署名値長格納アドレス

↓

C_Sign	署名 hSession: C_OpenSession で取得したハンドル pData: DigestInfo ulDataLen: DigestInfo のデータ長 pSigneture: 署名値格納アドレス pulSignetureLen: 署名値長格納アドレス

↓

終了処理	（2）終了処理参照

（5）署名生成処理(ハッシュ値を渡すパターン)[*2]

初期化処理	（1）初期処理参照

↓

C_FindObjectsInit	証明書検索操作の初期設定 hSession: C_OpenSession で取得したハンドル pTemplate: 以下の属性を指定

[*2] 署名生成処理(ハッシュ値を渡すパターン)のコーリングシーケンスを実行することで、IC カード内の利用者証明書、ハッシュ値に対する署名値を取得することができる。

	(1) type = CKA_CLASS value = CKO_CERTIFICATE (2) type = CKA_TOKEN value = TRUE ulCount: 設定する属性数:2

↓

C_FindObjects	証明書の検索 hSession: C_OpenSession で取得したハンドル phObject: オブジェクトハンドル格納アドレス ulMaxObjectCount: オブジェクトハンドル格納領域数:2 pulObjectCount: 発見したオブジェクト数格納アドレス

↓

①（pulObjectCount – 1）回分ループ
（カウンタを i、初期値を 0 とする）

↓

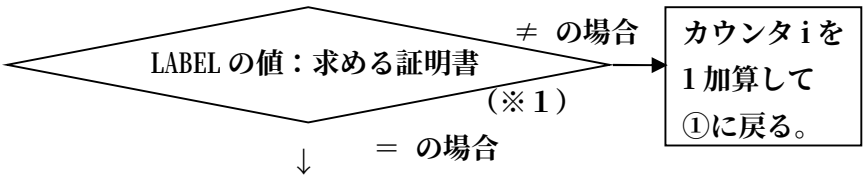
C_GetAttributeValue	証明書サイズの取得 hSession: C_OpenSession で取得したハンドル hObject: C_FindObjects で取得したオブジェクトハンドルリストの i 番目(phObject[i]) pTemplate:以下の属性を指定 (1) type = CKA_LABEL（※１） value = NULL ulValueLen = サイズ格納領域アドレス (2) type = CKA_VALUE value = NULL ulValueLen = サイズ格納領域アドレス ulCount:設定する属性数:2

↓

C_GetAttributeValue	証明書の取得 hSession: C_OpenSession で取得したハンドル hObject: C_FindObjects で取得したオブジェクトハンドルリストの i 番目(phObject[i]) pTemplate:以下の属性を指定 (1) type = CKA_LABEL（※１） value = ラベル格納アドレス ulValueLen =格納領域サイズ (2) type = CKA_VALUE value = 証明書格納アドレス ulValueLen = 格納領域サイズ

	ulCount:設定する属性数:2

↓



↓

C_FindObjectsFinal	証明書検索操作の終了処理 hSession: C_OpenSession で取得したハンドル

↓

利用者証明書から公開鍵の E と N を取得

↓

C_FindObjectsInit	利用者鍵検索操作の初期設定 hSession: C_OpenSession で取得したハンドル pTemplate: 以下の属性を指定する CKO_PRIVATE_KEY の指定は必須。他は任意。 (1) type = CKA_CLASS value = CKO_PRIVATE_KEY (2) type = CKA_TOKEN value = TRUE (3) type = CKA_MODULUS value = 公開鍵の N (4) type = CKA_PUBLIC_EXPONENT value = 公開鍵の E ulCount: 設定する属性数:4

↓

C_FindObjects	利用者鍵の検索 hSession: C_OpenSession で取得したハンドル phObject: 利用者鍵オブジェクトハンドル ulMaxObjectCount: オブジェクトハンドル格納領域数:1 pulObjectCount: 発見したオブジェクト数格納アドレス

↓

C_FindObjectsFinal	利用者鍵の検索操作の終了処理 hSession: C_OpenSession で取得したハンドル

↓

ハッシュ値を使用して DigestInfo を作成

↓

C_SignInit	署名処理の初期設定 hSession: C_OpenSession で取得したハンドル

	pMechanism: CKM_RSA_PKCS hKey: C_FindObjects で取得したオブジェクトハンドル

↓

C_Sign	署名値長取得 hSession: C_OpenSession で取得したハンドル pData: DigestInfo ulDataLen: DigestInfo のデータ長 pSigneture: NULL pulSignetureLen: 署名値長格納アドレス

↓

C_Sign	署名 hSession: C_OpenSession で取得したハンドル pData: DigestInfo ulDataLen: DigestInfo のデータ長 pSigneture: 署名値格納アドレス pulSignetureLen: 署名値長格納アドレス

↓

終了処理	（2）終了処理参照

（6）署名検証処理(検証対象データを渡すパターン)

初期化処理	（1）初期処理参照

↓

電子証明書から公開鍵の E と N を取得

↓

C_CreateObject	公開鍵オブジェクト作成 hSession: C_OpenSession で取得したハンドル pTemplate: 以下を指定する (1) type = CKA_CLASS value = CKO_PUBLIC_KEY (2) type = CKA_PUBLIC_EXPORNENT value = 公開鍵の E (3) type = CKA_MODULUS value = 公開鍵の N ulCount: 設定する属性数:3

↓

C_DigestInit	ダイジェスト作成開始 hSession: C_OpenSession で取得したハンドル pMechanism: CKM_SHA_1 または CKM_SHA256 を署名方式

	に合わせて指定（※） ※CKM_SHA_1 は Windows 版のみ指定可能。
↓	
C_DigestUpdate	データをハッシュ hSession: C_OpenSession で取得したハンドル pPart: 検証対象データ ulPartLen: 検証対象データ長
↓	
C_DigestFinal	ダイジェスト格納領域サイズ取得 hSession: C_OpenSession で取得したハンドル pDigest: NULL pulDigestLen: ダイジェスト値長格納アドレス
↓	
C_DigestFinal	ダイジェスト取得 hSession: C_OpenSession で取得したハンドル pDigest: ダイジェスト値格納アドレス pulDigestLen: ダイジェスト値長格納アドレス
↓	
C_DigestFinal で取得したダイジェスト値（pDigest）を使用して DigestInfo を作成	
↓	
C_VerifyInit	データ検証開始 hSession: C_OpenSession で取得したハンドル pMechanism: CKM_RSA_PKCS hKey: 公開鍵オブジェクトハンドル
↓	
C_Verify	データ検証 hSession: C_OpenSession で取得したハンドル pData: DigestInfo ulDataLen: DigestInfo のデータ長 pSigneture: 署名値 pulSignetureLen: 署名値長
↓	
C_DestroyObject	公開鍵オブジェクト解放 hSession: C_OpenSession で取得したハンドル hObject: 対象オブジェクトハンドル
↓	
終了処理	（2）終了処理参照

（7）署名検証処理(ハッシュ値を渡すパターン)

初期化処理	（1）初期処理参照

↓

電子証明書から公開鍵の E と N を取得

↓

C_CreateObject	公開鍵オブジェクト作成 hSession: C_OpenSession で取得したハンドル pTemplate: 以下を指定する (1) type = CKA_CLASS value = CKO_PUBLIC_KEY (2) type = CKA_PUBLIC_EXPORNENT value = 公開鍵の E (3) type = CKA_MODULUS value = 公開鍵の N ulCount: 設定する属性数:3

↓

ハッシュ値を使用して DigestInfo を作成

↓

C_VerifyInit	データ検証開始 hSession: C_OpenSession で取得したハンドル pMechanism: CKM_RSA_PKCS hKey: 公開鍵オブジェクトハンドル

↓

C_Verify	データ検証 hSession: C_OpenSession で取得したハンドル pData: DigestInfo ulDataLen: DigestInfo のデータ長 pSigneture: 署名値 pulSignetureLen: 署名値長

↓

C_DestroyObject	公開鍵オブジェクト解放 hSession: C_OpenSession で取得したハンドル hObject: 対象オブジェクトハンドル

↓

終了処理	（2）終了処理参照

（8）繰り返し署名生成処理(署名対象データを渡すパターン)

「（4）署名生成処理(署名対象データを渡すパターン)」の網掛け部分を署名対象データの個数分だけ繰り返して呼び出す。

（9）繰り返し署名生成処理(ハッシュ値を渡すパターン)

「（5）署名生成処理(ハッシュ値を渡すパターン)」の網掛け部分を署名対象データの個数分だけ繰り返して呼び出す。

（１０）繰り返し署名検証処理(検証対象データを渡すパターン)

「（6）署名検証処理(検証対象データを渡すパターン)」の網掛け部分を検証対象データの個数分だけ繰り返して呼び出す。(但し、すべての電子署名が同一の秘密鍵で生成された場合とする。)

（１１）繰り返し署名検証処理(ハッシュ値を渡すパターン)

「（7）署名検証処理(ハッシュ値を渡すパターン)」の網掛け部分を検証対象データの個数分だけ繰り返して呼び出す。(但し、すべての電子署名が同一の秘密鍵で生成された場合とする。)

第６章　その他

第１節　ライブラリのロード方法

汎用受付システム等の上位アプリケーションでは、利用する IC カード、証明書の種別に対応した PKCS#11 ライブラリのパス名を固定のファイル（ロード情報定義ファイル）から取得し、ロードする方式とする。

ロード情報定義ファイルのパス名およびファイル名は以下の通り。

＜Windows 対応版＞

C:¥Program Files¥Common Files¥e-gov_app¥load_path¥default.dat

（上記は OS のインストール先が C ドライブの場合）

＜Windows 64bit 対応版＞

C:¥Program Files¥Common Files¥e-gov_app¥load_path¥default.dat

C:¥Program Files(x86)¥Common Files¥e-gov_app¥load_path¥default.dat

（上記は OS のインストール先が C ドライブの場合）

＜Mac OS 対応版＞

/private/etc/e-gov_app/load_path/default.dat

（Mac OS の場合、「/etc」は「/private/etc」のシンボリックリンクとなる）

ロード情報定義ファイル内のフォーマットは以下の通り。

項目=指定内容

- 項目と指定内容とは「=」でつなぎ、不要な空白等は含まない。
- 各項目について、内容は 1 行に記述し、「=」以降、改行文字までが有効な指定内容とする。

項目として設定する情報は以下の通り。

項目	指定内容
name	「会社名-識別情報」の形式。「JPKI_Appli-01」とする。
path	住基カード対応 PKCS#11 ライブラリ格納先の絶対パス名を設定。 ※住基カードは Windows 版のみ使用可能。
pathSign	個人番号カード署名対応 PKCS#11 ライブラリ格納先の絶対パス名を設定。
pathAuth	個人番号カード利用者証明対応 PKCS#11 ライブラリ格納先の絶対パス名を設定。
path64	住基カード対応 PKCS#11 ライブラリ(64bit)格納先の絶対パス名を設定。
pathSign64	個人番号カード署名対応 PKCS#11 ライブラリ(64bit)格納先の絶対パス名を設定。

項目	指定内容
pathAuth64	個人番号カード利用者証明対応 PKCS#11 ライブラリ(64bit)の格納先の絶対パス名を設定。

例)ロード情報定義ファイルの例

Windows 版の場合：

```
name=JPKI_Appli-01
path=[インストールフォルダ]¥JPKIPKCS11.dll
pathSign=[インストールフォルダ]¥JPKIPKCS11Sign.dll
pathAuth=[インストールフォルダ]¥JPKIPKCS11Auth.dll
path64=[インストールフォルダ]¥JPKIPKCS1164.dll
pathSign64=[インストールフォルダ]¥JPKIPKCS11Sign64.dll
pathAuth64=[インストールフォルダ]¥JPKIPKCS11Auth64.dll
```

MacOS 版の場合：

```
name=JPKI_Appli-01
pathSign=/usr/local/lib/JPKIPKCS11Sign.dylib
pathAuth=/usr/local/lib/JPKIPKCS11Auth.dylib
```

第２節　拡張 API の呼出方法

拡張 API である JPKIGetRemain 関数を利用する場合は、OS が提供するアドレス取得関数(Windows なら GetProcAddress()、Mac なら dlsym())を使用して関数アドレスを取得する必要がある。

住基カードは JPKIGetRemain 関数について提供対象外の為、関数アドレス取得を実施した場合エラーとなる。

禁・無断転載

公的個人認証サービス

利用者クライアントソフト API 仕様書
【カード AP ライブラリ PKCS#11 編】

第 4.6 版

（注意事項）

※利用者クライアントソフトの著作権は、総務省、地方公共団体情報システム機構が保有しており、国際著作権条約及び日本国の著作権関連法令によって保護されています。

※利用者クライアントソフトの利用に当たっては、次に掲げる行為を禁止します。

（1）利用者クライアントソフトを電子署名に係る地方公共団体情報システム機構の認証業務に関する法律において制限されている電子証明書の用途で利用すること。

（2） 利用者クライアントソフトに対し、総務省、地方公共団体情報システム機構に許可なく改造等を行うこと。

※総務省、地方公共団体情報システム機構は、利用者が利用者クライアントソフトを利用したことにより発生した利用者の損害及び利用者が第三者に与えた損害について、一切の責任を負いません。

※商標については次の通りです。

（1） Microsoft Windows および Internet Explorer は、米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。

（2） Macintosh、Mac、MacOS、OS X および Safari は、米国およびその他の国で登録されている Apple Inc.の登録商標です。

（3） Android は、Google Inc.の米国およびその他の国における登録商標です。

（4）その他、記載されている会社名、製品名等は、各社の登録商標または商標です。