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 CryptoAPI 編】

第１.1版

公的個人認証サービス　指定認証機関

財団法人　自治体衛星通信機構

変更履歴

版数	変更内容
1.0 版	新規作成
1.1 版	Windows XP SP2 対応に伴い表 1(2 頁)のプラットホームを追加

－ 目次 －

第１章　はじめに

利用者クライアント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、CryptoAPI の API 仕様について説明する。

第２章　ドキュメント体系

利用者クライアント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。

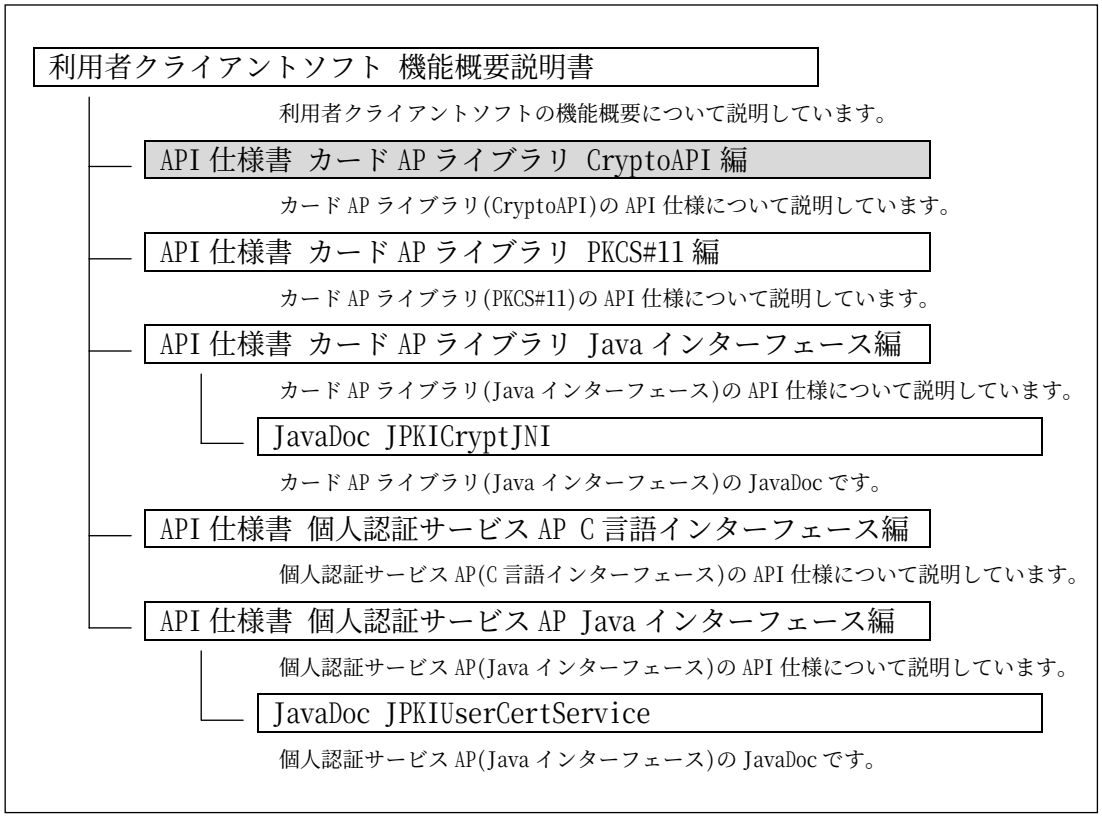

図 1 ドキュメント体系図

第３章　動作環境

カード AP ライブラリの動作環境は以下の通りとする。

表 1 動作環境

項目	条件
プラットフォーム	Windows98 Second Edition(※1) Windows Millennium Edition(※1) WindowsNT4.0 ServicePack6a(※1) Windows2000 ServicePack2 Windows2000 ServicePack3 Windows2000 ServicePack4 WindowsXP ServicePack1 WindowsXP ServicePack2
Web ブラウザ(※2)	Internet Explorer5.5 ServicePack2 Internet Explorer6 ServicePack1
IC カード	公的個人認証サービスカードアプリケーションを搭載し、公的個人認証サービスの電子証明書が格納されたICカードとする。
IC カードリーダライタ	以下の条件を満たす IC カードリーダライタとする。(「適合性検証済み IC カードリーダライター一覧」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 ・ USB や RS-232C 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 ・ 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 IC カードの利用のため、Microsoft Smart Card Base Components が必要になる。

※2 暗号機能等の利用のため、Internet Explorer5.5 ServicePack2 もしくは Internet Explorer6 ServicePack1 が必要になる。

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、利用者クライアントソフトのうち、下図の太枠に示すカード AP ライブラリ(CryptoAPI)の仕様をまとめる。

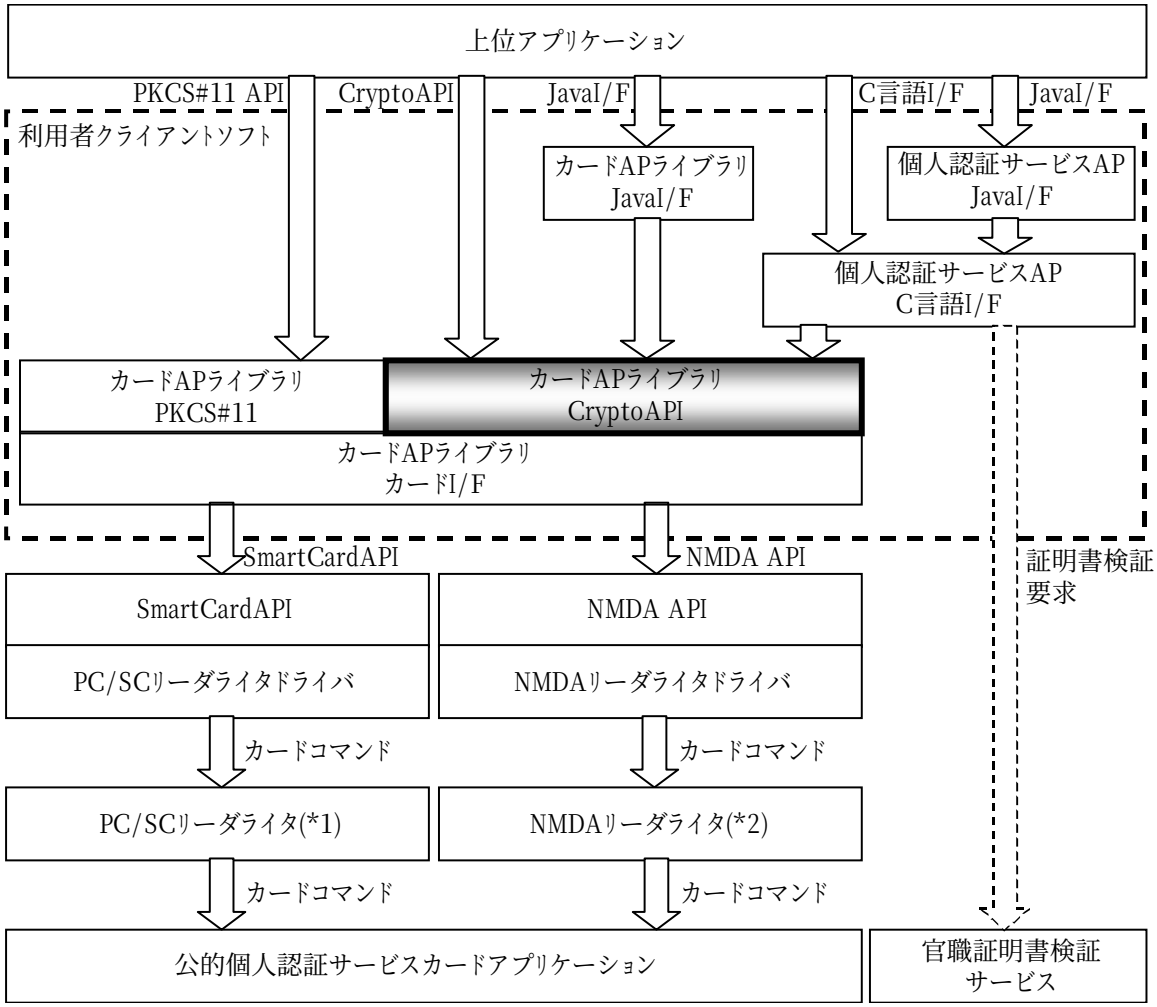

図 1 ソフトウェア構成図

*1 Personal Computer/Smart Cardの略。Microsoft社等のワーキンググループが推進する、Windows環境におけるICカード利用のための統一規格(PC/SC規格)に対応したICカードリーダライタのことを指す。

*2 New Media Development Associationの略。(財)ニューメディア開発協会「IT装備都市研究事業 リーダライタ共通インターフェース仕様書 1.1 版[平成 14 年 5 月 29 日]」に対応したICカードリーダライタのことを指す。

第２節　実現可能な機能の一覧

カード AP ライブラリ(CryptoAPI)で実現可能な機能の一覧を表 2に示す。

表 2 実現可能な機能の一覧

NO	機能	概要
1	利用者証明書取得	IC カードに格納された利用者証明書を取得する。
2	都道府県知事の自己署名証明書取得	IC カードに格納された都道府県知事の自己署名証明書を取得する。
3	署名生成(署名対象データを渡すパターン)	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。
4	署名生成(ハッシュ値を渡すパターン)	ハッシュ値に対して、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。
5	署名検証(検証対象データを渡すパターン)	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。
6	署名検証(ハッシュ値を渡すパターン)	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。
7	繰り返し署名生成(署名対象データを渡すパターン)	NO3 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。
8	繰り返し署名生成(ハッシュ値を渡すパターン)	NO4 の処理を繰り返し実行し、複数のハッシュ値に対する電子署名を生成する。
9	繰り返し署名検証(検証対象データを渡すパターン)	NO5 の処理を繰り返し実行し、複数の電子署名を検証する。
10	繰り返し署名検証(ハッシュ値を渡すパターン)	NO6 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

利用者クライアントソフト用 Cryptographic Service Provider(以下、CSP)がサポートする CryptoAPI の一覧を表 3に示す。なお、サポートしていない API を呼び出した場合には、戻り値が常に FALSE(失敗)となる。

表 3 サポート API 一覧

NO	CryptoAPI	概要
1	CryptAcquireContext	鍵コンテナのハンドルを生成する。
2	CryptReleaseContext	鍵コンテナのハンドルを開放する。
3	CryptGetProvParam	CSP のパラメータの値を取得する。
4	CryptDestroyKey	鍵の破棄を行う。
5	CryptGetKeyParam	鍵のパラメータの値を取得する。
6	CryptImportKey	外部から鍵を取り込む。
7	CryptGetUserKey	鍵コンテナ内の鍵ハンドルを取得する。
8	CryptCreateHash	ハッシュオブジェクトの生成を行う。
9	CryptDestroyHash	ハッシュオブジェクトの破棄を行う。
10	CryptSetHashParam	ハッシュオブジェクトのパラメータを設定する。
11	CryptGetHashParam	ハッシュオブジェクトのパラメータを取得する。
12	CryptHashData	ハッシュオブジェクトにデータを与えハッシュ値を計算する。
13	CryptSignHash	ハッシュ値に署名を行う。
14	CryptVerifySignature	署名を検証する。

第２節　サポート API 仕様詳細

各関数の戻り値は BOOL 型で、エラーが発生した場合は FALSE を返す。上位アプリケーションでエラー情報を取得する際は、GetLastError()関数をコールしてエラーコードを取得すること。なお、本仕様書に記述のないエラーコードが OS から返る場合があるが、それらのエラーコードについては MSDN を参照のこと。

（１） CryptAcquireContext

API 名	CryptAcquireContext		
概要	鍵コンテナのハンドルを生成する。		
関数インターフェース	CryptAcquireContext(HCRYPTPROV* phProv, LPCTSTR pszContainer, LPCTSTR pszProvider, DWORD dwProvType, DWORD dwFlags);		
戻り値	BOOL（TRUE:成功　FALSE:失敗）		
	型	I/O	内容
引数	HCRYPTPROV*	OUT	ハンドル格納場所のポインタ
	LPCTSTR	IN	NULL 文字で終了するコンテナ名称 NULL を指定すること。
	LPCTSTR	IN	NULL 文字で終了する CSP 名称 ”JPKI Crypto Service Provider” を指定すること。
	DWORD	IN	プロバイダタイプ PROV_RSA_FULL を指定すること。
	DWORD	IN	動作に関するパラメータ 0: 利用者証明書、利用者秘密鍵使用時に指定する。 CRYPT_VERIFYCONTEXT:都道府県知事の自己署名証明書取得、署名検証の際に指定する。
	値	内容	
エラーコード	NTE_BAD_FLAGS	dwFlags に不正な値が設定されている。	
	NTE_KEYSET_NOT_DEF	鍵コンテナが見つからない。	
	NTE_NO_MEMORY	メモリが不足している。	
	SCARD_W_CANCELLED_BY_USER	パスワード入力が利用者によってキャンセルされた。	

	SCARD_E_NOT_READY	R/W 接続不備、IC カード未挿入等のため IC カードに接続できない。
	SCARD_W_CHV_BLOCKED	利用者パスワードがロックしている。
備考	dwFlags に"0"を指定した場合、パスワード入力画面が表示される。	

（2） CryptReleaseContext

API 名	CryptReleaseContext		
概要	鍵コンテナのハンドルを開放する。		
関数インターフェース	CryptReleaseContext(HCRYPTPROV hProv, DWORD dwFlags);		
戻り値	BOOL（TRUE:成功　FALSE:失敗）		
	型	I/O	内容
引数	HCRYPTPROV	IN	CryptAcquireContext で取得したハンドル
	DWORD	IN	動作に関するパラメータ 0 を指定すること。
	値	内容	
エラーコード	NTE_BAD_UID	hProv に不正な値が設定されている。	

（3） CryptGetProvParam

API 名	CryptGetProvParam		
概要	CSP のパラメータの値を取得する。		
関数インターフェース	CryptGetProvParam(HCRYPTPROV hProv, DWORD dwParam, BYTE* pbData, DWORD* pdwDataLen, DWORD dwFlags);		
戻り値	BOOL（TRUE:成功　FALSE:失敗）		
	型	I/O	内容
引数	HCRYPTPROV	IN	CryptAcquireContext で取得したハンドル
	DWORD	IN	値を取得するためのパラメータ サポートする値を表 4に示す。

	BYTE*	OUT	値を格納するバッファのポインタ NULL を指定した場合は値の書き込みは行われず、pdwDataLen に値の格納に必要な長さが設定される。
	DWORD*	IN/OUT	値の長さを保持するバッファへのポインタ 関数呼び出し時には、pbData バッファに割り当てられたメモリサイズを指定する。関数終了時には、値の格納に必要な長さが設定される。
	DWORD	IN	動作に関するパラメータ 0 を設定すること。
	値	内容	
エラーコード	ERROR_MORE_DATA	pbData バッファが小さすぎる。	
	NTE_BAD_TYPE	dwParam に不正な値が設定されている。	
	NTE_BAD_UID	hProv に不正な値が設定されている。	

表 4 CryptGetProvParam の dwParam でサポートする値

#	値	内容
1	PP_IMPTYPE	CSP の実装形を DWORD 型で返す。 本 CSP は、CRYPT_IMPL_MIXED | CRYPT_IMPL_REMOVABLE を返す。
2	PP_NAME	CSP 名称を CHAR 型の NULL ターミネート文字列で返す。 本 CSP は、"JPKI Crypto Service Provider"を返す。
3	PP_VERSION	CSP のバージョンを DWORD 型で返す。 本 CSP は、1.0(0x00000100)を返す。
4	PP_PROVTYPE	CSP のプロバイダタイプを DWORD 型で返す。 本 CSP は、PROV_RSA_FULL を返す。
5	PP_JPKI_CA_CERTIFICATE	本 CSP は、IC カードに格納された都道府県知事の自己署名証明書(DER 形式)を返す。

（4） CryptDestroyKey

API 名	CryptDestroyKey
概要	鍵の破棄を行う。

関数インターフェース	CryptDestroyKey(HCRYPTKEY hKey);		
戻り値	BOOL（TRUE:成功　FALSE:失敗）		
	型	I/O	内容
引数	HCRYPTKEY	IN	破棄する鍵のハンドル
	値	内容	
エラーコード	NTE_BAD_KEY	hKey に不正な値が設定されている。	

（5） CryptGetKeyParam

API 名	CryptGetKeyParam		
概要	鍵のパラメータの値を取得する。 (IC カードに格納された利用者証明書(DER 形式)を返す。)		
関数インターフェース	CryptGetKeyParam(HCRYPTKEY hKey, DWORD dwParam, BYTE* pbData, DWORD* pdwDataLen, DWORD dwFlags);		
戻り値	BOOL（TRUE:成功　FALSE:失敗）		
	型	I/O	内容
引数	HCRYPTKEY	IN	値を取得する鍵のハンドル
	DWORD	IN	値を取得するパラメータ KP_CERTIFICATE: IC カードに格納された利用者証明書を返す。
	BYTE*	OUT	値を格納するバッファのポインタ NULL を指定した場合は値の書き込みは行われず、pdwDataLen に値の格納に必要な長さが設定される。
	DWORD*	IN/OUT	値の長さを保持するバッファへのポインタ 関数呼び出し時には、pbData バッファに割り当てられたメモリサイズを指定する。関数終了時には、値の格納に必要な長さが設定される。

	DWORD	IN	動作に関するパラメータ 0 を設定すること。
	値	内容	
エラーコード	ERROR_MORE_DATA	pbData バッファが小さすぎる。	
	NTE_BAD_KEY	hKey に不正な値が設定されている。	
	NTE_BAD_TYPE	dwParam に不正な値が設定されている。	

（6） CryptImportKey

API 名	CryptImportKey		
概要	外部から鍵を取り込む。		
関数インターフェース	CryptImportKey(HCRYPTPROV hProv, BYTE* pbData, DWORD dwDataLen, HCRYPTKEY hPubKey, DWORD dwFlags, HCRYPTKEY* phKey);		
戻り値	BOOL（TRUE:成功　FALSE:失敗）		
	型	I/O	内容
引数	HCRYPTPROV	IN	CryptAcquireContext で取得したハンドル
	BYTE*	IN	鍵データのバッファへのポインタ 公開鍵を取り込む際のデータ構造は構造体「第 3 節　（3）Public-key BLOBs」を参照のこと。
	DWORD	IN	鍵データのサイズ(バイト)
	HCRYPTKEY	IN	鍵データの各種パラメータ NULL を設定すること。
	DWORD	IN	動作に関するパラメータ 0 を設定すること。
	HCRYPTKEY*	OUT	取り込んだ鍵のハンドルをコピーするバッファのアドレス
	値	内容	
エラーコード	NTE_BAD_TYPE	サポートされていない BLOB タイプまたは不正な鍵をインポートしようとした。	
	NTE_BAD_UID	hProv に不正な値が設定されている。	

	NTE_BAD_VER	インポートしようとした鍵 BLOB のバージョンはサポートしていない。
備考	本関数では、暗号化されていない PUBLICKEYBLOB のみをサポートする。	

（7） CryptGetUserKey

API 名	CryptGetUserKey		
概要	鍵コンテナ内の鍵ハンドルを取得する。		
関数インターフェース	CryptGetUserKey(HCRYPTPROV hProv, DWORD dwKeySpec, HCRYPTKEY* phUserKey);		
戻り値	BOOL（TRUE:成功　FALSE:失敗）		
	型	I/O	内容
引数	HCRYPTPROV	IN	CryptAcquireContext で取得したハンドル
	DWORD	IN	鍵の種類 AT_KEYEXCHANGE: 鍵交換用鍵 AT_SIGNATURE: 署名用鍵 AT_SIGNATURE を設定すること。
	HCRYPTKEY*	OUT	鍵ハンドルをコピーするバッファのアドレス
	値	内容	
エラーコード	NTE_BAD_KEY	hKey に不正な値が設定されている。	
	NTE_BAD_UID	hProv に不正な値が設定されている。	
	NTE_NO_KEY	dwKeySpec に指定された種類の鍵が存在しない。	

（8） CryptCreateHash

API 名	CryptCreateHash
概要	ハッシュオブジェクトの生成を行う。
関数インターフェース	CryptCreateHash(HCRYPTPROV hProv, ALG_ID Algid, HCRYPTKEY hKey, DWORD dwFlags, HCRYPTHASH* phHash);

戻り値	BOOL（TRUE:成功　FALSE:失敗）		
	型	I/O	内容
引数	HCRYPTPROV	IN	CryptAcquireContext で取得したハンドル
	ALG_ID	IN	ハッシュアルゴリズム CALG_SHA: SHA1 アルゴリズム CALG_SHA1: SHA1 アルゴリズム CALG_SSL3_SHAMD5: SSL クライアント認証用アルゴリズム
	HCRYPTKEY	IN	キードハッシュの場合の鍵ハンドル 0 を設定すること。(本 CSP ではキードハッシュは未サポート)
	DWORD	IN	動作に関するパラメータ 0 を設定すること。
	HCRYPTHASH*	OUT	生成したハッシュオブジェクトのハンドルをコピーするバッファのポインタ
	値	内容	
エラーコード	NTE_BAD_ALGID	指定されたアルゴリズムはサポートしていない。	
	NTE_BAD_UID	hProv に不正な値が設定されている。	
	NTE_NO_MEMORY	メモリが不足している。	

（9） CryptDestroyHash

API 名	CryptDestroyHash		
概要	ハッシュオブジェクトの破棄を行う。		
関数インターフェース	CryptDestroyHash(HCRYPTHASH hHash);		
戻り値	BOOL（TRUE:成功　FALSE:失敗）		
	型	I/O	内容
引数	HCRYPTHASH	IN	破棄するハッシュオブジェクトのハンドル
	値	内容	
エラーコード	NTE_BAD_HASH	hHash に不正な値が設定されている。	

（１０） CryptSetHashParam

API 名	CryptSetHashParam		
概要	ハッシュオブジェクトのパラメータを設定する。		
関数インターフェース	CryptSetHashParam(HCRYPTHASH hHash, DWORD dwParam, BYTE* pbData, DWORD dwFlags);		
戻り値	BOOL（TRUE:成功　FALSE:失敗）		
	型	I/O	内容
引数	HCRYPTHASH	IN	パラメータを設定するハッシュオブジェクトのハンドル
	DWORD	IN	設定するパラメータ HP_HASHVAL: pbData バッファに格納された値をハッシュ値としてハッシュオブジェクトに設定する。
	BYTE*	IN	パラメータに設定するデータのポインタ
	DWORD	IN	動作に関するパラメータ 0 を設定すること。
	値	内容	
エラーコード	NTE_BAD_HASH	hHash に不正な値が設定されている。	
	NTE_BAD_TYPE	dwParam に不正な値が設定されている。	

（１１） CryptGetHashParam

API 名	CryptGetHashParam		
概要	ハッシュオブジェクトのパラメータを取得する。		
関数インターフェース	CryptGetHashParam(HCRYPTHASH hHash, DWORD dwParam, BYTE* pbData, DWORD* pdwDataLen, DWORD dwFlags);		
戻り値	BOOL（TRUE:成功　FALSE:失敗）		
	型	I/O	内容

引数	HCRYPTHASH	IN	パラメータを取得するハッシュオブジェクトのハンドル
	DWORD	IN	取得するパラメータ HP_ALGID: アルゴリズム ID を DWORD 型で返す。 HP_HASHSIZE: ハッシュサイズを DWORD 型で返す。 HP_HASHVAL: ハッシュ値を返す。
	BYTE*	OUT	値を格納するバッファのポインタ NULL を指定した場合は値の書き込みは行われず、pdwDataLen に値の格納に必要な長さが設定される。
	DWORD*	IN/OUT	値の長さを保持するバッファへのポインタ 関数呼び出し時には、pbData バッファに割り当てられたメモリサイズを指定する。関数終了時には、値の格納に必要な長さが設定される。
	DWORD	IN	動作に関するパラメータ 0 を設定すること。
	値	内容	
エラーコード	ERROR_MORE_DATA	pbData バッファが小さすぎる。	
	NTE_BAD_HASH	hHash に不正な値が設定されている。	
	NTE_BAD_TYPE	dwParam に不正な値が設定されている。	

（１２） CryptHashData

API 名	CryptHashData
概要	ハッシュオブジェクトにデータを与えハッシュ値を計算する。
関数インターフェース	CryptHashData(HCRYPTHASH hHash, BYTE* pbData, DWORD dwDataLen, DWORD dwFlags);
戻り値	BOOL（TRUE:成功　FALSE:失敗）

	型	I/O	内容
引数	HCRYPTHASH	IN	ハッシュオブジェクトのハンドル
	BYTE*	IN	ハッシュ値を計算するデータのポインタ
	DWORD	IN	ハッシュ値を計算するデータの長さ
	DWORD	IN	動作に関するパラメータ 0 を設定すること。
	値	内容	
エラーコード	NTE_BAD_ALGID	hHash で指定されたアルゴリズムはサポートしていない。	
	NTE_BAD_HASH	hHash に不正な値が設定されている。	
	NTE_BAD_HASH_STATE	ハッシュ計算は既に完了しているため、データを追加できない。	
	NTE_FAIL	予期しないエラーが発生した。	
	NTE_NO_MEMORY	メモリが不足している。	

（１３） CryptSignHash

API 名	CryptSignHash		
概要	ハッシュ値に署名を行う。		
関数インターフェース	CryptSignHash(HCRYPTHASH hHash, DWORD dwKeySpec, LPCTSTR sDescription, DWORD dwFlags, BYTE* pbSignature, DWORD* pdwSigLen);		
戻り値	BOOL（TRUE:成功　FALSE:失敗）		
	型	I/O	内容
引数	HCRYPTHASH	IN	署名を行うハッシュオブジェクトのハンドル
	DWORD	IN	鍵の種類 AT_KEYEXCHANGE: 鍵交換用鍵 AT_SIGNATURE: 署名用鍵 AT_SIGNATURE を設定すること。

	LPCTSTR	IN	ハッシュの概要についての NULL ターミネート文字列 NULL を設定すること。
	DWORD	IN	署名時のフラグ 0 を設定すること。
	BYTE*	OUT	署名データを格納するバッファのポインタ NULL を指定した場合は値の書き込みは行われず、pdwSigLen に値の格納に必要な長さが設定される。
	DWORD*	IN/OUT	署名データの長さを保持するバッファへのポインタ 関数呼び出し時には、pbSignature バッファに割り当てられたメモリサイズを指定する。関数終了時には、署名データの格納に必要な長さが設定される。
	値	内容	
エラーコード	ERROR_MORE_DATA	pbSignature バッファが小さすぎる。	
	NTE_BAD_ALGID	hHash で指定されたアルゴリズムはサポートしていない。	
	NTE_BAD_FLAGS	dwFlags に不正な値が設定されている。	
	NTE_BAD_HASH	hHash に不正な値が設定されている。	
	NTE_NO_KEY	dwKeySpec に指定した種類の鍵が存在しない。	
	NTE_NO_MEMORY	メモリが不足している。	

（１４） CryptVerifySignature

API 名	CryptVerifySignature
概要	署名を検証する。

関数インターフェース	CryptVerifySignature(HCRYPTHASH hHash, BYTE* pbSignature, DWORD dwSigLen, HCRYPTKEY hPubKey, LPCTSTR sDescription, DWORD dwFlags);		
戻り値	BOOL（TRUE:成功　FALSE:失敗）		
	型	I/O	内容
引数	HCRYPTHASH	IN	検証するハッシュオブジェクトのハンドル
	BYTE*	IN	署名データのバッファへのポインタ
	DWORD	IN	署名データの長さ
	HCRYPTKEY	IN	署名の検証に使用する公開鍵のハンドル
	LPCTSTR	IN	ハッシュの概要についての NULL ターミネート文字列 NULL を設定すること。
	DWORD	IN	署名検証時のフラグ 0 を設定すること。
	値	内容	
エラーコード	NTE_BAD_FLAGS	dwFlags に不正な値が設定されている。	
	NTE_BAD_HASH	hHash に不正な値が設定されている。	
	NTE_BAD_KEY	hPubKey に不正な値が設定されている。	
	NTE_BAD_SIGNATURE	署名の検証に失敗した。	
	NTE_BAD_UID	hProv に不正な値が設定されている。	
	NTE_NO_MEMORY	メモリが不足している。	

第３節　構造体仕様

（1） PUBLICKEYSTRUC

構造体名		PUBLICKEYSTRUC		
概要		公開鍵の BLOB ヘッダを格納する構造体。		
NO	変数名	型	値	備考
1	bType	BYTE	BLOB タイプ	PUBLICKEYBLOB を指定する。
2	bVersion	BYTE	BLOB バージョン	CUR_BLOB_VERSION を指定する。
3	reserved	WORD	未使用	-
4	aiKeyAlg	ALG_ID	アルゴリズム ID	CALG_RSA_KEYX または CALG_RSA_SIGN を指定する。

（2） RSAPUBKEY

構造体名		RSAPUBKEY		
概要		公開鍵の BLOB ヘッダを格納する構造体。		
NO	変数名	型	値	備考
1	magic	DWORD	鍵のタイプ	RSA1(0x31415352)を指定する。
2	bitlen	DWORD	鍵長	1024(鍵長)を指定する。
3	pubexp	DWORD	public exponent	public exponent を指定する。

（3） Public-key BLOBs

構造体名		なし(構造体としては定義されていない)		
概要		公開鍵 BLOB		
NO	変数名	型	値	備考
1	publickeystruc	PUBLICKEYSTRUC	PUBLICKEYSTRUC 構造体	PUBLICKEYSTRUC 構造体を指定する。
2	rsapubkey	RSAPUBKEY	RSAPUBKEY 構造体	RSAPUBKEY 構造体を指定する。
3	BYTE	DWORD	Modulus	Modulus を指定する。

第４節　コーリングシーケンス

「第４章　第２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。

（１） 利用者証明書取得処理

CryptAcquireContext	鍵コンテナ生成 phProv: ハンドル格納領域アドレス pszContainer: NULL pszProvider: “JPKI Crypto Service Provider” dwProvType: PROV_RSA_FULL dwFlags: 0

↓

CryptGetUserKey	利用者鍵ハンドル取得 hProv: CryptAcquireContext で取得したハンドル dwKeySpec: AT_SIGNATURE phUserKey: 利用者鍵ハンドル格納領域アドレス

↓

CryptGetKeyParam	利用者証明書サイズ取得 hKey: 利用者鍵ハンドル dwParam: KP_CERTIFICATE pbData: NULL pdwDataLen: 利用者証明書長格納領域アドレス dwFlags: 0

↓

CryptGetKeyParam	利用者証明書取得 hKey: 利用者鍵ハンドル dwParam: KP_CERTIFICATE pbData: 利用者証明書格納領域アドレス pdwDataLen: 利用者証明書長格納領域アドレス dwFlags: 0

↓

CryptDestroyKey	利用者鍵ハンドル破棄 hKey: 破棄する鍵ハンドル

↓

CryptReleaseContext	鍵コンテナ開放 hProv: CryptAcquireContext で取得したハンドル dwFlags: 0

（2） 都道府県知事の自己署名証明書取得処理

CryptAcquireContext	鍵コンテナ生成 phProv: ハンドル格納領域アドレス pszContainer: NULL pszProvider: “JPKI Crypto Service Provider” dwProvType: PROV_RSA_FULL dwFlags: CRYPT_VERIFYCONTEXT
↓	
CryptGetProvParam	都道府県知事の自己署名証明書サイズ取得 hProv: CryptAcquireContext で取得したハンドル dwParam: PP_JPKI_CA_CERTIFICATE pbData: NULL pdwDataLen: 都道府県知事の自己署名証明書長格納領域アドレス dwFlags: 0
↓	
CryptGetProvParam	都道府県知事の自己署名証明書取得 hProv: CryptAcquireContext で取得したハンドル dwParam: PP_JPKI_CA_CERTIFICATE pbData: 都道府県知事の自己署名証明書格納領域アドレス pdwDataLen: 都道府県知事の自己署名証明書長格納領域アドレス dwFlags: 0
↓	
CryptReleaseContext	鍵コンテナ開放 hProv: CryptAcquireContext で取得したハンドル dwFlags: 0

（3） 署名生成処理(署名対象データを渡すパターン)[*1]

CryptAcquireContext	鍵コンテナ生成 phProv: ハンドル格納領域アドレス pszContainer: NULL pszProvider: “JPKI Crypto Service Provider” dwProvType: PROV_RSA_FULL dwFlags: 0
↓	

[*1] (3)署名生成処理(署名対象データを渡すパターン)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、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。

CryptGetUserKey	利用者鍵ハンドル取得 hProv: CryptAcquireContext で取得したハンドル dwKeySpec: AT_SIGNATURE phUserKey: 利用者鍵ハンドル格納領域アドレス
↓	
CryptGetKeyParam	利用者証明書サイズ取得 hKey: 利用者鍵ハンドル dwParam: KP_CERTIFICATE pbData: NULL pdwDataLen: 利用者証明書長格納領域アドレス dwFlags: 0
↓	
CryptGetKeyParam	利用者証明書取得 hKey: 利用者鍵ハンドル dwParam: KP_CERTIFICATE pbData: 利用者証明書格納領域アドレス pdwDataLen: 利用者証明書長格納領域アドレス dwFlags: 0
↓	
CryptDestroyKey	利用者鍵ハンドル破棄 hKey: 利用者鍵ハンドル
↓	
CryptCreateHash	ハッシュオブジェクト生成 hProv: CryptAcquireContext で取得したハンドル ALG_ID: アルゴリズム ID hKey: 0 dwFlags: 0 phHash: ハッシュオブジェクトのハンドル格納領域アドレス
↓	
CryptHashData	ハッシュ値計算 hHash: ハッシュオブジェクトのハンドル pbData: 署名対象データ dwDataLen: 署名対象データ長 dwFlags: 0
↓	
CryptGetHashParam	ハッシュ値取得 hHash: ハッシュオブジェクトのハンドル

	dwParam: HP_HASHVAL pbData: NULL pdwDataLen: ハッシュデータ長格納領域アドレス dwFlags: 0

↓

CryptGetHashParam	ハッシュ値取得 hHash: ハッシュオブジェクトのハンドル dwParam: HP_HASHVAL pbData: ハッシュデータ格納領域アドレス pdwDataLen: ハッシュデータ長格納領域アドレス dwFlags: 0

↓

CryptSignHash	署名値長取得 hHash: 署名対象ハッシュオブジェクトのハンドル dwKeySpec: AT_SIGNATURE sDescription: NULL dwFlags: 0 pbSignature: NULL pdwSigLen: 署名データ長格納領域アドレス

↓

CryptSignHash	署名 hHash: 署名対象ハッシュオブジェクトのハンドル dwKeySpec: AT_SIGNATURE sDescription: NULL dwFlags: 0 pbSignature: 署名データ格納領域アドレス pdwSigLen: 署名データ長格納領域アドレス

↓

CryptDestroyHash	ハッシュオブジェクト破棄 hHash: 破棄するハッシュオブジェクトハンドル

↓

CryptReleaseContext	鍵コンテナ開放 hProv: CryptAcquireContext で取得したハンドル dwFlags: 0

（4） 署名生成処理(ハッシュ値を渡すパターン)[*2]

CryptAcquireContext	鍵コンテナ生成 phProv: ハンドル格納領域アドレス pszContainer: NULL pszProvider: “JPKI Crypto Service Provider” dwProvType: PROV_RSA_FULL dwFlags: 0
↓	
CryptGetUserKey	利用者鍵ハンドル取得 hProv: CryptAcquireContext で取得したハンドル dwKeySpec: AT_SIGNATURE phUserKey: 利用者鍵ハンドル格納領域アドレス
↓	
CryptGetKeyParam	利用者証明書サイズ取得 hKey: 利用者鍵ハンドル dwParam: KP_CERTIFICATE pbData: NULL pdwDataLen: 利用者証明書長格納領域アドレス dwFlags: 0
↓	
CryptGetKeyParam	利用者証明書取得 hKey: 利用者鍵ハンドル dwParam: KP_CERTIFICATE pbData: 利用者証明書格納領域アドレス pdwDataLen: 利用者証明書長格納領域アドレス dwFlags: 0
↓	
CryptDestroyKey	利用者鍵ハンドル破棄 hKey: 利用者鍵ハンドル
↓	
CryptCreateHash	ハッシュオブジェクト生成 hProv: CryptAcquireContext で取得したハンドル ALG_ID: アルゴリズム ID hKey: 0 dwFlags: 0 phHash: ハッシュオブジェクトのハンドル格納領域アドレス

[*2] (4)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。

↓

CryptSetHashParam	ハッシュ値設定 hHash: ハッシュオブジェクトのハンドル dwParam: HP_HASHVAL pbData: ハッシュ値の格納領域アドレス dwFlags: 0

↓

CryptSignHash	署名値長取得 hHash: 署名対象ハッシュオブジェクトのハンドル dwKeySpec: AT_SIGNATURE sDescription: NULL dwFlags: 0 pbSignature: NULL pdwSigLen: 署名データ長格納領域アドレス

↓

CryptSignHash	署名 hHash: 署名対象ハッシュオブジェクトのハンドル dwKeySpec: AT_SIGNATURE sDescription: NULL dwFlags: 0 pbSignature: 署名データ格納領域アドレス pdwSigLen: 署名データ長格納領域アドレス

↓

CryptDestroyHash	ハッシュオブジェクト破棄 hHash: 破棄するハッシュオブジェクトハンドル

↓

CryptReleaseContext	鍵コンテナ開放 hProv: CryptAcquireContext で取得したハンドル dwFlags: 0

（5） 署名検証処理(検証対象データを渡すパターン)

CryptAcquireContext	鍵コンテナ生成 phProv: ハンドル格納領域アドレス pszContainer: NULL pszProvider: “JPKI Crypto Service Provider” dwProvType: PROV_RSA_FULL dwFlags: CRYPT_VERIFYCONTEXT

↓

CertCreateCertificateContext (本 CSP の対象外。OS 標準機能を使用する。)	証明書コンテキスト生成 dwCertEncodingType: X509_ASN_ENCODING pbCertEncoded: X.509 DER 形式の証明書データ cbCertEncoded: 証明書長

↓

CryptImportPublicKeyInfo (CryptImportKey がこの関数の内部でコールされる)	公開鍵インポート hCryptProv: CryptAcquireContext で取得したハンドル dwCertEncodingType: X509_ASN_ENCODING pInfo: 公開鍵情報 CertCreateCertificateContext の戻り値を pcCert とした時、&pcCert->pCertInfo->SubjectPublicKeyInfo phKey: 公開鍵ハンドル格納領域アドレス

↓

CertFreeCertificateContext (本 CSP の対象外。OS 標準機能を使用する。)	証明書コンテキスト破棄 pCertContext: 破棄する証明書コンテキスト

↓

CryptCreateHash	ハッシュオブジェクト生成 hProv: CryptAcquireContext で取得したハンドル ALG_ID: アルゴリズム ID hKey: 0 dwFlags: 0 phHash: ハッシュオブジェクトのハンドル格納領域アドレス

↓

CryptHashData	ハッシュ値計算 hHash: ハッシュオブジェクトのハンドル pbdata: ハッシュ値計算対象データ dwDataLen: ハッシュ値計算対象データ長 dwFlags: 0

↓

CryptVerifySignature	署名検証 hHash: ハッシュオブジェクトのハンドル pbSignature: 署名データ格納領域のアドレス dwSigLen: 署名データ長 hPubKey: 公開鍵ハンドル sDescription: NULL dwFlags: 0

↓

CryptDestroyHash	ハッシュオブジェクト破棄 hHash: 破棄するハッシュオブジェクトハンドル

↓

CryptDestroyKey	公開鍵破棄 hKey: 破棄する公開鍵ハンドル

↓

CryptReleaseContext	鍵コンテナ開放 hProv: CryptAcquireContext で取得したハンドル dwFlags: 0

（6） 署名検証処理(ハッシュ値を渡すパターン)

CryptAcquireContext	鍵コンテナ生成 phProv: ハンドル格納領域アドレス pszContainer: NULL pszProvider: “JPKI Crypto Service Provider” dwProvType: PROV_RSA_FULL dwFlags: CRYPT_VERIFYCONTEXT

↓

CertCreateCertificateContext (本 CSP の対象外。OS 標準機能を使用する。)	証明書コンテキスト生成 dwCertEncodingType: X509_ASN_ENCODING pbCertEncoded: X.509 DER 形式の証明書データ cbCertEncoded: 証明書長

↓

CryptImportPublicKeyInfo	公開鍵インポート hCryptProv: CryptAcquireContext で取得したハンドル dwCertEncodingType: X509_ASN_ENCODING pInfo: 公開鍵情報 CertCreateCertificateContext の戻り値を pcCert とした時、&pcCert->pCertInfo->SubjectPublicKeyInfo phKey: 公開鍵ハンドル格納領域アドレス

↓

CertFreeCertificateContext (本 CSP の対象外。OS 標準機能を使用する。)	証明書コンテキスト破棄 pCertContext: 破棄する証明書コンテキスト

↓

CryptCreateHash	ハッシュオブジェクト生成

	hProv: CryptAcquireContext で取得したハンドル ALG_ID: アルゴリズム ID hKey: 0 dwFlags: 0 phHash: ハッシュオブジェクトのハンドル格納領域アドレス

↓

CryptSetHashParam	ハッシュ値設定 hHash: ハッシュオブジェクトのハンドル dwParam: HP_HASHVAL pbData: ハッシュ値の格納領域アドレス dwFlags: 0

↓

CryptVerifySignature	署名検証 hHash: ハッシュオブジェクトのハンドル pbSignature: 署名データ格納領域のアドレス dwSigLen: 署名データ長 hPubKey: 公開鍵ハンドル sDescription: NULL dwFlags: 0

↓

CryptDestroyHash	ハッシュオブジェクト破棄 hHash: 破棄するハッシュオブジェクトハンドル

↓

CryptDestroyKey	公開鍵破棄 hKey: 破棄する公開鍵ハンドル

↓

CryptReleaseContext	鍵コンテナ開放 hProv: CryptAcquireContext で取得したハンドル dwFlags: 0

（7） 繰り返し署名生成処理(署名対象データを渡すパターン)

「（3）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。

（8） 繰り返し署名生成処理(ハッシュ値を渡すパターン)

「（4）署名生成処理(ハッシュ値を渡すパターン)」の網掛け部分をハッシュ値の個数分だけ繰り返して呼び出す。

（9） 繰り返し署名検証処理(検証対象データを渡すパターン)

「（5）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。)

（１０） 繰り返し署名検証処理(ハッシュ値を渡すパターン)

「（6）署名検証処理(ハッシュ値を渡すパターン)」の網掛け部分をハッシュ値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　画面仕様

第１節　画面一覧

NO	機能名	画面名	機能概要
1	ログイン機能	パスワード入力画面	公的個人認証サービスのパスワード入力要求を行う。

第２節　画面仕様詳細

画面名	パスワード入力画面
概要	公的個人認証サービスのパスワード入力を要求し、入力されたパスワードで公的個人認証サービスカード AP にログインを行う。

画面レイアウト

画面項目説明

NO	項目名	概要
①	パスワード	公的個人認証サービスのパスワードを入力する。(伏字(*)で表示する。)
②	OK	入力したパスワードで IC カードにログインを行う。 ログインできない場合は以下のメッセージを表示する。 ・ パスワードが誤っている場合 →「パスワードが違います。」 ・ ロックしている場合 →「パスワードの確認に失敗しました。」 ・ ログイン時に予期しないエラーが発生した場合 →「予期しないエラーが発生しました。(エラーコード:XXXXX)」
③	キャンセル	パスワード入力を行わずにダイアログを閉じる。
④	×	パスワード入力を行わずにダイアログを閉じる。

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 CryptoAPI 編】

第１．１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。

※総務省は、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。

※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申請・届出等の行政手続等以外の目的で利用すること。

（2） 利用者クライアントソフトに対し、総務省に許可なく改造等を行うこと。