商業登記リモート署名ドライバソフト
API 仕様書
【CryptoAPI 編】

1.0 版

令和７年１０月
法務省民事局商事課

改訂履歴

版数	改訂日付	改訂内容
1.0 版	2025 年 10 月 24 日	・新規作成

目次

第１章　はじめに

商業登記リモート署名ドライバソフト（以降、署名ドライバとする）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、本書では署名ドライバのうち、CryptoAPIのAPI仕様について説明する。

第２章　動作環境

署名ドライバ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	Windows 11 Home/Pro
Web ブラウザ	Microsoft Edge

第３章　必要となるヘッダーファイル

署名ドライバで必要となるヘッダーファイルは以下の通りとする。

表１ ヘッダーファイル

ヘッダーファイル名	概要
CRPKICSP_ex.h	CSP 外部公開ヘッダ

第４章　機能仕様

１．ソフトウェア構成図

本仕様書では、署名ドライバのうち、下図の太枠に示す署名ライブラリ(CryptoAPI/CSP)の仕様をまとめる。

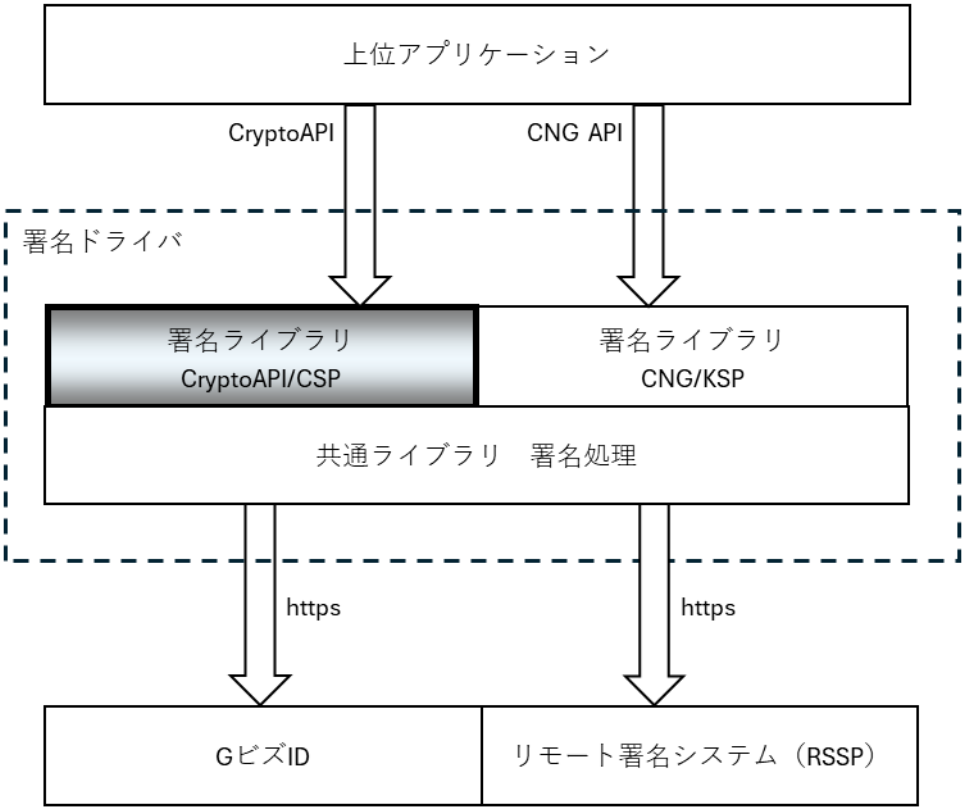


図1 ソフトウェア構成図

２．実現可能な機能の一覧

署名ライブラリ(CryptoAPI/CSP)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	証明書ストアから商業登記電子証明書（管理ツール*1 から設定したもの）を取得する。
2	認証局の自己署名証明書取得	商業登記電子証明書の登記官証明書（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、商業登記電子証明書に紐づく RSSP 上の秘密鍵を使用して署名値を生成する。
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､商業登記電子証明書に紐づく RSSP 上の秘密鍵を使用して署名値を生成する。
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、署名値、公開鍵を使用して署名値を検証する。
6	署名検証 （ハッシュ値を渡すパターン）	ハッシュ値､署名値､公開鍵を使用して署名値を検証する。
7	繰り返し署名生成 （署名対象データを渡すパターン）	NO 3 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署名値を生成する。
8	繰り返し署名生成 （ハッシュ値を渡すパターン）	NO 4 の処理を繰り返し実行し、複数のハッシュ値に対する署名値を生成する。
9	繰り返し署名検証 （検証対象データを渡すパターン）	NO 5 の処理を繰り返し実行し、複数の署名値を検証する。
10	繰り返し署名検証 （ハッシュ値を渡すパターン）	NO 6 の処理を繰り返し実行し、複数の署名値を検証する。

*1 署名アプリケーションから電子署名を行う際に必要な電子署名用の鍵および証明書を設定するためのツール。

第５章　API仕様

１．サポート API 一覧

署名ドライバ用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	キーコンテナのハンドルを生成する。
2	CryptReleaseContext	キーコンテナのハンドルを解放する。
3	CryptGetProvParam	CSP のパラメータ値を取得する。
4	CryptDestroyKey	使用済みの鍵を破棄し、メモリを解放する。
5	CryptGetKeyParam	鍵のパラメータを取得する。
6	CryptImportKey	外部から鍵を取込む。
7	CryptGetUserKey	キーコンテナ内のキー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付与し、 ハッシュ値の計算を行う。
13	CryptSignHash	ハッシュ値に署名を行う。
14	CryptVerifySignature	署名の検証を行う。

２．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 ”CRPKI Crypto Service Provider for Remote Sign”を指定すること。
	DWORD	IN	プロバイダタイプ PROV_RSA_AES を指定すること。
	DWORD	IN	動作に関するパラメータ 0、CRYPT_VERIFYCONTEXT のいずれかを指定すること。 **詳細は備考(A)を参照。**
	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。	
備考	(A) dwFlags に指定する動作に関するパラメータは以下の値を設定する。 **「0」:利用者証明書、利用者秘密鍵使用時の場合、指定する。** **「CRYPT_VERIFYCONTEXT」:認証局の自己署名証明書取得、署名検証の場合、指定する。**		

（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 4-2 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 5-2 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、CryptAcquireContext 実行時に 引数 pszProvider で指定された CSP 名称を返す。
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、CryptAcquireContext 実行時に引数 dwProvType で定されたプロバイダタイプを返す。
5	PP_CRPKI_CA_CERTIFICATE （CRPKICSP_ex.h に定義された定数値）	本 CSP は、認証局の自己署名証明書(DER 形式)を返す。 定数値は「1000」で定義する。

（4） CryptDestroyKey

API名	CryptDestroyKey		
概要	鍵ハンドルの破棄を行う。		
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。 （リモート署名システム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:鍵に関連付けられた証明書を返す。
	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_SHA_256: SHA256 アルゴリズム CALG_SHA_384: SHA384 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 CALG_SHA_256 または CALG_SHA_384 を設定すること。
	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。	
	ERR_TOKEN_GET_F	セッション情報（アクセストークン、リフレッシュトーク	

	AILED	ン）の取得に失敗した。
	ERR_TOKEN_REFRESH_FAILED	セッション情報の再取得（リフレッシュトークンを使用したアクセストークンの再取得）に失敗した。
	ERR_AUTH_FAILED	G ビズ ID の認証エラーが発生した。
	ERR_GBIZID_COMM_FAILED	G ビズ ID との通信に失敗した。
	ERR_SAD_EXTEND_FAILED	SAD の延長に失敗した。
	ERR_AUTHCODE_GEN_FAILED	認可コードの生成に失敗した。
	ERR_AUTH_REQUEST_FAILED	認可の要求に失敗した。
	ERR_AUTH_REQUEST_CHECK_FAILED	認可の要求に対するチェックが失敗した。
	ERR_SIGN_VALUE_GET_FAILED	署名値の取得に失敗した。
	ERR_SIGN_LIST_RETURN_FAILED	取得した署名値のリスト返却に失敗した。
	ERR_RSSP_COMM_FAILED	リモート署名システムとの通信に失敗した。
備考	本 API 実行時、「第６章　画面仕様　１．画面一覧」に記載の画面が表示される。	

（本ドキュメント中の「ERR_～」から始まるエラーコードの表記は、今後定義予定の独自エラーコードを示します。正式なエラーコードは、次版以降のドキュメントで提供します。）

（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。	

３．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：RSA 鍵交換 CALG_RSA_SIGN：RSA 署名用 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	2048(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。

４．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。

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。

（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: “CRPKI Crypto Service Provider for Remote Sign” dwProvType: PROV_RSA_AES 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

↓

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

（2）認証局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: “CRPKI Crypto Service Provider for Remote Sign” dwProvType: PROV_RSA_AES dwFlags: CRYPT_VERIFYCONTEXT

↓

CryptGetProvParam	認証局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_CRPKI_CA_CERTIFICATE pbData: NULL pdwDataLen: 認証局の自己署名証明書長格納領域アドレス dwFlags: 0

↓

CryptGetProvParam	認証局の自己署名証明書取得 hProv: CryptAcquireContext で取得したハンドル dwParam: PP_CRPKI_CA_CERTIFICATE pbData: 認証局の自己署名証明書格納領域アドレス pdwDataLen: 認証局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: “CRPKI Crypto Service Provider for Remote Sign” dwProvType: PROV_RSA_AES 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

↓

*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、
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。

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: CALG_SHA_256 または CALG_SHA_384 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	ハッシュデータ長取得 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 （※本 API 実行時、「第６章　画面仕様　１．画面一覧」に記載

	の画面が表示される。）
	↓
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: “CRPKI Crypto Service Provider for Remote Sign” dwProvType: PROV_RSA_AES 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

↓

*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、
署名対象のハッシュ値に対する署名値を取得することができる。

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: CALG_SHA_256 または CALG_SHA_384 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

↓

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 （※本 API 実行時、「第６章　画面仕様　１．画面一覧」に記載の画面が表示される。）

↓

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: “CRPKI Crypto Service Provider for Remote Sign” dwProvType: PROV_RSA_AES 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。	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

OS 標準機能を使用する。)	

↓

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: CALG_SHA_256 または CALG_SHA_384 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	利用者鍵ハンドル破棄 hKey: 利用者鍵ハンドル

↓

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: “CRPKI Crypto Service Provider for Remote Sign” dwProvType: PROV_RSA_AES 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: CALG_SHA_256 または CALG_SHA_384 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	利用者鍵ハンドル破棄 hKey: 利用者鍵ハンドル

↓

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。

※（７)、（８）の繰り返し署名生成処理の回数の上限は 10 回とする。

（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

１．画面一覧

NO	機能名	画面名	機能概要
1	ログイン機能	G ビズ ID ログイン	G ビズ ID ログイン画面を表示する。（外部サイト）
2	署名認可	鍵パスワード入力画面	署名認可時に必要な鍵パスワードを入力する。
3	署名認可	認可コード表示画面	G ビズ ID の端末認可に入力する認可コードを画面に表示する。

２．画面仕様詳細

（１）Ｇビズ ID ログイン

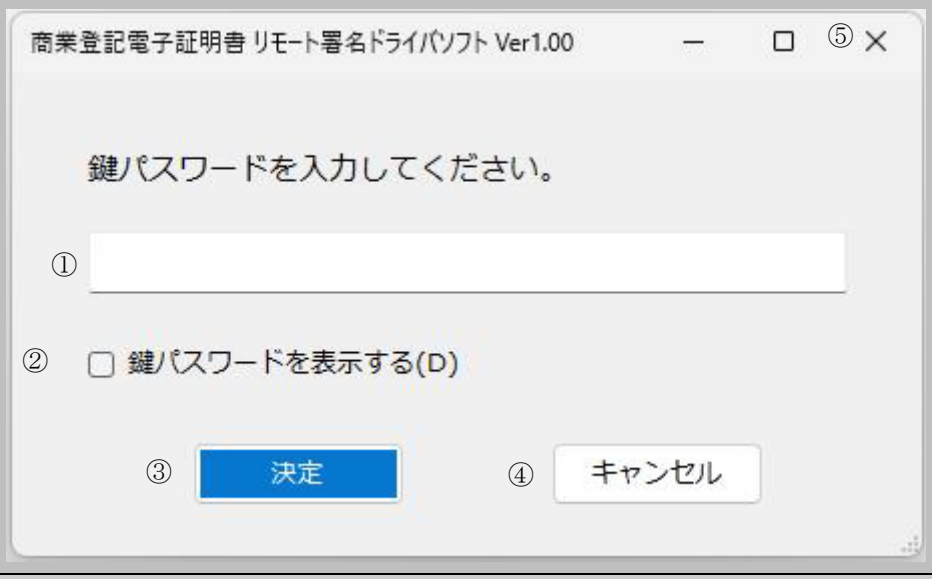
画面名	G ビズ ID ログイン（外部サイト）
概要	G ビズ ID のログインが必要な場合、ブラウザが起動されて G ビズ ID ログイン画面が表示され、G ビズ ID ログイン認証を行う。

画面レイアウト

（2） 鍵パスワード入力画面

画面名	鍵パスワード入力画面
概要	署名認可時に必要な鍵パスワードを入力する。

画面レイアウト



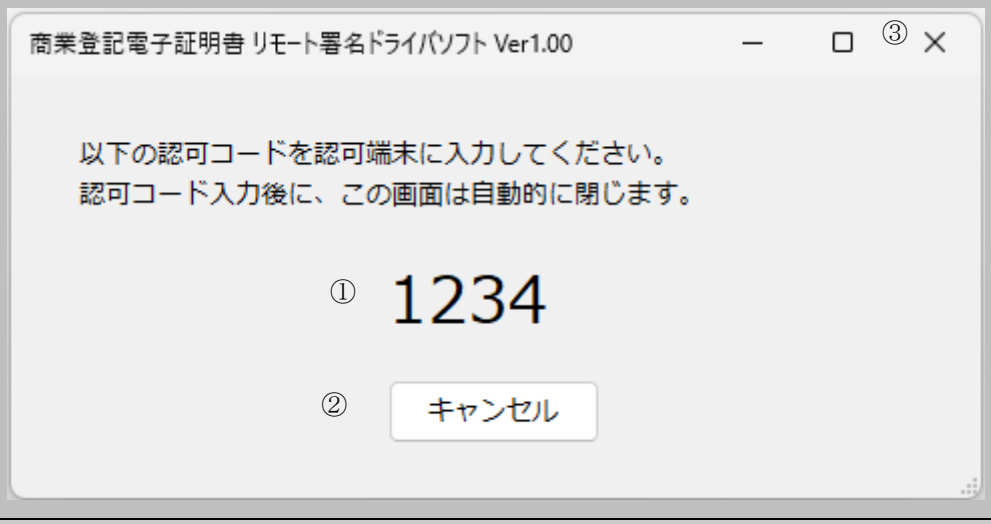
画面項目説明

NO	項目名	概要
①	鍵パスワード入力欄	鍵パスワードを入力する。（伏字(*)で表示する。）
②	鍵パスワード表示切替	入力した鍵パスワードの表示/非表示を切替える。
③	決定	リモート署名の認可要求を行う。 ３回までリトライ可能。３回目失敗時はエラーメッセージを表示して署名処理を終了する。
④	キャンセル	画面を閉じて鍵パスワード入力処理を終了する。
⑤	×	画面を閉じて鍵パスワード入力処理を終了する。

（3） 認可コード表示画面

画面名	認可コード表示画面
概要	G ビズ ID の端末認可に入力する認可コードを画面に表示する。

画面レイアウト



画面項目説明

NO	項目名	概要
①	認可コード	認可コードを表示する。
②	キャンセル	画面を閉じて認可コード表示処理を終了する。
③	×	画面を閉じて認可コード表示処理を終了する。

禁・無断転載

商業登記リモート署名ドライバソフト API 仕様書

【CryptoAPI 編】

第１.０版

（注意事項）

※　署名ドライバの著作権は、法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。

※　法務省は、利用者が署名ドライバ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。

※　署名ドライバの利用に当たっては、次に掲げる行為を禁止します。

（1）　署名ドライバに対し、**法務省**に許可なく改造等を行うこと。

※　商標については次の通りです。

(1)　Microsoft Windows 及び Microsoft Edge 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)　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。