ラング・エッジ クライアント署名

LE:Client:Sign V2.2 マニュアル

２０２６年７月１日版

□ 目次

□ 目次 ……1
○ 履歴 ……4
１．概要……6
１．１．クライアント署名 LE:ClientSign 概要 ……6
１．２．ブラウザ（ActiveX プラグインと署名コマンド）の選択 ……8
１．３．CAPI（CryptoAPI）と PKCS#11 の選択 ……9
１．４．署名サーバ連携 ……10
１．５．フォルダ構成 ……11
１．６．利用外部ライブラリとライセンス情報 ……12
１．７．HTTP 通信の仕様変更（V2.2） ……13
１．８．HTTP 通信のプロキシ設定 ……14
２．クライアント側実装：ActiveX プラグイン（LE:Client:Sign） ……17
２．１．ActiveX プラグインモジュール……17
２．２．ActiveX プラグイン 利用シーケンス ……18
２．２．１．一括 API 利用シーケンス ……19
２．２．２．個別 API 利用シーケンス ……19
２．３．ActiveX 利用 API……20
２．３．１．init：初期化 ……22
２．３．２．cert：証明書取得……24
２．３．３．info：証明書情報取得……25
２．３．４．sign：署名値計算……26
２．３．５．xadd：署名 XAdES-BES への署名対象追加……27
２．３．６．xades：署名 XAdES-BES 生成……28
２．３．７．term：解放（終了処理） ……29
２．３．８．errorGet：エラー値取得 ……29
２．３．９．errorApi：エラーAPI 名取得 ……30
２．３．１０．errorClear：エラー値クリア……30
２．３．１１．httpSet：HTTP 通信用設定……31
２．３．１２．httpSend：HTTP 通信実行 ……32
２．３．１３．version：バージョン情報取得 ……33
２．３．１４．status：IC カード状態取得（PKCS#11 のみ） ……33
２．３．１５．xmlGet：XML 値取得 ……33
２．３．１６．exec：一括実行 ……34
２．４．ActiveX プラグイン実装例……36
２．４．１．LeCSignCapiA.cab / CAPI 版 個別 API の利用……36
２．４．２．LeCSignCapiA.cab / CAPI 版 一括 API の利用……40
２．４．３．LeCSignP11A.cab / PKCS#11 版 個別 API の利用 ……43
２．４．４．LeCSignP11A.cab / PKCS#11 版 一括 API の利用 ……47

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．４．５．XAdES と証明書情報の利用（V2.1 新機能）……50
３．クライアント側実装：署名コマンド（LE:Client:Sign）……56
３．１．署名コマンドモジュール……56
３．１．１．署名コマンドインストーラ……57
３．１．２．署名コマンドのインストール手順……58
３．１．３．署名コマンドのアンインストール手順……61
３．２．署名コマンド利用シーケンス……63
３．２．１．一括 API 利用シーケンス……65
３．２．２．個別 API 利用シーケンス……65
３．２．３．署名コマンド実行の判定……66
３．３．署名コマンド利用 API……68
３．３．１．constructor：生成/初期化……69
３．３．２．connect：署名コマンド実行と接続……71
３．３．３．check：署名コマンドインストールチェック……71
３．３．４．cert：証明書取得……72
３．３．５．info：証明書情報取得……73
３．３．６．sign：署名値計算……74
３．３．７．xadd：署名 XAdES-BES への署名対象追加……75
３．３．８．xades：署名 XAdES-BES 生成……76
３．３．９．terminate：署名コマンド終了……77
３．３．１０．errorGet：エラー値取得……77
３．３．１１．status：IC カード状態取得（PKCS#11 のみ）……77
２．３．１２．version：バージョン情報取得……77
３．３．１３．http：HTTP/HTTPS 通信……78
３．３．１４．pxml：独自 XML 応答の解析……78
３．３．１５．exec：一括実行……79
３．３．１６．LeCSignCmdWS クラス エラー値……80
３．４．署名コマンド利用実装例……81
３．４．１．CAPI（CryptoAPI）利用……81
３．４．２．PKCS#11 利用……83
３．４．３．JavaScript の async/await 補足……84
３．４．４．XAdES と証明書情報の利用（V2.1 新機能）……86
３．５．商業登記リモート署名……92
４．サーバ側実装（LE:XAdES:Lib/LE:PAdES:LIb）……93
４．１．独自通信 XML プロトコル（LE:XAdES:Lib/LE:PAdES:Lib 共通）……93
４．１．１．証明書 XML（LCS_CERT：クライアント→サーバ：step 1）……94
４．１．２．ハッシュ値 XML（LCS_HASH：サーバ→クライアント：step 2）……94
４．１．３．署名値 XML（LCS_SIGN：クライアント→サーバ：step 3）……95
４．１．４．結果 XML（LCS_RSLT：サーバ→クライアント：step 4）……95
４．２．XML 長期署名ライブラリ LE:XAdES:Lib……96

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

４．２．１．XAdES サーバ試験コマンド LcsXAdES.exe ……96
４．３．PDF 長期署名ライブラリ LE:PAdES:Lib …… 100
４．３．１．PAdES サーバ試験コマンド LcsPAdES.exe …… 100
４．４．サーバ試験コマンド LcsXAdES/LcsPAdES（オプション） …… 102
付録１．エラーコード一覧 …… 104
付録１．１．LCS_ERR：基本エラーコード …… 104
付録１．２．LCSX_ERR：XAdES エラーコード …… 105
付録２．証明書情報一覧（info() が返す JSON/XML 情報） …… 107
付録３．XAdES-BES 仕様（xades() が返す XML 情報） ……112

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

○ 履歴

バージョン	日付	主な修正内容
Ver 1.00.B1	2017/04/10	LE:Client:Sign 仕様書のベータ 1 版
Ver 1.00.RC1	2017/05/01	C++の API は利用者では直接使わないので削除 xmlElmt/xmlAtrb の API を追加、httpSet/httpSend の引数を変更 exec の API と param によるオプション指定の追加 その他利用例等のブラッシュアップ
Ver 1.00.RC2	2017/06/30	3.2.1. PAdES サーバ試験コマンド LcsPAdES.exe の更新 PAdES の API の更新による修正
Ver 2.00.B1	2020/05/07	LE:Client:Sign V2 仕様書のベータ 1 版 ・Java Applet の廃止 ・署名コマンド（カスタム URL スキーム）のサポート
Ver 2.00.B2	2020/06/29	LE:Client:Sign V2 仕様書のベータ 2 版 ・WebSocket の同期 API による署名コマンド利用 API の見直し ➢ WebScoket 利用の為の connect/terminate の API を追加。 ➢ 個別実行 API の cert/sign を追加、exec も引数を見直し。 ・署名コマンドのシーケンスが個別と一括の 2 つとなった。 ・エラーコードの一部修正
Ver 2.00.B2a	2020/07/03	・署名コマンド API の説明更新（API に変更なし）。 ・署名コマンドのエラーコードの追加
Ver 2.00.RC1	2020/08/06	署名コマンドインストーラの対応 ・3.1.1. 署名コマンドインストーラ の追加 ・3.1.2. 署名コマンドのインストール手順 ・3.1.3. 署名コマンドのアンインストール手順 ・3.3.3. check の追加（以後章番号は繰り上げ） ・3.3.9. version の追加（以後章番号は繰り上げ）
Ver 2.00.R1	2020/08/18	・3.2.3. 署名コマンド実行の判定を更新 ・3.3.2. connect の引数追加 ・3.3.3. check の備考を追加 ・3.3.12. exec の記述更新 ・3.3.13. LeCSignCmdWS クラス エラー値 の更新
Ver 2.00.R2	2020/08/21	・3.2.3. 署名コマンド実行の判定を更新 ・3.3.2. connect の備考を追加
Ver 2.00.R2a	2020/10/29	・3.2.3. 署名コマンド実行の判定を更新（新仕様への対応の為） ・3.3.2. connect のインストール済みチェックの仕様変更 ・3.3.3. check のインストール済みチェックの仕様変更 ・3.4.3. JavaScript の async/await 補足 の追加
Ver 2.10.B1	2020/11/26	本ドキュメント名を仕様書からマニュアルに変更 以下の Ver2.1 新機能の仕様を追加

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

		・2.3.3. / 3.3.5. 新 API の info を追加（以下章番号はずれる） ・2.3.5. / 3.3.7. 新 API の xadd を追加（以下章番号はずれる） ・2.3.6. / 3.3.8. 新 API の xades を追加（以下章番号はずれる） ・付録 2. 証明書情報一覧 を追加
Ver 2.10.R1	2021/01/15	・2.3.3. / 3.3.4. 新 API の info を更新 ・2.3.5. / 3.3.7. 新 API の xadd を更新 ・2.3.6. / 3.3.8. 新 API の xades を更新 ・2.4.5. / 3.4.4. V2.1 機能の利用例を追加 ・付録 1.2. LCSX_ERR：XAdES エラーコード を追加 ・付録 3. XAdES-BES 仕様 を追加
Ver 2.10.R2	2022/09/01	・3.3.1. constructor に新フラグ LCSI_CAPI_CHKEKU を追加 ・3.3.4. cert に証明書選択画面他を追加 ・4.1. 独自通信 XML プロトコル を更新
Ver 2.2.R1	2026/01/28	・1.7. HTTP 通信の仕様変更(V2.2) を追加 ・1.8. HTTP 通信のプロキシ設定 を追加 ・3.3.1. constructor に新フラグ LCSI_SET_WININET と補足を追加
Ver 2.2.R2	2026/07/01	・3.5. 商業登記リモート署名 を追加

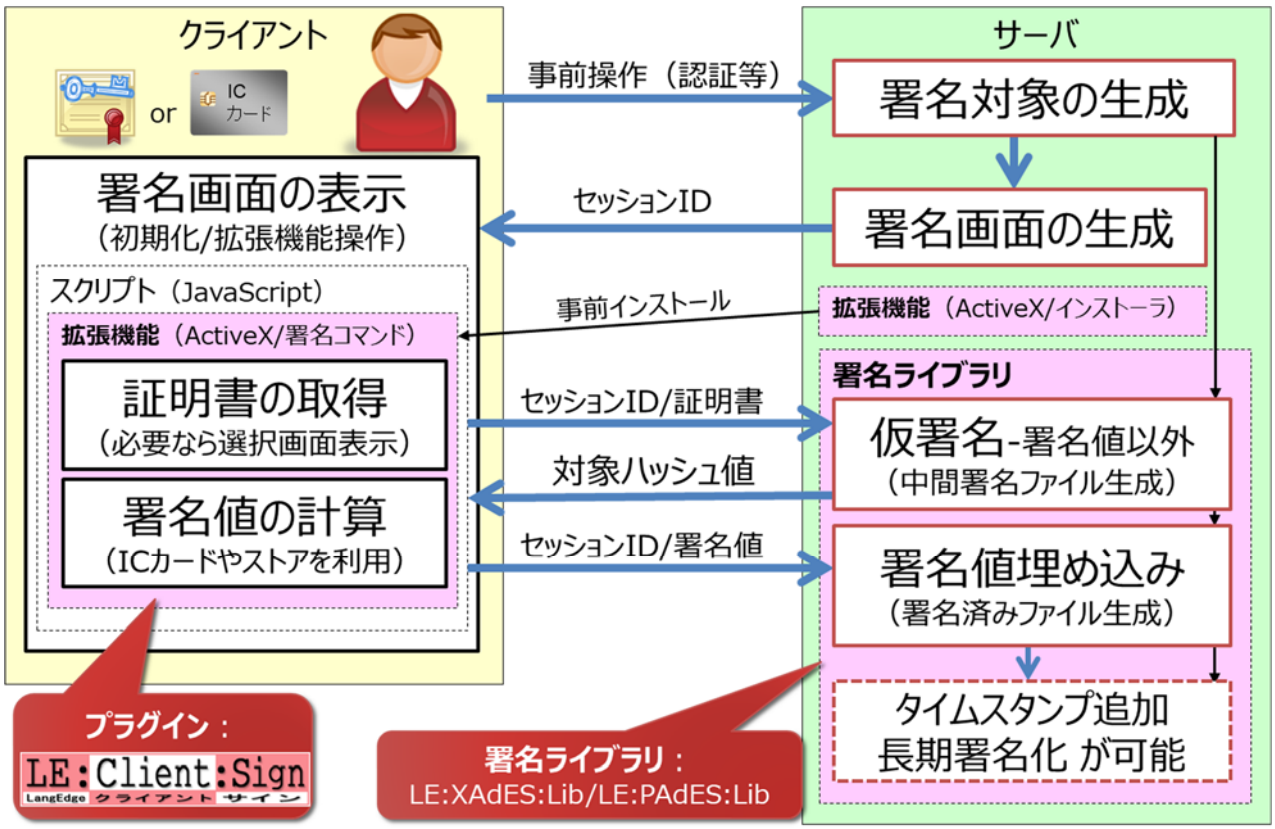
- 本製品マニュアルに記載の会社名、製品名は、各社の商標または登録商標です。
- 本製品マニュアルに記載の内容の一部または全部を無断で複写・転載することを禁じます。
- 本製品マニュアルに記載の内容及び製品の仕様等は予告なく変更される場合があります。

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

１．概要

１．１．クライアント署名 LE:ClientSign 概要

ラング・エッジのクライアント署名（LE:Client:Sign）は、サーバ側にあるラング・エッジの長期署名ライブラリ（LE:PAdES:Lib や LE:XAdES:Lib）と連携し、クライアント側にある秘密鍵（IC カードや証明書ストア）を利用して署名ファイルを生成する仕組みを提供する。クライアント側として ActiveX コンポーネント（V1 からサポート）と、カスタム URL スキームを使った署名コマンド（V2 新機能）の 2 種類を Windows 環境においてサポートしている。MacOS やスマホ等への対応も技術的に可能とはなっているが現在は提供していない。必要な場合には別途開発が必要となる。



ラング・エッジ クライアント署名概要図

※ PDF 長期署名ライブラリ LE:PAdES:Lib では既に ActiveX を利用した独自クライアント署名をサポートしているが、LE:Client:Sign とは互換性が無いので注意。LE:Client:Sign の方が高機能かつ汎用性がある仕様となっている。

※ LE:PAdES-Basic:Lib でも LE:Client:Sign の利用は可能だが別途ライセンスが必要となるので注意が必要。

※ Ver2.1 より LE:Client:Sign 単体で XAdES-BES の生成（ローカル署名）が可能となった。

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

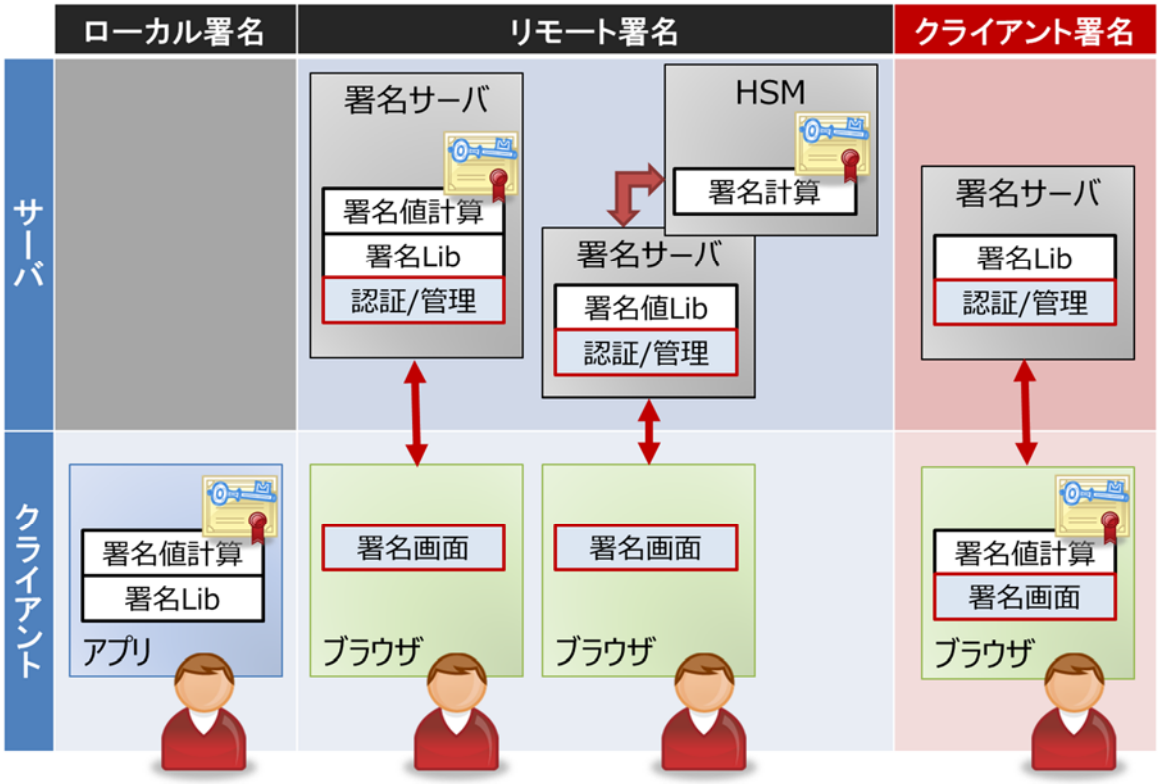
参考：電子署名の実現方法の分類

電子署名の実現方式は大きく分けると「ローカル署名」「リモート署名」「クライアント署名」の 3 つに分類できる。このうちクライアント署名はラング・エッジの造語であり一般用語ではない。これは署名処理と秘密鍵の署名値の計算をサーバ・クライアントのどこで行うかの違うとなる。ラング・エッジの長期署名ライブラリでは全ての方法に対応している。

1. **ローカル署名（署名アプリ等）**
   - ➢ クライアント端末上で署名処理**全て**を行う
     - ✓ タイムスタンプ取得や検証時にはネット接続が必要

2. **リモート署名（クラウド署名）**
   - ➢ サーバ上で署名処理**全て**を行う
     - ✓ 利用者の認証が必須であり近年注目を集める方式
     - ※ 参考：JT2A リモート署名ガイドライン（JNSA サイト）
       https://www.jnsa.org/result/jt2a/data/RemoteSignatureGguide_All.pdf

3. **クライアント署名（サーバ連携）**
   - ➢ サーバとクライアントが**連携**して署名を行う
     - ✓ サーバ部は署名ファイルの生成とハッシュ値計算
     - ✓ クライアント部はハッシュ値から署名値の生成のみ



Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

１．２．ブラウザ（ActiveX プラグインと署名コマンド）の選択

クライアント署名 V2 には、ActiveX プラグイン（コンポーネント）と署名コマンド（カスタム URL スキーム）の 2 種類があり、それぞれ利用可能となるブラウザが異なる。利用者にどちらを利用させるかを決める必要がある。

ActiveX プラグインは IE11 のみで動作する。署名コマンドは OS 機能のカスタム URL スキームを利用して実行する為にほぼ全てのブラウザで利用することが可能だがブラウザ依存もあるので現在では Chrome と Chromium ベースの Edge（新 Edge）のみ動作保証をしている。

どちらも JavaScript から実行するが、署名コマンドの利用には JavaScript ライブラリ（LeCSignCmdWS.js）の API を提供する。

	Active X プラグイン	署名コマンド（カスタム URL）
クライアント OS 対応	Windows 10（Windows 8.1）	
ブラウザ対応	Internet Explorer 11 （IE11）	Chrome / 新 Edge ※ レガシ Edge は対象外
事前インストール	不要 ※ 実行時自動インストール	必要（インストーラ提供） ※ exe ファイル配置とレジストリ登録
提供ファイル	LeCSignCapiA.cab / LeCSignP11A.cab	LeCSignCmd.exe / LeCSignCmdWS.js （ LeCSignCmdSetup.msi ）
サーバ連携	必要（サーバ側に LE:PAdES:Lib か LE:XAdES:Lib が必要） ※ 署名部のサーバ側実装は V1 のままで共通利用が可能	
呼び出し方法	object 要素の API 呼び出し	実行用 JavaScript API 提供 ※ 実行後に WebSocket 通信
署名結果確認方法	API の戻り値で確認可能	
ライセンス	LE:Client:Sign V2 ライセンスが必要 ※ Chrome/Edge 対応は標準機能の為に別料金は不要	

ラング・エッジ LE:Client:Sign V2 の機能

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

１．３．CAPI（CryptoAPI）と PKCS#11 の選択

Windows 環境の署名用 API として、PKCS#11 と CAPI（CryptoAPI）の 2 種類が良く使われる。どちらを使うかは利用する秘密鍵の利用形態に依存する。例えば IC カードの場合には PKCS#11 の API のみ利用可能な場合がある。つまりどちらを利用するかは、利用する証明書と秘密鍵の提供形態に依存する。ActiveX プラグインではコンポーネント自体が異なり、署名コマンドでは初期化フラグでどちらを利用するかを指定する。

	CAPI（CryptoAPI）	PKCS#11
Windows	Windows 標準の証明書ストアと IC カード	DLL ファイル経由で利用
初期化情報	通常は特に情報は必要ないが、CSP 名の指定が必要となる IC カードの場合には CSP 名他の情報を与える	DLL ファイルのパスを与える
IC カード	CAPI 対応の IC カード	PKCS#11 対応の IC カード
補足	PKCS#12 形式で秘密鍵と証明書が提供されている場合には、Windows 証明書ストアにインストールして利用可能	HPKI 等では PKCS#11 の利用が推奨されている

CAPI と PKCS#11 の違い

ActiveX プラグインは CAPI と PKCS#11 それぞれ異なるコンポーネントを利用する。署名コマンド初期化時の引数により CAPI と PKCS#11 を切り替えて利用する。

	CAPI（CryptoAPI）	PKCS#11
ActiveX プラグイン	LeCSignCapiA.cab	LeCSignP11A.cab
署名コマンド	LeCSignCmd.js（LeCSignCmdSetup.msi）	

ActiveX プラグインと署名コマンドの提供モジュール

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

１．４．署名サーバ連携

クライアント署名は署名サーバと連携して署名処理を行う。サーバとクライアント間の通信プロトコルは XML を使った独自仕様となっている。サーバ側にて独自 XML 通信プロトコルに対応した実装を行う必要がある。またサーバ側では取得した通信結果から「仮署名」と「署名値セット」の処理を実装する必要がある。

またサーバ側のライブラリを PDF 署名用の LE:PAdES:Lib（LE:PAdES-Basic:Lib も可）を利用するのか、XML 署名用の LE:XAdES:Lib を利用するのかを決めておく必要がある。なお LE:PAdES:Lib では独自通信プロトコルに対応したクラス PdaClientXml を提供している。

クライアント側 （LE:Client:Sign）	通信 （独自 XML）	サーバ側 （LE:PAdES/XAdES:Lib）	サーバ上 ファイル
1. 事前操作と署名実行（ブラウザ）	← 連携 →	**0. 署名対象の準備** 対象となるファイルの用意 セッション ID の生成（以後通信に利用）	情報取得 署名対象ファイル
拡張機能 **2. 証明書選択 LE:Client:Sign** 証明書の取得/秘密鍵の指定 ＞ 証明書選択画面の表示	← セッション ID 証明書 →	**3. 仮署名 LE:PAdES/XAdES:Lib** 署名値が無い PAdES/XAdES 生成 （署名）対象ハッシュ値の計算 ＞ 署名証明書が必要	仮署名ファイル
拡張機能 **4. 署名値計算 LE:Client:Sign** 対象ハッシュ値から署名値を計算 ＞ 秘密鍵の利用	← ハッシュ値 署名値 →	**5. 署名値セット LE:PAdES/XAdES:Lib** 署名済みの PAdES/XAdES 完成 処理結果（成功/エラー）を返す	署名済みファイル
6. 処理結果の確認（ブラウザ） 必要に応じてリダイレクト等で移動 ※ 署名コマンドは直接結果を取得できないのでリダイレクトまたは通信で処理結果を取得する必要がある。	← 処理結果 リダイレクト →	**7. 署名後の処理（結果表示等）** タイムスタンプ付与や長期署名化も可	保管等

クライアントとサーバ間の連携とサーバ側の処理

※ Ver2.1 よりサーバ連携せずクライアント側で XAdES-BES の生成（ローカル署名）が可能となった。XAdES-BES の生成には API として sign() では無く xades() を利用する。Xades を使う場合も cert() の事前呼び出しは必要となる。

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

１．５．フォルダ構成

フォルダ/ファイル	説明	提供
readme-LeClientSign2.txt	LE:Client:Sign V2 リリースノート	バイナリ版
[bin_activex]	ActiveX モジュール及び試験ファイル	バイナリ版
LeCSignCapiA.cab	CAPI 用 ActiveX プラグイン	バイナリ版
LeCSignP11A.cab	PKCS#11 用 ActiveX プラグイン	バイナリ版
LeCSignIE11.js	LE:Client:Sign V2 フラグ定義 JavaScreipt	バイナリ版
*.htm	IE11 用試験ファイル	バイナリ版
unregistcapi.bat	CAPI 用 ActiveX プラグイン登録解除バッチ	バイナリ版
[bin_cmd]	署名コマンドモジュール及び試験ファイル	バイナリ版
LeCSignCmd.exe	署名コマンド実行ファイル（試験用）	バイナリ版
LeCSignCmdSetup.zip LeCSignCmdSetup.msi	署名コマンドインストーラ（実行用）	バイナリ版
LeCSignCmdWS.js	署名コマンド API 定義 JavaScript	バイナリ版
*.html	Edge/Chrome 用試験ファイル	バイナリ版
Regist.bat Unregist.bat	bin_cmd/LeCSignCmd.exe の登録と解除用のバッチファイル（非インストール試験用）	バイナリ版
[server_pades]	LE:PAdES:Lib のサーバ側実装 Java サンプル	バイナリ版
[ServerTest]	ローカル試験用のダミーサーバ用フォルダ	バイナリ版
[bin]	ダミーサーバ実行環境 ※PATH 環境変数に追加	バイナリ版
[LcsPAdES]	ダミーサーバ LE:PAdES:Lib 実装サンプル	製品版
[LcsXAdES]	ダミーサーバ LE:XAdES:Lib 実装サンプル	製品版
[doc]	マニュアル等のドキュメント	バイナリ版
LeClientSign2-manual.pdf	マニュアル（本ファイル）	バイナリ版
*.pdf	補足ドキュメント	バイナリ版
[include]	ビルド用のインクルードフォルダ	製品版
[LeClientSignA]	ActiveX インターフェイス部実装	製品版
[LeCSignCapi]	CAPI 機能部実装	製品版
[LeCSignCmd]	署名コマンドインターフェイス部実装	製品版
[LeCSignCmdSetup]	署名コマンドインストーラ用プロジェクト	製品版
[LeCSignCommon]	LE:Client:Sign 共通部実装	製品版
[LeCSignP11]	PKCS#11 機能部実装	製品版
[LeXAdES]	ローカル XAdES 機能部実装（V2.1 以降）	製品版
[lib]	ビルド用（libxml2 を含む）	製品版
[WTL]	ビルド用（Microsoft）	製品版
LeClientSign2.sln	ビルド用プロジェクトファイル	製品版

※　バイナリ版：LE:PAdES:Lib/LE:XAdES:Lib での提供、および評価版としての提供。

※　製品版：LE:Client:Sign 正規購入（ソースコード他一式が付属）

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

１．６．利用外部ライブラリとライセンス情報

クライアント署名は基本的には Windows 標準提供されているライブラリのみで構成されていたが、V2.1 の XAdES 生成機能では外部ライブラリとしてオープンソースの libxml2 を利用している。。

モジュール	利用範囲	提供
libxml2	XAdES 生成に利用	オープンソース（http://www.xmlsoft.org/）
CryptoAPI	Windows 証明書ストアの利用	Windows 標準提供
WinHTTP	HTTP/HTTPS 通信に利用 ※ V1.2 からのデフォルト	Windows 標準提供
WinInet	HTTP/HTTPS 通信に利用 ※ V1.1 までのデフォルト ※ LCSI_SET_WININET 指定時利用	Windows 標準提供
msxml3	引数の XML 解析等に利用	Windows 標準提供

利用外部ライブラリ

以下にオープンソースの libxml2 に関するライセンス情報を示す。なおライセンスは商用利用可能な MIT ライセンスとなっている。

ライブラリ	ライセンス（下段はコピーライト表示）
libxml2	MIT License（**コピーライト表示義務あり**）
	Copyright (C) 1998-2003 Daniel Veillard. All Rights Reserved.

オープンソースライセンス

libxml2 のコピーライト表示

```
[libxml2 License]
This product includesIftwares developed by:
 Copyright (C) 1998-2003 Daniel Veillard.  All Rights Reserved.
 http://www.xmlsoft.org/
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

１．７．HTTP 通信の仕様変更（V2.2）

LE:Client:Sign の Ver2.2.R1（2026 年 1 月リリース）では大きな仕様変更として、HTTP/HTTPS 通信が WinInet から標準 WinHTTP への切り替えがあった。デフォルト設定の変更であり、別途オプション指定により従来通りの動作も可能となっている。

仕様変更：HTTP/HTTPS 通信モジュールが WinInet から WinHTTP になった

対象環境	署名コマンド（LeCSignCmd）
影響	プロキシ設定として WinHTTP 用に変更する必要がある可能性がある。ただし WinHTTP は Windows Update にも使われる通信モジュールであり、通常であればプロキシ等の設定は済んでいるはず。
変更理由	WinInet は IE の通信モジュールであり、IE 廃止に伴い将来的に対応されなくなる可能性がある為に Windows Update にも使われている WinHTTP に変更した。なお WinInet が廃止されると言う情報はまだ無い。
従来仕様指定	LeCSignCmdWS クラスのコンストラクタの初期化フラグ（initflag）に、LCSI_SET_WININET（0x00008000）を指定することで、Ver2.1 までと同じく WinInet を利用する。詳しくは「3.3.1. constructor：生成/初期化」を参照。

WinHTTP/WinInet のどちらでも環境変数 LE_HTTP_TIMEOUT により HTTP 通信のタイムアウト時間が設定できるようになった。

○ Windows 版の環境変数による設定項目（WinHTTP/WinInet 利用時）

環境変数	説明
LE_HTTP_TIMEOUT	HTTP 通信時のタイムアウト時間を秒で設定。 30 秒未満が指定された場合は 30 秒となる（最低 30 秒）。 省略時にはシステムデフォルトの 30 秒となる。

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

１．８．HTTP 通信のプロキシ設定

１）WinHTTP 利用時のプロキシ設定（V2.2 のデフォルト設定）

WinHTTP は Windows Update にも使われている。この為にプロキシ設定は Windows システムの設定で良い。設定の「ネットワークとインターネット」＞「プロキシ」の「プロキシサーバーを使う」の「詳細」により手動による設定も可能となっている。

現在認証付きのプロキシ接続に対応する場合には、以下の環境変数としてユーザ名とパスワードをセットする。なお現在は Basic 認証のみに対応している。NTLM/PASSPORT/DIGEST 等の認証が必要な場合は別途対応が必要となるのでご相談ください。Ver2.2 より設定に proxy.ini ファイルを利用したプロキシ設定に対応した。proxy.ini ファイルの置き場所は以下のいずれかとなる。

１：実行モジュール(LeCSignCmd.exe)と同じディレクトリ下の proxy.ini
２：LE_PROXY_INIPATH 環境変数により指定された proxy.ini 利用

proxy.ini の設定は現在 1 行だけで、プロキシの URL を指定する。
例 http://133.242.232.124:3128/　　※ この例の IP アドレスは利用できません

○ Windows 版の環境変数による設定項目（WinHTTP 利用時）

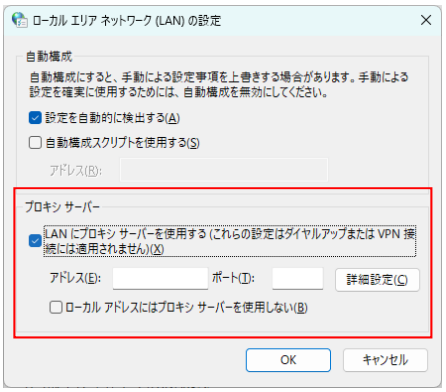
環境変数	説明
LE_PROXY_USERNAME	プロキシのユーザ名
LE_PROXY_PASSWORD	プロキシのパスワード
LE_PROXY_INIPATH	proxy.ini ファイルのパスを指定（V2.2 以降利用可） 設定例："c:¥test_setup¥proxy.ini"

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

2）WinInet 利用時のプロキシ設定（V1.1 までのデフォルト設定：LCSI_SET_WININET 指定時）

WinInet は本来 IE（InternetExplorer）のモジュールであるので、IE のプロキシ設定がそのまま利用される。プロキシ設定は基本的にはログインしているユーザ毎に設定が必要となるので、特に ASP 環境等では注意が必要となる。PC 毎に設定をする場合は以下の手順で可能となる。

※ インターネットのプロパティから設定する

1	管理者権限のあるユーザでログオンする。
2	[ファイル名を指定して実行] で [inetcpl.cpl] 入力して[OK]することで[インターネットのプロパティ] を起動する。 ※ Windows キー＋R の同時押しで［ファイル名を指定して実行］の表示が可能。
3	[接続] タブを開き、[LAN の設定] ボタンをクリックして、[プロキシ サーバーを使用する] にチェックを入れ［アドレス］と［ポート］を入力する。 
4	[OK] と [OK] で画面を閉じる。

※ レジストリを直接編集（スクリプトやポリシーで使う場合）

レジストリキー：

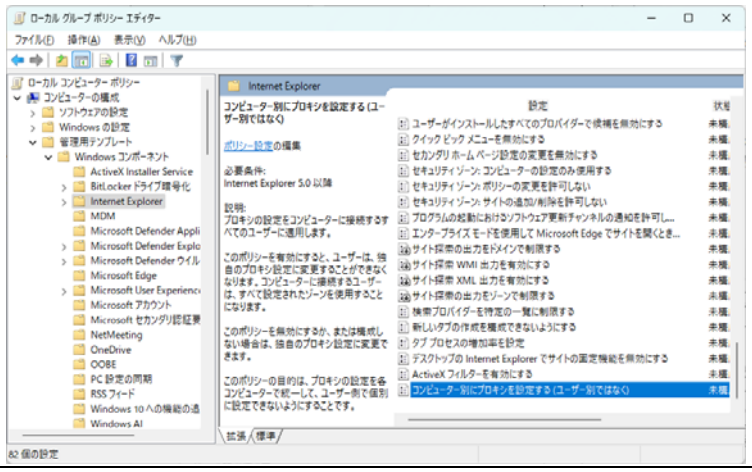
HKEY_CURRENT_USER¥Software¥Microsoft¥Windows¥CurrentVersion¥Internet Settings

値：

名前	種別	設定例
ProxyEnable	DWORD	1 で有効、0 で無効
ProxyServer	文字列	proxy.example.com:8080
ProxyOverride	文字列	localhost;*.example.com

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

※ ローカルグループポリシーエディタから設定のリセット（必要に応じて）

1	管理者権限のあるユーザでログオンする。
2	[ファイル名を指定して実行] で [gpedit.msc] 入力して[OK]することで[ローカルグループポリシーエディタ] を起動する。 ※ Windows キー＋R の同時押しで［ファイル名を指定して実行］の表示が可能。
3	[コンピュータの構成] - [管理用テンプレート] - [Windows コンポーネント] - [Internet Explorer] を開き、[コンピュータ別にプロキシを設定する（ユーザ別ではなく）] を有効にする。 ※ 項目のダブルクリックで編集可能になるので有効にセットする。 
4	IE（InternetExplorer）プロキシ設定がリセットされるので再度設定する。

現在認証付きのプロキシ接続に対応する場合には、以下の環境変数としてユーザ名とパスワードをセットする。また Ver2.2 より設定に proxy.ini ファイルを利用したプロキシ設定に対応した。proxy.ini ファイルの置き場所は以下のいずれかとなる。

1 ：実行モジュール(LeCSignCmd.exe)と同じディレクトリ下の proxy.ini
2 ：LE_PROXY_INIPATH 環境変数により指定された proxy.ini 利用

proxy.ini の設定は現在 1 行だけで、プロキシの URL を指定する。
例 http://133.242.232.124:3128/ ※ この例の IP アドレスは利用できません

○ Windows 版の環境変数による設定項目（WinInet 利用時）

環境変数	説明
LE_PROXY_USERNAME	プロキシのユーザ名
LE_PROXY_PASSWORD	プロキシのパスワード
LE_PROXY_INIPATH	proxy.ini ファイルのパスを指定（Windows 版 Ver1.09.R2 以降利用可） 設定例："c:¥test_setup¥proxy.ini"

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．クライアント側実装：ActiveX プラグイン（LE:Client:Sign）

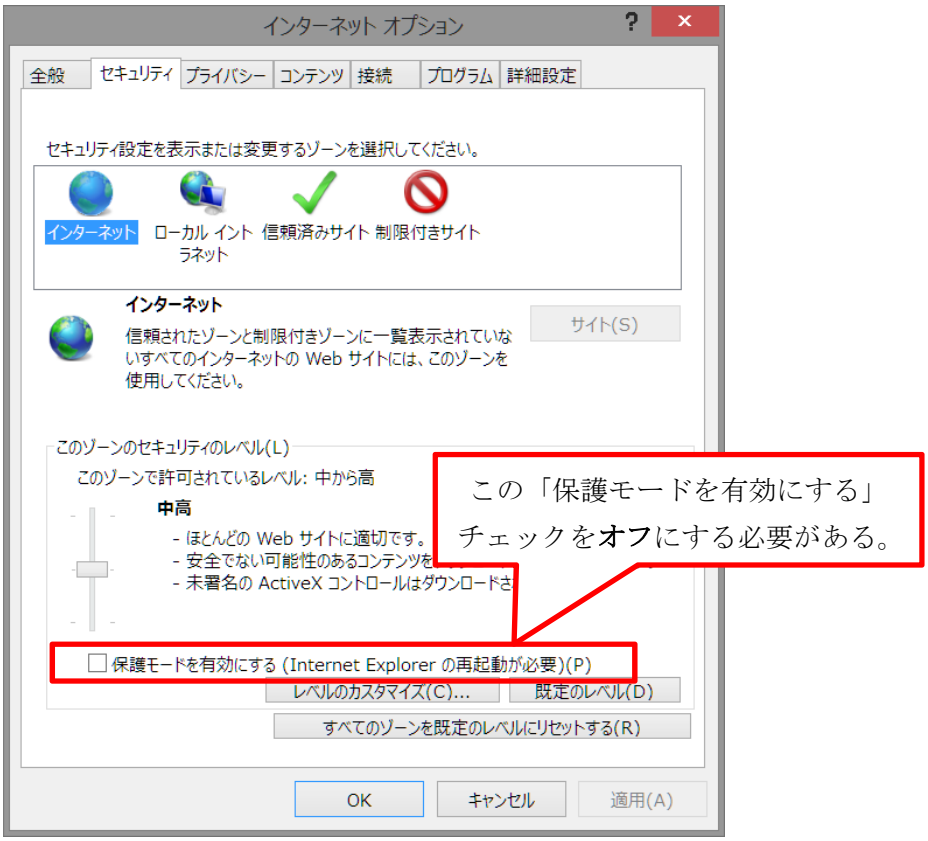
２．１．ActiveX プラグインモジュール

ActiveX プラグインを JavaScript から使う場合にバイナリ型は利用しない。バイナリ型の引数や戻り値は、HEX 形式の文字列として扱う。サーバとの通信部は基本的に JavaScript で実装するか、一括実行の exec()「2.1.16 参照」を利用する。

種類	ファイル名	CLSID	利用
PKCS#11	LeCSignP11A.cab	5F494C03-3D02-4E8A-8211-FBD51E1DF5B0	PKCS#11 対応の IC カード
CAPI	LeCSignCapiA.cab	2DB6008C-CF18-49C4-9CDC-1075B34414BE	Windows 証明書ストア CAPI 対応 IC カード
定義	LeCSignIE11.js	---	定義用：オプション

ActiveX プラグインの仕様

注意：ActiveX プラグインを利用する場合、PKCS#11 の場合にはローカル DLL ファイルへのアクセスの為、CAPI の場合には Windows 証明書ストアへのアクセスの為に、**IE のインターネットオプションの「保護モード」を無効にする**必要があります。



Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

インターネットオプションの保護モードを無効にする

２．２．ActiveX プラグイン 利用シーケンス

ActiveX プラグインでクライアント署名を利用するには大きく初期化・証明書取得（署名ハッシュ値計算）・署名値計算（署名値埋め込み）・終了の 4 処理が必要になるが、その間にサーバとの HTTP 通信と、XML 等による通信内容記述（セッション ID と情報の受け渡し）が必要となる。HTTP 通信と通信内容はカスタマイズが必要なケースもあるだろう。また本クライアント署名においては、HTTP 通信と XML 解析の API を個別提供しているので組み合わせて利用が可能となっている。また独自の通信 XML も用意しており、これら独自の通信手順を 1 つの一括実行 API で利用することも可能となっている。

サーバ連携の署名時の手順：

処理内容	個別 API	一括 API	処理詳細（一括）
初期化 （IC カード状態）	**init** status	**exec**	**初期化（※必須）**/ dll（P11）・証明書ストア（CAPI） P11 では status で状態取得（オプション）
通信初期化	(httpSet)	↓	サーバホスト名等のセット（オプション）
証明書取得	**cert**	↓	**証明書の取得（※必須）**
証明書 HTTP 通信	(httpSend)	↓	一括 API では独自 XML の送信/取得（オプション）
ハッシュ値取得	(xmlGet)	↓	一括 API では独自 XML の解析（オプション）
署名値計算	**sign**	↓	**署名値の計算（※必須）**
署名値 HTTP 通信	(httpSend)	↓	一括 API では独自 XML の送信/取得（オプション）
結果取得	(xmlGet)	↓	一括 API では独自 XML の解析（オプション）
解放	**term**	↓	**解放やリダイレクト（オプション）**

クライアント側の処理手順概要（ActiveX プラグイン）

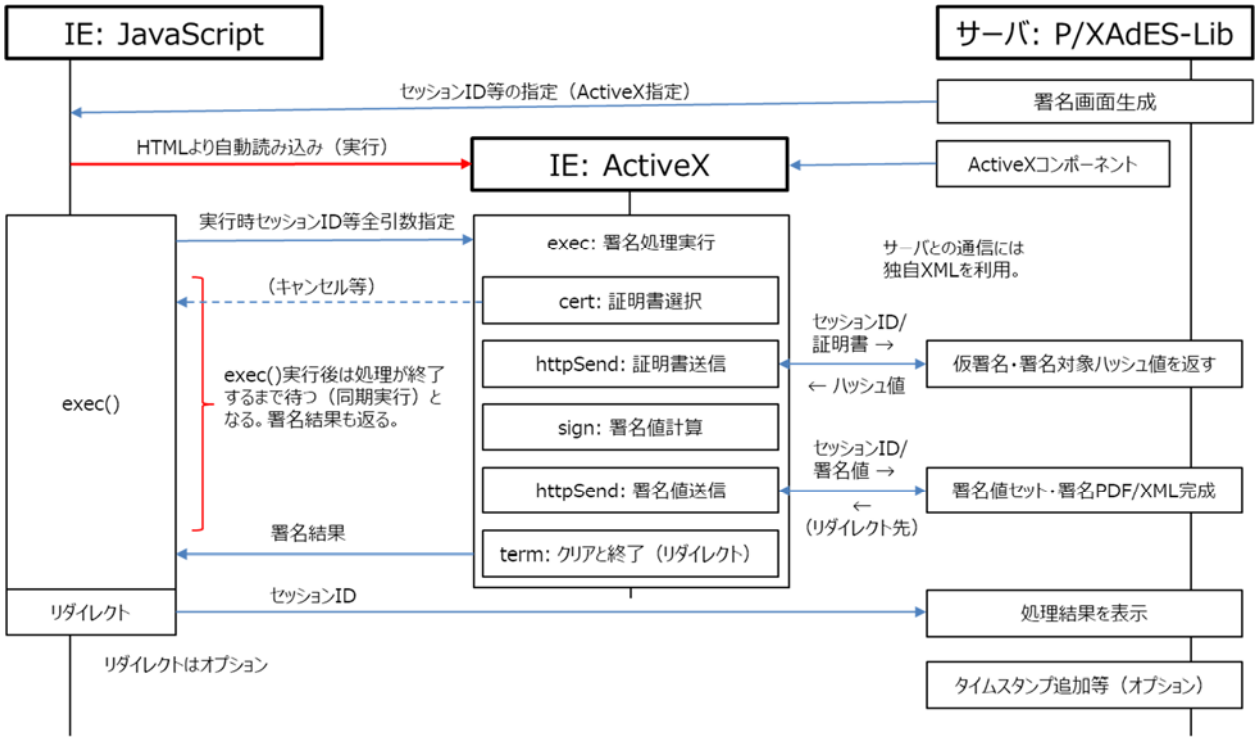
ローカルでの XAdES-BES 生成時の手順（Ver2.1 以降）：

処理内容	個別 API	処理詳細（一括）
初期化 （IC カード状態）	**init** status	**初期化（※必須）**/ dll（P11）・証明書ストア（CAPI） P11 では status で状態取得（オプション）
通信初期化	(httpSet)	サーバホスト名等のセット（オプション）
証明書取得	**cert**	**証明書の取得（※必須）**
証明書情報取得	(info)	証明書情報の取得（オプション）
XAdES-BES 対象追加	(xadd)	Detached/Enveloping 署名対象の追加（オプション）
XAdES-BES 署名	**xades**	**署名値の計算と XAdES-BES の生成（※必須）**
XAdES-BES 送信	(httpSend)	署名済み XAdES-BES 情報の送信（オプション） ※ 別の方法で取得するなら通信は不要
解放	**term**	**解放やリダイレクト（オプション）**

クライアント側の処理手順概要（ActiveX プラグイン）

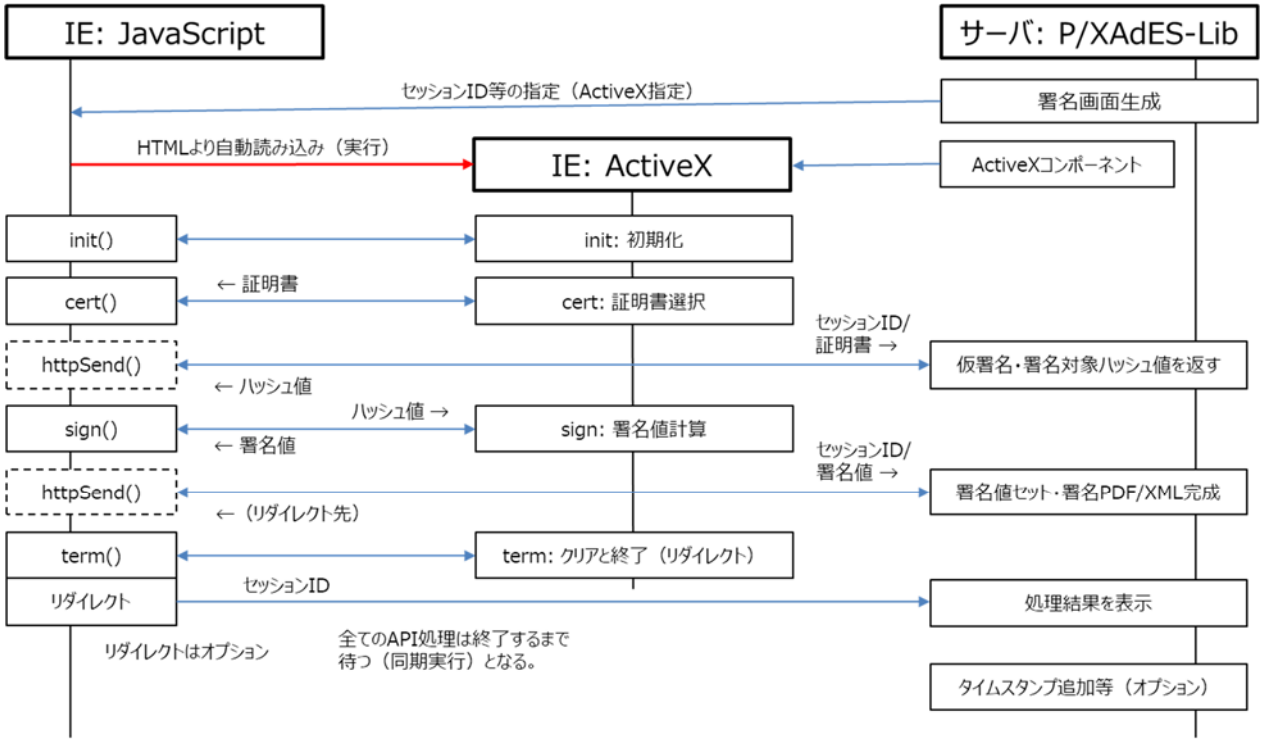
Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．２．１．一括 API 利用シーケンス



ActiveX プラグイン 一括 API 利用シーケンス図

２．２．２．個別 API 利用シーケンス



ActiveX プラグイン 個別 API 利用シーケンス図

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．３．ActiveX 利用 API

クライアント署名の API は、ActiveX ではオブジェクトのメソッド、Java API では LeClientSign クラスのメンバとして提供される。

メソッド名	必須	機能	説明/補足
init ()	○	初期化	PKCS#11/CAPI 等の初期化を実行
cert ()	○	証明書取得	署名に利用する証明書を選択し取得 ※ 連携するサーバに証明書を送信
info ()	－	証明書情報取得	証明書を解析して情報を取得する ※ Ver2.1 よりサポート
sign ()	△	署名値計算	与えられるハッシュ/データの署名値を計算 ※ 連携するサーバからハッシュ/データを取得 ※ 連携するサーバに署名値を送信
xades ()	△	XAdES-BES 生成	指定された対象に XAdES-BES 署名を付与して生成する ※ xades() 利用時には sign() は使わない ※ Ver2.1 よりサポート
term ()	△	後処理（解放）	プラグイン終了時に解放される為必須では無い
errorGet ()	△	エラー値取得	詳細は「付録１．エラーコード一覧」を参照
errorApi ()	×	エラーAPI 名取得	エラー生じた内部位置を確認する為の補助情報 例）PKCS#11 や CAPI の API 名
errorClear ()	×	エラー値クリア	エラー値を 0（NO_ERROR）クリア
httpSet ()	×	ネットワーク設定	セッション ID とベース URL を設定
httpSend ()	×	ネットワーク接続	HTTP 通信の POST コマンドを実行
version ()	×	バージョン情報取得	クライアント署名モジュールのバージョン情報
status ()	×	IC カード状態取得	PKCS#11 時のみ利用可能
xmlGet ()	×	xml 要素値/属性取得	XPATH 指定により XML 要素値/属性を取得
exec ()	○	一括実行	独自仕様による署名までの一括実行

クライアント署名 API 一覧

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

手順	内容	補足
Step0	事前：初期画面生成/ブラウザセット ※ サーバ側で生成しブラウザ表示	セッション ID や通信先を埋め込んだ、JavaScript や HTML を生成してブラウザに表示する
Step1	機能 1：初期化	init() を呼び出しドライバや設定を初期化 必要なら httpSet() も行う
Step2	アクション：署名トリガー	署名ボタンの onClick() 等で署名処理を開始
Step3	機能 2：証明書取得	cert() を呼び出して証明書を取得する
Step4	通信 1：証明書送信/署名対象取得 ※ セッション ID 等も必要	httpSend() または何らかの通信手段により、 証明書をサーバに送信して、署名対象を取得
Step5	機能 3：署名値計算	sign() を呼び出して署名値を取得する
Step6	通信 2：署名値と結果の取得 ※ セッション ID 等も必要	httpSend() または何らかの通信手段により、 署名値をサーバに送信して、結果を取得
Step7	機能 4：後処理（省略可）	結果次第でリダイレクト等の後処理を行う

クライアント署名の基本的な処理手順

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．３．１．init：初期化

機能		init：プラグイン初期化：署名用のドライバや設定の初期化		
ActiveX		BSTR init (ULONG iflag, LPCTSTR iopt);		
引数	1	フラグ	iflag	初期化フラグ：共通部 0x00000000 LCSI_NO_FLAG XML 形式で結果を返す 0x00000004 LCSI_SET_RETHEX HEX 文字列で結果を返す 0x00000008 LCSI_SET_RETB64 Base64 文字列で結果を返す 0x00001000 LCSI_SET_DEBUG デバッグモードを利用
	2	文字列	iopt	オプション：「PKCS#11 補足」「CAPI 補足」を参照 PKCS#11：優先利用の DLL ファイルのパスを指定可能 CAPI：追加の IC カード設定を指定可能 ※ 複数指定時は改行コード[¥n]を利用して繋げる ※ 省略時は空文字指定（null 指定不可）
戻り値		文字列		正常時： PKCS#11：正常時には利用する DLL パス文字列が返る CAPI：正常時には NULL が返る エラー時： エラーメッセージが返る（"Error:"で始まる文字列） 別途 errorGet()/errorApi() の利用も可能
PKCS#11 補足		初期化フラグ： 0x00000010 LCSI_P11_FILEDLG ファイルダイアログで dll 指定 オプション（PKCS#11 ドライバの DLL 指定）： ※ opt に最低限１つ以上の DLL ファイルパスの指定が必要（NULL 不可）		

		// ActiveX プラグインの JavaScript からの利用例（dll を 2 つ指定）
		var paths = "C:¥¥Program Files¥¥JPKI¥¥JPKIPKCS11Sign.dll" + "¥n" // path1
		+ "C:¥¥Windows¥¥System32¥¥p11driver.dll"; // path2
		var ret = signobj.init(paths, iflag);		

CAPI 補足		初期化フラグ： 0x00000100 LCSI_CAPI_ICCARD 標準 IC カード(JPKI/特定 CA 等)のチェックも行う 0x00000200 LCSI_CAPI_NOSTORE 個人 Windows 証明書ストアをチェックしない 0x00000800 LCSI_CAPI_NOCHECK 証明書チェック無し(LCSC_CAPI_ICCARD と利用) オプション（追加 IC カード設定の指定）： var cards = "24:-1:0:JPKI Crypto Service Provider for Sign"; var ret = signobj.init(cards, iflag); ※ opt（追加 IC カード設定）は使わなければ NULL で良い ※ CAPI IC カードのオプション指定内容：		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

	形式："PROV_TYPE:KEY_TYPE:FLAG:CSP_NAME" PROV_TYPE ： PROV_RSA_FULL = 1 / PROV_RSA_AES = 24 KEY_TYPE ： AT_KEYEXCHANGE = 1 / AT_SIGNATURE = -1 FLAG ： NONE = 0 / CRYPT_SILENT = 0x00000040 (64) EX1) "24:-1:0:JPKI Crypto Service Provider for Sign" EX2) "24:-1:64:NEC Secure Ware AES Cryptographic Provider"
V2.1 補足	1. LCSI_SET_RETXML（0x00000002）は廃止されデフォルト設定が XML を返すようになった。LCSI_SET_RETXML は指定しても無視される。 2. LCSI_CAPI_NOCHECK（0x00000800）が追加され指定時には init では証明書チェックは一切行わない。その代わり cert にて LCSC_CAPI_ICCARD（0x00000100）を指定して IC カード（マイナンバーカードを含む標準 IC カード）のチェックを行う。これによりカード未挿入時の再チェックと初期化が可能となる。

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．３．２．cert：証明書取得

機能		cert：証明書取得：署名に使う証明書を選択する		
ActiveX		BSTR cert（　　// HEX 文字列で返される ULONG cflag, LPCTSTR copt);		
引数	1	フラグ	cflag	証明書フラグ：共通部 0x00000000　LCSC_NO_FLAG　　オプション無し 0x00000001　LCSC_CONFIRM　　証明書１つの場合も確認
	2	文字列	copt	オプション：省略する場合は空文字指定（null 指定不可） 利用する証明書を制限する（未サポート）
戻り値		バイナリ		正常時： 選択した証明書の X.509/DER 形式バイナリが返る ※ 標準は独自 XML 形式で返る（4.1.1.参照） ※ init.LCSI_SET_RETHEX 指定時は HEX 文字列で返る ※ init.LCSI_SET_RETB64 指定時は Base64 文字列で返る エラー時： NULL が返る 別途 errorGet()/errorApi() の利用も可能
PKCS#11 補足		証明書フラグ： 0x00000010　LCSC_P11_ALLCERT　CA 証明書も含め全ての証明書を表示 0x00000020　LCSC_P11_USEPIN　　ログイン（PIN 必要）をして利用（JPKI 等） オプション（LCSC_P11_USEPIN 指定時）： opt に PIN 文字列の指定が可能（NULL 指定で独自ダイアログ）		
CAPI 補足		証明書フラグ： 0x00000100　LCSC_CAPI_ICCARD　標準 IC カード(JPKI/特定 CA 等)のチェックを行う 説明： 通常は初期化（init）時に証明書や IC カードのチェックを行うが、初期化時のフラグに LCSI_CAPI_NOCHECK（0x00000800）を指定することで、cert の証明書フラグ LCSC_CAPI_ICCARD（0x00000100）を指定することで IC カードのチェックを行える。 LCSC_CAPI_ICCARD したのに LCS_NO_CERT_ERR（-11）となる場合には IC カードが挿入されていない。		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．３．３．info：証明書情報取得

機能		info：証明書情報取得：証明書を解析して情報を取得する		
ActiveX		BSTR info（　　// HEX 文字列で返される LPCTSTR cert, ULONG infoflag);		
引数	1	文字列	copt	解析する証明書の X.509/DER 形式バイナリの Base64 文字列 ※ cert 戻り値の独自 XML 形式の指定も可能。つまり cert の戻り値をそのまま指定可能。 ※ 省略不可なので、空文字や null 指定不可
	2	数値	infoflag	情報フラグ： 0x00000000　LCSF_NO_FLAG　　オプション無し（JSON 返し） 0x00000001　LCSF_RETXML　　XML 形式で情報を返す
戻り値		文字列		正常時： 選択した証明書の情報が JSON 文字列で返る ※ LCSF_RETXML 指定時は独自 XML 形式で返る エラー時： NULL が返る 別途 errorGet() の利用も可能
補足		戻り値で返される情報の形式は「付録２．証明書情報一覧」を参照		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．３．４．sign：署名値計算

機能		sign：署名値計算：署名を実行して署名値を返す（直前の選択証明書を利用）		
ActiveX		BSTR sign（　　// HEX 文字列で返される ULONG sflag, LPCTSTR sopt LONG signAlg, LPCTSTR data,　　// HEX 文字列で指定 LPCTSTR passwd)；		
引数	1	フラグ	sflag	署名フラグ：共通部 0x00000000　LCSS_DEFAULT　　data にハッシュ値を指定 0x00000001　LCSS_TDIRECT　　data に対象自体を指定 0x00000008　LCSS_DATAHEX　　data を HEX 文字列で指定
	2	文字列	sopt	オプション：省略する場合は空文字指定（null 指定不可） 利用する証明書を指定する（未サポート）
	3	数値	signAlg	署名アルゴリズム： SHA256=0 / SHA384=1 / SHA512=2 / SHA1=10
	4	バイナリ	Data	署名対象データ： LCSS_DEFAULT 時 = 署名対象のハッシュ値を指定 LCSS_TDIRECT 時 = 署名対象自体を指定（ハッシュ計算含） ※ ActiveX か LCSS_DATAHEX 指定時は HEX 文字列形式で指定
	5	文字列	passwd	PKCS#11： 利用パスワード指定（省略空文字の場合は通常エラー） ※ LCSS_P11_NODLG 未指定時は空文字指定で独自ダイアログ CAPI： 未使用（IC カード利用時はドライバ側でダイアログ表示） 省略時は空文字指定（null 指定不可）
戻り値		バイナリ		正常時： 選択した署名値のバイナリが返る ※ LCSI_RETXML 指定時は独自 XML 形式で返る（4.1.3.参照） ※ ActiveX か LCSS_RETHEX 指定時は HEX 文字列形式で返る エラー時： NULL が返る 別途 errorGet()/errorApi() の利用も可能
PKCS#11 補足		署名フラグ： 0x00000010　LCSS_P11_NODLG　　passwd が NULL の場合にエラーにする		
CAPI 補足				

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．３．５．xadd：署名 XAdES-BES への署名対象追加

機能		xadd：署名 XAdES-BES への署名対象追加：Detached の追加 （複数回呼び出すことで複数の署名対象追加が可能）		
ActiveX		LONG xadd (LONG type, LPCTSTR id, LPCTSTR fname, LPCTSTR data);　　// HEX 文字列で指定		
引数	**1**	数値	type	XAdES 対象追加種別： 0x00000000　LCSA_DETACHED　Detached 形式で対象を追加 0x00000001　LCSA_ENVELOPING　Enveloping 形式で対象を追加 ※ Enveloping の追加は現在未対応
	2	文字列	id	ID 文字列を指定、省略不可、例："DET01"
	3	文字列	fname	ファイル名を指定 Enveloping 時には data を指定すれば省略可能 Detached 指定時には省略不可、例："sample.docx" ※ フルパスを指定した場合にはファイル名と拡張子部のみが利用され、フルパスはデータの読み込みのみに利用される。 ※ 省略時は空文字指定（null 指定不可）
	4	文字列	data	署名対象バイナリを Base64 文字列化したデータ指定 ※ 空指定にて省略可能（指定時にはフルパス指定が必要）
戻り値		数値	正常時： LCS_OK が返る エラー時： LCS_OK 以外が返る（マイナス値）	
補足		※ 現在 Enveloping（内包）形式の署名対象追加は未サポート		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．３．６．xades：署名 XAdES-BES 生成

機能		xades：署名 XAdES-BES 生成：XAdES-BES 署名を実行して XAdES の XML を返す （直前の選択証明書を利用） ※ 本機能ではフルセットの XAdES では無く簡易な XAdES-BES のみ対応		
ActiveX		BSTR xades（　// HEX 文字列で返される ULONG xflag, LONG signAlg, LPCTSTR xml,　// HEX 文字列で指定 LPCTSTR id, LPCTSTR xpath, LPCTSTR passwd); 		
引数	**1**	数値	xflag	XAdES フラグ：共通部 0x00000000　LCSX_DEFAULT　共通：指定無し
	2	数値	signAlg	署名アルゴリズム： SHA256=0 / SHA384=1 / SHA512=2 / SHA1=10
	3	文字列	xml	署名対象 XML： 省略（空文字）の場合は Detached/Enveloping の指定が必要
	4	文字列	id	署名対象 id の指定：xml 中の署名対象となる id を指定 省略（空文字）の場合は Enveloped 形式となる
	5	文字列	xpath	署名対象の xpath の指定：オプション 空文字にて省略可能
	6	文字列	passwd	PKCS#11： 利用パスワード指定（NULL の場合は通常エラー） ※ LCSX_P11_NODLG 未指定時は空文字指定で独自ダイアログ CAPI： 未使用（IC カード利用時はドライバ側でダイアログ表示）
戻り値		文字列	正常時： 生成した XAdES-BES の XML が返される ※ 初期化に LCSI_SET_RETHEX 指定時は HEX 文字列で返される ※ 初期化に LCSI_SET_RETB64 指定時は Base64 文字列で返される エラー時： NULL が返る 別途 errorGet()/errorApi() の利用も可能	
PKCS#11 補足		署名フラグ： 0x00000010　LCSX_P11_NODLG　passwd が NULL の場合にエラーにする		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．３．７．term：解放（終了処理）

機能		term：後処理（解放）：独自に確保したリソースを解放する		
ActiveX		LONG term (ULONG tflag, LPCTSTR topt);		
引数	1	フラグ	tflag	解放フラグ：共通部 0x00000000　LCST_NO_FLAG　　オプション無し
	2	文字列	sopt	オプション：省略する場合は空文字指定（null 指定不可）
戻り値		数値（エラーコード）		正常時： LCS_OK が返る エラー時： LCS_OK 以外が返る（マイナス値） 別途 errorGet()/errorApi() の利用も可能
ActiveX 補足		ActiveX 利用の場合には topt にリダイレクト先 URI の指定が可能 処理終了後に指定された URI にリダイレクトされます		

２．３．８．errorGet：エラー値取得

機能		errorGet：エラー値取得：直前に生じたエラー値を返す	
ActiveX		LONG errorGet ();	
引数	1	無し	
戻り値		数値（エラーコード）	正常時： LCS_OK が返る エラー時： LCS_OK 以外が返る（マイナス値）

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．３．９．errorApi：エラーAPI 名取得

機能		errorApi：エラーAPI 名取得：直前に生じたエラーの API 名を返す	
ActiveX		BSTR errorApi ();	
引数	1	無し	
戻り値		文字列	正常時： NULL が返る エラー時： エラーを生じた API 名（PKCS#11/CAPI/その他）が返る

２．３．１０．errorClear：エラー値クリア

機能		errorClear：エラー値クリア：エラー値とエラーAPI 名をクリアする
ActiveX		VOID errorClear ();
引数	1	無し
戻り値		無し

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．３．１１．httpSet：HTTP 通信用設定

機能		httpSet：HTTP 通信設定：ホスト名（port 付可）とセッション ID をセットする		
ActiveX		LONG httpSet (LPCTSTR host, LPCTSTR session, LPCTSTR hopt);		
引数	1	文字列	host	スキームとホスト名をセット、オプションでポート番号： 形式："scheme://host(:port)" 例 1）"https://www.langedge.jp" 例 2）"http://test.langedge.jp:3000"
	2	文字列	sid	セッション ID、無ければ NULL 指定
	3	文字列	hopt	HTTP 通信オプション
戻り値		数値（エラーコード）		正常時： LCS_OK が返る エラー時： LCS_OK 以外が返る（マイナス値）
デバッグ		host に "debug://LcsXAdES.exe" を指定することで、LcsXAdES.exe を使ったローカル試験の実行が可能になっている。LcsXAdES.exe が入っているフォルダを PATH 環境変数に加える必要がある。LeClientSign¥ServerTest¥bin を PATH に加える。 LcsXAdES.exe は、-cert で証明書取得処理を、-sign で署名値計算処理のサーバ部同等処理を行なえる。標準入力にクライアント生成の独自 XML を与えると、標準出力に LcsXAdES.exe が生成した独自 XML を返す。		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．３．１２．httpSend：HTTP 通信実行

機能		httpSend：HTTP 通信実行：文字列（XML/JSON 等）を POST メソッドで送信する ※ 先に httpSet() でホスト名をセットする必要あり ※ WinInet を利用した通信		
ActiveX		BSTR httpSend (LPCTSTR path, LPCTSTR header, LPCTSTR data, LPCTSTR agent, ULONG hflag);		
引数	1	文字列	path	通信先のパスを指定（inetSet の host と結合して利用） ※ デバッグ時には LcsXAdES.exe の引数として利用。
	2	文字列	header	ヘッダ指定があればセット、無ければ NULL 指定 例 1）"Content-Type: application/json; charset=utf-8" 例 2）"Authorization: Basic dGVzdDpwc3dk" ※ 複数セットする場合は "¥r¥n" で連結する
	3	文字列	data	通信内容を文字列で指定（バイナリ指定はできない）
	4	文字列	agent	エージェント名指定があればセット、無ければ NULL 指定 ※ 省略時には "Mozilla/4.0 (compatible; MSIE 6.0; Win32)" を使用
	5	フラグ	hflag	HTTP 通信フラグ： 0x00000000　LCSH_NO_FLAG　　オプション無し
戻り値		文字列		正常時： サーバからの応答が返る エラー時： エラーメッセージが返る（"Error:"で始まる文字列） 別途 errorGet()/errorApi() の利用も可能

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．３．１３．version：バージョン情報取得

機能		version：バージョン情報取得：クライアント署名のバージョン情報を返す	
ActiveX		BSTR version ();	
引数	1	無し	
戻り値		文字列	バージョン情報文字列が返る 例）"API:v1.00B1"

２．３．１４．status：IC カード状態取得（PKCS#11 のみ）

機能		status：IC カード状態取得：IC カードの状態を返す（PKCS#11 のみ）	
ActiveX		LONG status ();	
引数	1	無し	
戻り値		数値	正常時： IC カードがセット済みなら 1 が未セットなら 0 が返る エラー時： マイナス値が返る

２．３．１５．xmlGet：XML 値取得

機能		xmlGet：XML から値を取得：XPATH で指定された値を取得		
ActiveX		BSTR xmlElmt (LPCTSTR xml, LPCTSTR xpath);		
引数	1	文字列	xml	解析する XML を指定
	2	文字列	xpath	取得する XML 要素を XPATH 形式で指定 例 1：独自 XML のハッシュ値（要素）を取得 "/LcsResponse/hash" 例 2：独自 XML のリダイレクト属性を取得 "/LcsResponse/@redirect"
戻り値		文字列		正常時： XML の値が返る エラー時： NULL が返る

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．３．１６．exec：一括実行

機能		exec：一括実行：独自 HTTP 通信/XML 形式にて署名処理を一括実行する		
ActiveX		LONG exec（ ULONG iflag, // <param name="iflag" value=0> LPCTSTR iopt, // <param name="iopt" value=""> LPCTSTR host, // <param name="sid" value="session001"> LPCTSTR sid, // <param name="host" value="http://langedge.jp"> LPCTSTR cpath, // <param name="cpath" value="/cert"> LPCTSTR spath, // <param name="spath" value="/sign"> LONG signAlg, // <param name="salg" value=0> ULONG eflag, // <param name="eflag" value=0> LPCTSTR eopt); // <param name="eopt" value="">		
引数	**1**	フラグ	iflag	init の iflag と同じ「2.3.1 参照」 param 利用時には -1 を指定
	2	文字列	iopt	init の iopt と同じ「2.3.2 参照」 param 利用時には NULL か空文字 "" を指定
	3	文字列	host	httpSet の host と同じ「2.3.11 参照」 param 利用時には NULL か空文字 "" を指定
	4	文字列	sid	httpSet の sid と同じ「2.3.11 参照」 param 利用時には NULL か空文字 "" を指定
	5	文字列	cpath	証明書通信のパス httpSend の path と同じ「2.3.12 参照」 param 利用時には NULL か空文字 "" を指定
	6	文字列	spath	署名値通信のパス httpSend の path と同じ「2.3.12 参照」 param 利用時には NULL か空文字 "" を指定
	7	数値	signAlg	sign の signAlg と同じ「2.3.4 参照」 param 利用時には -1 を指定
	8	フラグ	eflag	一括実行フラグ：共通部 0x00000000 LCSE_NO_FLAG オプション無し param 利用時には -1 を指定
	9	文字列	eopt	オプション：現在未サポート param 利用時には NULL か空文字 "" を指定
戻り値		数値（エラーコード）		正常時： LCS_OK が返る エラー時： LCS_OK 以外が返る（マイナス値）
ActiveX		引数は全て別途パラメーター（param）指定により指定可能		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

補足	引数とパラメーターの両方で指定された場合には引数が優先される

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．４．ActiveX プラグイン実装例

２．４．１．LeCSignCapiA.cab / CAPI 版 個別 API の利用

実装サンプル：bin_activex/test-capi.htm

ActiveX による CAPI を利用したクライアント署名プラグイン。Windows 証明書ストアの個人にある証明書と、CAPI に対応した IC カードや USB トークンにてクライアント署名が可能となる。

HTML 部の例：タイトルと署名ボタンのみ表示

```
<body>
<p>LeClientSign: ActiveX CAPI Sample 1</p>
<p>[ PARTS API execution ]</p>
<hr/>
LOCAL debug: Need LcsXAdES.exe and LcsPAdES.exe pre-installed.
<p><button onclick="SignMethod(true, true)">XAdES CAPI SIGN (Local Debug)</button></p>
<p><button onclick="SignMethod(false, true)">PAdES CAPI SIGN (Local Debug)</button></p>
<hr/>
HTTP connection: Use LE Dummy Server.
<p><button onclick="SignMethod(true, false)">XAdES CAPI SIGN (Remote)</button></p>
<p><button onclick="SignMethod(false, false)">PAdES CAPI SIGN (Remote)</button></p>
<hr/>
<object id="LeCSignCapiA" width="0" height="0"
  CODEBASE="./LeCSignCapiA.cab#Version=1,1,1,0"
  classid="CLSID:2DB6008C-CF18-49C4-9CDC-1075B34414BE">
</object>
</body>
```

JavaScript 初期化の例：

```
// 初期化処理
function InitLcs(csign) {
    // 初期化（一度だけ実行すれば良い）
    if(gInitFlag == 1)
        return;
    try {
        var iflag = 2 + 0;      // 独自 XML を返す(標準 IC-CARD はチェックしない)
//        var iflag = 2 + 256;  // 標準 IC-CARD チェック
        var cards = "";   // オプション IC-CARD の指定無し
        var ret = csign.init(iflag, cards);
        if(!ret || ret.substr(0, 6) === "Error:") {
            // エラー
            ErrorAlert(csign, "初期化エラー", ret);
            return;
        }
    } catch (e) {
        ErrorAlert(csign, "例外エラー", e);
        return;
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
    }
    // 初期化成功
    gInitFlag = 1;
    alert("初期化完了¥n" + ret);
}
```

JavaScript 署名時の例：

```
// 署名実行
function SignMethod(xades, debug) {

    // 署名オブジェクト取得
    var csign = document.getElementById('LeCSignCapiA');
    if(!csign) {
        alert("署名処理：オブジェクト取得エラー");
        return;
    }
    // 初期化
    var host, cpath, spath;
    if(xades) {
        alert("[XAdES 署名処理開始]");
        xades = true;
        if(debug)
            host = "debug://LcsXAdES.exe";
        else
            host = "http://www.langedge.jp/cgi-bin/xades";
    } else {
        alert("[PAdES 署名処理開始]");
        xades = false;
        if(debug)
            host = "debug://LcsPAdES.exe";
        else
            host = "http://www.langedge.jp/cgi-bin/pades";
    }
    var sessionId = "AX0123";   // セッション ID（通常サーバセットだがここでは仮セット）
    var hopt = "";
    if(debug) {
        cpath = " -cert";
        spath = " -sign";
    } else {
        cpath = "/cert.cgi";
        spath = "/sign.cgi";
    }
    var header = "";
    var agent = "";
    var hflag = 0;
    // クライアント署名初期化
    InitLcs(csign);
    var status = csign.status();    // 証明書数を返す
    if(status < 1) {
        alert("証明書がありません。");
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
        return;
    }
    // 通信設定
    var ret = csign.httpSet(host, sessionId, hopt);
    if(ret != 0) {
        ErrorAlert(csign, "通信設定エラー", null);
        return;
    }
    // 署名処理
    try {
        //(Client) 証明書取得
        var cflag = 0;
        var copt = "";
        var cert = csign.cert(cflag, copt);
        if(!cert) {
            var lastError = csign.errorGet();
            if(lastError == 1) {
                // キャンセル
                alert("証明書取得キャンセル");
                return;
            }
            ErrorAlert(csign, "証明書取得エラー", null);
            return;
        }
//      alert("証明書取得 cert = ¥n" + cert);
        //(Server) 証明書をサーバに送信し署名対象データ取得
        var hashXml = csign.httpSend(cpath, header, cert, agent, hflag);
        if(!hashXml || hashXml.substr(0, 6) === "Error:") {
            // エラー
            ErrorAlert(csign, "証明書通信エラー", hashXml);
            return;
        }
        alert("署名 XML¥n" + hashXml);
        // ハッシュ値
        var hash = csign.xmlGet(hashXml, "/LcsResponse/hash");
        if(!hash) {
            // エラー
            ErrorAlert(csign, "ハッシュ値取得エラー", hash);
            return;
        }
        //(Client)署名値生成メソッド
        var signAlg = CheckSignAlg(hash);    // 署名アルゴリズム取得
        if(signAlg < 0) {
            alert("ハッシュサイズエラー：len= " + hash.length);
            return;
        }
        var sflag = 0;      // ハッシュ値の指定
        var sopt = "";
        var passwd = "";
        var sign = csign.sign(sflag, sopt, signAlg, hash, passwd);
        if(!sign) {
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
            ErrorAlert(csign, "署名値生成エラー", null);
            return;
        }
        alert("署名値生成 sign = ¥n" + sign);
        //(Server) 署名値データを送信
        var resp = csign.httpSend(spath, header, sign, agent, hflag);
        if(!resp || resp.substr(0, 6) === "Error:") {
            // エラー
            ErrorAlert(csign, "署名値通信エラー", resp);
            return;
        }
        //(Client)終了処理
        var redirect = csign.xmlGet(resp, "/LcsResponse/@redirect");
        csign.term(0, redirect);
        gInitFlag = 0;
    } catch (e) {
        ErrorAlert(csign, "例外エラー", e);
        return;
    }
    alert("[署名処理終了]");
}
```

JavaScript 補助メソッド：

```
// エラー表示
function ErrorAlert(csign, mesg, opt) {
    var lastError = csign.errorGet();
    var lastErrorAPI = csign.errorApi();
    var alertMesg = mesg + " :" + lastErrorAPI + " (" + lastError + ")";
    if(opt != null)
        alertMesg += "¥n" + opt;
    alert(alertMesg);
}
```

```
// ハッシュ値からハッシュ方式の取得
function CheckSignAlg(hexValue) {
    var signAlg = -1;   // ERROR
    if(hexValue.length == 40) {
        signAlg = 10;   // SHA-1 (20byte)
    } else if(hexValue.length == 64) {
        signAlg = 0;    // SHA-256 (32byte)
    } else if(hexValue.length == 96) {
        signAlg = 1;    // SHA-384 (48byte)
    } else if(hexValue.length == 128) {
        signAlg = 2;    // SHA-512 (64byte)
    }
    return signAlg;
}
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．４．２．LeCSignCapiA.cab / CAPI 版 一括 API の利用

実装サンプル：bin_activex/test-capi-exec.htm

ActiveX による CAPI を利用したクライアント署名プラグイン。Windows 証明書ストアの個人にある証明書と、CAPI に対応した IC カードや USB トークンにてクライアント署名が可能となる。

HTML 部の例：タイトルと署名ボタンのみ表示し param にて設定：

```
<body>
<p>LeClientSign: ActiveX CAPI Sample 1</p>
<p>[ EXEC API execution ]</p>
<hr/>
LOCAL debug: Need LcsXAdES.exe and LcsPAdES.exe pre-installed.
<p><button onclick="SignMethod('XLeCSignCapiA-D')">XAdES CAPI SIGN (Debug)</button></p>
<p><button onclick="SignMethod('PLeCSignCapiA-D')">PAdES CAPI SIGN (Debug)</button></p>
<hr/>
HTTP connection: Use LE Dummy Server.
<p><button onclick="SignMethod('XLeCSignCapiA')">XAdES CAPI SIGN (Remote)</button></p>
<p><button onclick="SignMethod('PLeCSignCapiA')">PAdES CAPI SIGN (Remote)</button></p>
<hr/>
<object id="XLeCSignCapiA-D" width="0" height="0"
  CODEBASE="./LeCSignCapiA.cab#Version=1,1,1,0"
  classid="CLSID:2DB6008C-CF18-49C4-9CDC-1075B34414BE">
  <param name="iflag" value=4096><!-- 4096=0x1000 はデバッグフラグ -->
  <param name="iopt" value="">
  <param name="sid" value="XID_00001">
  <param name="host" value="debug://LcsXAdES.exe">
  <param name="cpath" value=" -cert">
  <param name="spath" value=" -sign">
  <param name="salg" value=0>
  <param name="eflag" value=0>
  <param name="eopt" value="">
</object>
<object id="PLeCSignCapiA-D" width="0" height="0"
  CODEBASE="./LeCSignCapiA.cab#Version=1,1,1,0"
  classid="CLSID:2DB6008C-CF18-49C4-9CDC-1075B34414BE">
  <param name="iflag" value=4096><!-- 4096=0x1000 はデバッグフラグ -->
  <param name="iopt" value="">
  <param name="sid" value="PID_00001">
  <param name="host" value="debug://LcsPAdES.exe">
  <param name="cpath" value=" -cert">
  <param name="spath" value=" -sign">
  <param name="salg" value=0>
  <param name="eflag" value=0>
  <param name="eopt" value="">
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
</object>
<object id="XLeCSignCapiA" width="0" height="0"
  CODEBASE="./LeCSignCapiA.cab#Version=1,1,1,0"
  classid="CLSID:2DB6008C-CF18-49C4-9CDC-1075B34414BE">
  <param name="iflag" value=0>
  <param name="iopt" value="">
  <param name="sid" value="XID_00001">
  <param name="host" value="http://www.langedge.jp/cgi-bin/xades">
  <param name="cpath" value="/cert.cgi">
  <param name="spath" value="/sign.cgi">
  <param name="salg" value=0>
  <param name="eflag" value=0>
  <param name="eopt" value="">
</object>
<object id="PLeCSignCapiA" width="0" height="0"
  CODEBASE="./LeCSignCapiA.cab#Version=1,1,1,0"
  classid="CLSID:2DB6008C-CF18-49C4-9CDC-1075B34414BE">
  <param name="iflag" value=0>
  <param name="iopt" value="">
  <param name="sid" value="PID_00001">
  <param name="host" value="http://www.langedge.jp/cgi-bin/pades">
  <param name="cpath" value="/cert.cgi">
  <param name="spath" value="/sign.cgi">
  <param name="salg" value=0>
  <param name="eflag" value=0>
  <param name="eopt" value="">
</object>
</body>
```

JavaScript 署名実行の例：

```
// 署名実行
function SignMethod(name) {
    alert("[" + name + "署名処理開始]");

    // 署名オブジェクト取得
    var csign = document.getElementById(name);
    if(!csign) {
        alert("署名処理：オブジェクト取得エラー");
        return;
    }

    // 署名処理
    try {
        //(Client) 証明書取得 - 本サンプルは param で全てセット
        var rslt = csign.exec(-1, "", "", "", "", "", -1, -1, "");
        if(rslt == 1) {
            alert("キャンセルされました");
            return;
        }
        if(rslt < 0) {
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
            var lastError = csign.errorGet();
            var lastErrorAPI = csign.errorApi();
            var alertMesg = "署名エラー:" + lastErrorAPI + " (" + lastError + ")";
            alert(alertMesg);
            return;
        }
    } catch (e) {
        ErrorAlert(csign, "例外エラー", e);
        return;
    }
    alert("[署名処理終了]");
}
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．４．３．LeCSignP11A.cab / PKCS#11 版 個別 API の利用

実装サンプル：bin_activex/test-p11.htm

ActiveX による PKCS#11 ドライバを利用したクライアント署名プラグイン。PKCS#11 に対応した IC カードや USB トークンにてクライアント署名が可能となる。

HTML 部の例：タイトルと署名ボタンのみ表示

```
<body>
LeClientSign: ActiveX PKCS#11 Sample 1<p/>
<object id="LeCSignP11A" width="0" height="0"
  CODEBASE="./LeCSignP11A.cab#Version=1,1,1,1"
  classid="CLSID:5F494C03-3D02-4E8A-8211-FBD51E1DF5B0">
</object>
<button onclick="SignMethod()">SIGN</button>
</body>
```

JavaScript 初期化の例：

```
// 初期化処理
function InitLcs(csign) {
    // 初期化（一度だけ実行すれば良い）
    if(gInitFlag == 1)
        return;
    try {
        var iflag = 2;      // 独自 XML を返す
        var paths = "C:¥¥Windows¥¥System32¥¥onepin-opensc-pkcs11.dll" + "¥n"  // OpenSC
                  + "C:¥¥Program Files¥¥JPKI¥¥JPKIPKCS11Sign.dll";            // JPKI
        var ret = csign.init(iflag, paths);
        if(!ret || ret.substr(0, 6) === "Error:") {
            // エラー
            ErrorAlert(csign, "初期化エラー", ret);
            return;
        }
    } catch (e) {
        ErrorAlert(csign, "例外エラー", e);
        return;
    }
    // 初期化成功
    gInitFlag = 1;
    alert("初期化完了¥n" + ret);
}
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

JavaScript 署名時の例：

```
// 署名実行
function SignMethod(xades, debug) {

    // 署名オブジェクト取得
    var csign = document.getElementById('LeCSignP11A');
    if(!csign) {
        alert("署名処理：オブジェクト取得エラー");
        return;
    }

    // 初期化
    var host, cpath, spath;
    if(xades) {
        alert("[XAdES 署名処理開始]");
        xades = true;
        if(debug)
            host = "debug://LcsXAdES.exe";
        else
            host = "http://www.langedge.jp/cgi-bin/xades";
    } else {
        alert("[PAdES 署名処理開始]");
        xades = false;
        if(debug)
            host = "debug://LcsPAdES.exe";
        else
            host = "http://www.langedge.jp/cgi-bin/pades";
    }
    var sessionId = "AX0123";  // セッション ID（通常サーバセットだがここでは仮セット）
    var hopt = "";
    if(debug) {
        cpath = " -cert";
        spath = " -sign";
    } else {
        cpath = "/cert.cgi";
        spath = "/sign.cgi";
    }
    var header = "";
    var agent = "";
    var hflag = 0;

    // クライアント署名初期化
    InitLcs(csign);
    var status = csign.status();    // 証明書数を返す
    if(status != 1) {
        alert("IC カードを挿入してから再度署名ボタンをクリックしてください。");
        return;
    }

    // 通信設定
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
    var ret = csign.httpSet(host, sessionId, hopt);
    if(ret != 0) {
        ErrorAlert(csign, "通信設定エラー", null);
        return;
    }

    // 署名処理
    try {
        //(Client) 証明書取得
        var cflag = 0;
        var copt = "";
        var cert = csign.cert(cflag, copt);
        if(!cert) {
            var lastError = csign.errorGet();
            if(lastError == 1) {
                // キャンセル
                alert("証明書取得キャンセル");
                return;
            }
            ErrorAlert(csign, "証明書取得エラー", null);
            return;
        }
//      alert("証明書取得 cert = ¥n" + cert);

        //(Server) 証明書をサーバに送信し署名対象データ取得
        var hashXml = csign.httpSend(cpath, header, cert, agent, hflag);
        if(!hashXml || hashXml.substr(0, 6) === "Error:") {
            // エラー
            ErrorAlert(csign, "証明書通信エラー", hashXml);
            return;
        }
        alert("署名 XML¥n" + hashXml);
        // ハッシュ値
        var hash = csign.xmlGet(hashXml, "/LcsResponse/hash");
        if(!hash) {
            // エラー
            ErrorAlert(csign, "ハッシュ値取得エラー", hash);
            return;
        }

        //(Client)署名値生成メソッド
        var signAlg = CheckSignAlg(hash);    // 署名アルゴリズム取得
        if(signAlg < 0) {
            alert("ハッシュサイズエラー：len= " + hash.length);
            return;
        }
        var sflag = 0;      // ハッシュ値の指定
        var sopt = "";
        var passwd = "";
        var sign = csign.sign(sflag, sopt, signAlg, hash, passwd);
        if(!sign) {
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
            ErrorAlert(csign, "署名値生成エラー", null);
            return;
        }
        alert("署名値生成 sign = ¥n" + sign);

        //(Server) 署名値データを送信
        var resp = csign.httpSend(spath, header, sign, agent, hflag);
        if(!resp || resp.substr(0, 6) === "Error:") {
            // エラー
            ErrorAlert(csign, "署名値通信エラー", resp);
            return;
        }

        //(Client)終了処理
        var redirect = csign.xmlGet(resp, "/LcsResponse/@redirect");
        csign.term(0, redirect);
        gInitFlag = 0;

    } catch (e) {
        ErrorAlert(csign, "例外エラー", e);
        return;
    }
    alert("[署名処理終了]");
}
```

JavaScript 補助メソッド：

```
// エラー表示
function ErrorAlert(csign, mesg, opt) {
    var lastError = csign.errorGet();
    var lastErrorAPI = csign.errorApi();
    var alertMesg = mesg + " :" + lastErrorAPI + " (" + lastError + ")";
    if(opt != null)
        alertMesg += "¥n" + opt;
    alert(alertMesg);
}
```

```
// ハッシュ値からハッシュ方式の取得
function CheckSignAlg(hexValue) {
    var signAlg = -1;   // ERROR
    if(hexValue.length == 40) {
        signAlg = 10;   // SHA-1 (20byte)
    } else if(hexValue.length == 64) {
        signAlg = 0;    // SHA-256 (32byte)
    } else if(hexValue.length == 96) {
        signAlg = 1;    // SHA-384 (48byte)
    } else if(hexValue.length == 128) {
        signAlg = 2;    // SHA-512 (64byte)
    }
    return signAlg;
}
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．４．４．LeCSignP11A.cab / PKCS#11 版 一括 API の利用

実装サンプル：bin_activex/test-p11-exec.htm

ActiveX による PKCS#11 ドライバを利用したクライアント署名プラグイン。PKCS#11 に対応した IC カードや USB トークンにてクライアント署名が可能となる。

HTML 部の例：タイトルと署名ボタンのみ表示し param にて設定：

```
<body>
<p>LeClientSign: ActiveX PKCS#11 Sample 1</p>
<p>[ EXEC API execution ]</p>
<hr/>
LOCAL debug: Need LcsXAdES.exe and LcsPAdES.exe pre-installed.
<p><button onclick="SignMethod('XLeCSignP11A-D')">XAdES PKCS#11 SIGN (Debug)</button></p>
<p><button onclick="SignMethod('PLeCSignP11A-D')">PAdES PKCS#11 SIGN (Debug)</button></p>
<hr/>
HTTP connection: Use LE Dummy Server.
<p><button onclick="SignMethod('XLeCSignP11A')">XAdES PKCS#11 SIGN (Remote)</button></p>
<p><button onclick="SignMethod('PLeCSignP11A')">PAdES PKCS#11 SIGN (Remote)</button></p>
<hr/>
<object id="XLeCSignP11A-D" width="0" height="0"
  CODEBASE="./LeCSignP11A.cab#Version=1,1,1,1"
  classid="CLSID:5F494C03-3D02-4E8A-8211-FBD51E1DF5B0">
  <param name="iflag" value=4096><!-- 4096=0x1000 はデバッグフラグ -->
  <param name="iopt1" value="C:¥Windows¥System32¥onepin-opensc-pkcs11.dll">
  <param name="iopt2" value="C:¥Program Files¥JPKI¥JPKIPKCS11Sign.dll">
  <param name="sid" value="XID_00001">
  <param name="host" value="debug://LcsXAdES.exe">
  <param name="cpath" value=" -cert">
  <param name="spath" value=" -sign">
  <param name="salg" value=0>
  <param name="eflag" value=0>
  <param name="eopt" value="">
</object>
<object id="PLeCSignP11A-D" width="0" height="0"
  CODEBASE="./LeCSignP11A.cab#Version=1,1,1,1"
  classid="CLSID:5F494C03-3D02-4E8A-8211-FBD51E1DF5B0">
  <param name="iflag" value=4096><!-- 4096=0x1000 はデバッグフラグ -->
  <param name="iopt1" value="C:¥Windows¥System32¥onepin-opensc-pkcs11.dll">
  <param name="iopt2" value="C:¥Program Files¥JPKI¥JPKIPKCS11Sign.dll">
  <param name="sid" value="PID_00001">
  <param name="host" value="debug://LcsPAdES.exe">
  <param name="cpath" value=" -cert">
  <param name="spath" value=" -sign">
  <param name="salg" value=0>
  <param name="eflag" value=0>
  <param name="eopt" value="">
</object>
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
<object id="XLeCSignP11A" width="0" height="0"
  CODEBASE="./LeCSignP11A.cab#Version=1,1,1,1"
  classid="CLSID:5F494C03-3D02-4E8A-8211-FBD51E1DF5B0">
  <param name="iflag" value=0>
  <param name="iopt1" value="C:¥Windows¥System32¥onepin-opensc-pkcs11.dll">
  <param name="iopt2" value="C:¥Program Files¥JPKI¥JPKIPKCS11Sign.dll">
  <param name="sid" value="XID_00001">
  <param name="host" value="http://www.langedge.jp/cgi-bin/xades">
  <param name="cpath" value="/cert.cgi">
  <param name="spath" value="/sign.cgi">
  <param name="salg" value=0>
  <param name="eflag" value=0>
  <param name="eopt" value="">
</object>
<object id="PLeCSignP11A" width="0" height="0"
  CODEBASE="./LeCSignP11A.cab#Version=1,1,1,1"
  classid="CLSID:5F494C03-3D02-4E8A-8211-FBD51E1DF5B0">
  <param name="iflag" value=0>
  <param name="iopt1" value="C:¥Windows¥System32¥onepin-opensc-pkcs11.dll">
  <param name="iopt2" value="C:¥Program Files¥JPKI¥JPKIPKCS11Sign.dll">
  <param name="sid" value="PID_00001">
  <param name="host" value="http://www.langedge.jp/cgi-bin/pades">
  <param name="cpath" value="/cert.cgi">
  <param name="spath" value="/sign.cgi">
  <param name="salg" value=0>
  <param name="eflag" value=0>
  <param name="eopt" value="">
</object>
</body>
```

JavaScript 署名実行の例：

```
// 署名実行
function SignMethod(name) {
    alert("[" + name + "署名処理開始]");

    // 署名オブジェクト取得
    var csign = document.getElementById(name);
    if(!csign) {
        alert("署名処理：オブジェクト取得エラー");
        return;
    }

    // 署名処理
    try {
        //(Client) 証明書取得 - 本サンプルは param で全てセット
        var rslt = csign.exec(-1, "", "", "", "", "", -1, -1, "");
        if(rslt < 0) {
            var lastError = csign.errorGet();
            var lastErrorAPI = csign.errorApi();
            var alertMesg = "署名エラー：" + lastErrorAPI + " (" + lastError + ")";
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
            alert(alertMesg);
            return;
        }
    } catch (e) {
        ErrorAlert(csign, "例外エラー", e);
        return;
    }
    alert("[署名処理終了]");
}
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

２．４．５．XAdES と証明書情報の利用（V2.1 新機能）

実装サンプル：bin_activex/test-capi-xades.htm

ActiveX による CAPI を利用したクライアント署名プラグイン。Windows 証明書ストアの個人にある証明書と、CAPI に対応した IC カードや USB トークンにてクライアント側にて XAdES 署名が可能となる。

HTML 部の例：タイトルと署名ボタンを表示：

```
<body>
<h2>XAdES ローカル署名 [IE11/ActiveX 試験]</h2>
<hr/>
<p><b>試験実行：</b>
<!-- ボタンクリックで署名コマンドを実行(セッション ID を指定) -->
<input type="button" value="XAdES ローカル署名実行" onclick="execSign()">
<p><b>オプション：</b><br/>
  <input type="checkbox" id="jpki">マイナンバーカード/IC カード試験（オフの場合は証明書選択画面の利用）<br/>
</p>
<hr/>
<div><b>[証明書情報]</b></div>
<div id="CTYPE">種別：</div>
<div id="CTIME">期間：</div>
<div id="COPT1">オプション 1：</div>
<div id="COPT2">オプション 2：</div>
<div id="COPT3">オプション 3：</div>
<div id="COPT4">オプション 4：</div>
<hr/>
<object id="LeCSignCapiA" width="0" height="0"
  CODEBASE="./LeCSignCapiA.cab#Version=1,1,1,1"
  classid="CLSID:2DB6008C-CF18-49C4-9CDC-1075B34414BE">
</object>
</body>
```

JavaScript 署名実行の例：

```
<!-- LE:署名 API 定義 JS (※先に指定が必要) -->
<script src="LeCSignIE11.js"></script>

<!-- 署名個別実装 JS -->
<script type="text/javascript">

  // ============================================================
  // 署名処理実行(参考用の実装例).
  function execSign(){

    // ------------------------------------------------------
    // オプション 1：マイナンバーカード指定チェック
    var jpkitest  = false;
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
    const jpki = document.getElementById("jpki");
    if(jpki.checked) {
      console.log("試験：マイナンバーカード指定");
      jpkitest = true;
    }

    // ------------------------------------------------------------
    // 処理開始.
    var rc = 0;
    console.log('署名処理開始:');
    // 出力要素取得
    var CTYPE = document.getElementById("CTYPE");
    var CTIME = document.getElementById("CTIME");
    var COPT1 = document.getElementById("COPT1");
    var COPT2 = document.getElementById("COPT2");
    var COPT3 = document.getElementById("COPT3");
    var COPT4 = document.getElementById("COPT4");
    CTYPE.innerHTML = "";
    CTIME.innerHTML = "";
    COPT1.innerHTML = "";
    COPT2.innerHTML = "";
    COPT3.innerHTML = "";
    COPT4.innerHTML = "";

    // ------------------------------------------------------------
    // ブラウザ確認(現在 Chrome と新 Edge のみ利用可能).
    var agent = window.navigator.userAgent.toLowerCase();
    if(agent.indexOf('trident') == -1) {
      alert('未サポートのブラウザです。¥nIE11 を使ってください。');
      return;
    }
    console.log('ブラウザ確認 OK');

    // ------------------------------------------------------------
    // 署名オブジェクト取得
    var csign = document.getElementById('LeCSignCapiA');
    if(!csign) {
        alert("署名処理：オブジェクト取得エラー");
        return;
    }
    console.log('オブジェクト取得 OK');

    // ------------------------------------------------------------
    // 署名コマンド生成/初期化(必須).
    var initflag = LCSI_FLAG.LCSI_NO_FLAG;        // 初期化フラグ(デフォルト CAPI)
    if(jpkitest == true) {
      // マイナンバーカード
      initflag  |= LCSI_FLAG.LCSI_CAPI_NOCHECK;  // CAPI:証明書チェック無し(LCSC_CAPI_JPKI
利用の為)
    } else {
      // 証明書選択
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
      initflag  |= LCSI_FLAG.LCSI_CAPI_ICCARD;   // CAPI:標準 IC カード利用
    }
//  initflag    |= LCSI_FLAG.LCSI_SET_DEBUG;     // デバッグモード指定
    var initopt1 = "";                           // 初期化オプション 1(PKCS#11 の DLL パス等、
省略時空文字、null 指定不可)
    var ret = csign.init(initflag, initopt1);
    rc = csign.errorGet();
    if(!ret || rc < 0) {
      // エラー
      alert('エラー(' + rc + '):初期化エラーです。');
      return;
    }
    console.log("初期化 OK");

    // ------------------------------------------------------
    // 証明書設定
    var certdata  = null;                        // 証明書データ
    var certopt   = "";                          // 証明書選択オプション(省略時空文字、null
指定不可)
    var certflag  = LCSC_FLAG.LCSC_NO_FLAG;   // 証明書選択フラグ(デフォルト=0x00000000)
//  certflag    |= LCSC_FLAG.LCSC_CONFIRM;   // デバッグ:証明書 1 つの場合も確認
    if(jpkitest == true) {

      // マイナンバーカード(標準 IC カードにマイナンバーカードも含まれる)
      certflag  |= LCSC_FLAG.LCSC_CAPI_ICCARD;  // CAPI:標準 IC カードチェック
      alert('IC カードをセットしてください。¥n ご注意:PIN 入力画面が裏に表示される場合があ
ります。');
      while(true) {
        // 証明書取得コマンド(cert)実行:IC カードのチェックと取得
        certdata = csign.cert(certflag, certopt);
        rc = csign.errorGet();
        if(rc == 0 && certdata != null)
          break;                          // 取得できたのでループを終了
        // 取得できなかった
        if(rc == -11) {
          // IC カードが見つからない(繰り返し)
          var res = confirm('IC カードをセットしてください。');
          if(res == false) {
            // キャンセル
            alert('IC カードチェックがキャンセルされました。');
            return;
          }
        } else {
          // その他エラー
          alert('エラー(' + rc + '):証明書取得エラーです。');
          return;
        }
      }

    } else {
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
      // 証明書取得コマンド(cert)実行：通常の証明書選択
      certdata = csign.cert(certflag, certopt);
      rc = csign.errorGet();
      if(rc != 0) {
        // エラー
        if(rc == 1) {
          // キャンセル
          alert('証明書選択ダイアログがキャンセルされました。');
        } else {
          // その他エラー
          if(certdata == null) {
            alert('エラー(' + rc + ')：証明書取得エラーです。');
          } else {
            alert('エラー(' + rc + ')：証明書取得エラーです。¥n' + certdata);
            console.log('証明書取得エラー:¥n' + certdata);
          }
        }
        return;
      }

    }
    console.log("証明書取得 OK");

    // ------------------------------------------------------------
    // 証明書情報取得設定
    var infoflag   = LCSF_FLAG.LCSF_NO_FLAG;  // 結果を JSON 形式で返す(デフォルト)
//  infoflag       |= LCSF_FLAG.LCSF_RETXML;   // 結果を XML 形式で返す
    // 証明書情報取得(info)実行
    var certinfo = csign.info(certdata, infoflag);
    rc = csign.errorGet();
    if(rc < 0) {
      if(certinfo == null) {
        alert('エラー(' + rc + ')：証明書情報取得エラーです。');
      } else {
        alert('エラー(' + rc + ')：証明書情報取得エラーです。¥n' + certinfo);
        console.log('証明書情報取得エラー:¥n' + certinfo);
      }
      return;
    }
    console.log("証明書情報取得 OK");

    // ------------------------------------------------------------
    // 結果の解析
    var certobj = JSON.parse(certinfo);
    // 期間
    var ctime = "期間：";
    if(certobj.from != null)
      ctime += certobj.from;
    ctime += "～";
    if(certobj.to != null)
      ctime += certobj.to;
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
    // 種別
    if(certobj.jpki != null) {
      CTYPE.innerHTML = "種別：マイナンバーカード(JPKI)";
      CTIME.innerHTML = ctime;
      COPT1.innerHTML = "氏名：" + certobj.jpki.name;
      COPT2.innerHTML = "性別：" + certobj.jpki.gender;
      COPT3.innerHTML = "生年月日：" + certobj.jpki.birth;
      COPT4.innerHTML = "住所：" + certobj.jpki.address;
    } else if(certobj.regist != null) {
      CTYPE.innerHTML = "種別：商業登記証明書";
      CTIME.innerHTML = ctime;
      COPT1.innerHTML = "法人名：" + certobj.regist.corp;
      COPT2.innerHTML = "代表者：" + certobj.regist.name;
      COPT3.innerHTML = "肩書：" + certobj.regist.title;
      COPT4.innerHTML = "住所：" + certobj.regist.address;
    } else {
      CTYPE.innerHTML = "種別：一般証明書";
      CTIME.innerHTML = ctime;
      var info = "発行：";
      if(certobj.issuer.cn != null)
        info += certobj.issuer.cn;
      else if(certobj.issuer.ou != null)
        info += certobj.issuer.ou;
      else if(certobj.issuer.o != null)
        info += certobj.issuer.o;
      else
        info += "CN/OU/O 無し";
      COPT1.innerHTML = info;
      COPT2.innerHTML = "";
      COPT3.innerHTML = "";
      COPT4.innerHTML = "";
    }
    console.log("証明書情報セット OK");

    // ------------------------------------------------------------
    // 署名設定
        var orgxml  = "<LeTest><Contents Id=¥"TEST01¥">";
        orgxml     += "<Data>Sign Data</Data>";
        orgxml     += "<日本語>試験です</日本語>";
        orgxml     += "</Contents></LeTest>";   // 署名対象 XML
    var xflag    = 0x00000000;  // XAdES フラグを指定(デフォルト)
    var signAlg  = 0;           // 署名アルゴリズム(SHA256=0/SHA384=1/SHA512=2/SHA1=10)
    var xml      = orgxml;              // 署名対象 XML
    var id       = "TEST01";    // 署名対象を ID 指定(Enveloped の場合は空文字、null 指定不
可)
    var xpath    = "";          // 署名対象の XPATH 指定(不要時には空文字、null 指定不可)
    var passwd   = "";          // 署名時パスワード指定(PKCS#11 利用時に PIN 指定、省略時空
文字、null 指定不可)
    // XAdES-BES 生成コマンド(xades)実行
    var xadesXml = csign.xades(xflag, signAlg, xml, id, xpath, passwd);
    rc = csign.errorGet();
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
    if(rc != 0) {
      if(xadesXml == null)
        alert('エラー(' + rc + '):XAdES-BES 生成エラーです。');
      else
        alert('エラー(' + rc + '):' + xadesXml);
      return;
    }
    console.log("XAdES 生成 OK");

//      saveXml("xades.xml", xadesXml);

    // ------------------------------------------------------------
    // 通信設定
    var host = "https://dev.langedge.jp/";
    var sessionId = "DUMMY";
    var hopt = 0x00000000;  // HTTP 通信フラグ
    var ret = csign.httpSet(host, sessionId, hopt);
    if(ret != 0) {
        alert("HTTP 通信設定エラー");
        return;
    }
    // ------------------------------------------------------------
    // 結果のアップロード(/signcmd/up は送信した内容をそのまま返す)
    var spath = "tom/signcmd/up";      // 連携ホスト.
    var head  = "";         // オプションヘッダ指定
    var agent  = "";         // オプションヘッダ指定
    var hflag = 0x00000000;  // HTTP 通信フラグ
    var resp = csign.httpSend(spath, head, xadesXml, agent, hflag);
    rc = csign.errorGet();
    if(!resp || rc < 0) {
        // エラー
        alert("HTTP 通信エラー");
        return;
    }
    console.log("XAdES アップロード OK");

    // ------------------------------------------------------------
    // 結果の保存（表示）
    alert(resp);         // 仮：とりあえず xades データを alert 表示

    // ------------------------------------------------------------
    // 処理正常終了
    console.log('署名処理終了: OK');

  }

</script>
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．クライアント側実装：署名コマンド（LE:Client:Sign）

カスタム URL スキームを利用した署名コマンド（LeCSignCmd）はクライアント署名 V2 からサポートされた機能である。署名コマンドは ActiveX プラグインと違い外部プロセスとして実行される。署名コマンドは実行後にブラウザ側の JavaScript と WebSocket 通信することで同期実行される。

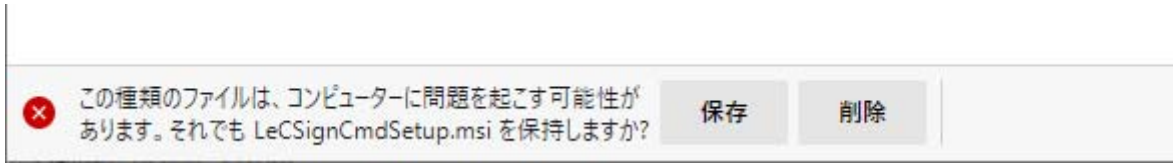
３．１．署名コマンドモジュール

機能	モジュール	説明	配置
JavaScript API	LeCSignCmdWS.js	LeCSignCmdWS クラス提供。署名コマンド実行後 WebSocket 通信接続。署名コマンドの未インストール検出可能。	サーバ上に配置
署名コマンド本体	LeCSignCmd.exe	カスタム URL スキームにより LeCSignCmd クラスから呼び出される。	インストーラで配置
インストーラ	LeCSignCmdSetup.zip LeCSignCmdSetup.msi	ダウンロード用の ZIP ファイルとインストーラ本体の MSI ファイルを提供 LeCSignCmd.exe の配置と、レジストリへの登録を行う。	ZIP ファイルをサーバ上に配置

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．１．１．署名コマンドインストーラ

署名コマンドインストーラは LeCSignCmdSetup.msi ファイルとなります。これを Edge/Chrome で直接ダウンロードしようとすると警告が表示され簡単にはインストールできません。この為に ZIP ファイルの LeCSignCmdSetup.zip としてダウンロードするようにします。



非推奨：msi ファイルのダウンロード時の警告（画面下部）

推奨：zip ファイルのダウンロード時の画面（画面下部）

署名コマンドインストーラは LeCSignCmdSetup.msi ファイルがインストールするファイルとレジストリは以下となります。

○ インストールファイル

インストールファイル	LeCSignCmd.exe
インストールフォルダ（標準）	C:¥Program Files (x86)¥LangEdge¥LeCSignCmd

○ インストールレジストリ

Registory Path	Name	Value
HKEY_CLASSES_ROOT	---	---
lcsign01	(Default)	"URL:LE Client Sign Protocol"
	URL Protocol	""
DefaultIcon	(Default)	"[TARGETDIR]LeCSignCmd.exe,0"
sheel¥open¥command	(Default)	""[TARGETDIR]LeCSignCmd.exe" %1"

※［TARGETDIR］はインストール先ディレクトリ

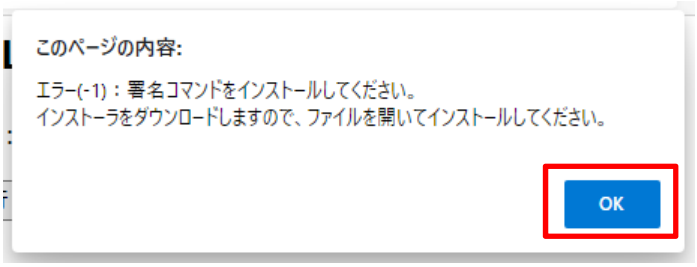
Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．１．２．署名コマンドのインストール手順

１）未インストール時に LeCSignCmdSetup.zip をダウンロードさせます。

```
rc = await cscmd.connect(port, waitSec);
if(rc == LSCERR.CMD_NOT_REGIST) {
  // 署名コマンドが未登録(未インストール).
  alert('インストーラをダウンロードして、ファイルを開いてインストールしてください。');
  window.location.href = "./LeCSignCmdSetup.zip";
}
```

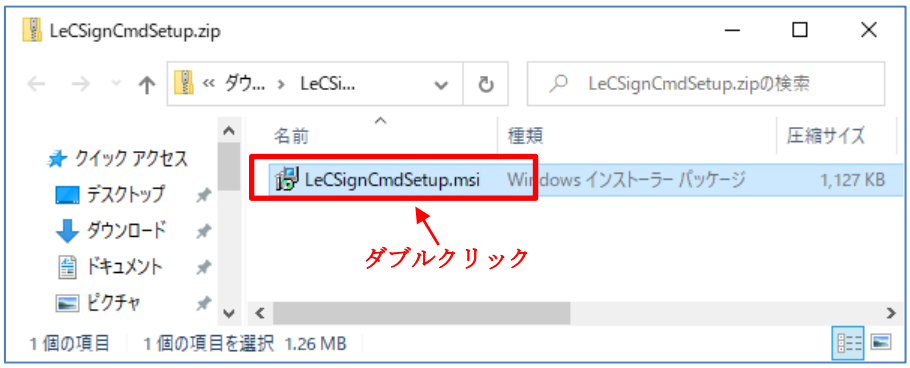
ダウンロード JavaScript 例



ダウンロード alert

２）LeCSignCmdSetup.zip をダウンロードして「ファイルを開く」を実行。

zip ファイルのダウンロード時の画面（ブラウザ画面下部）

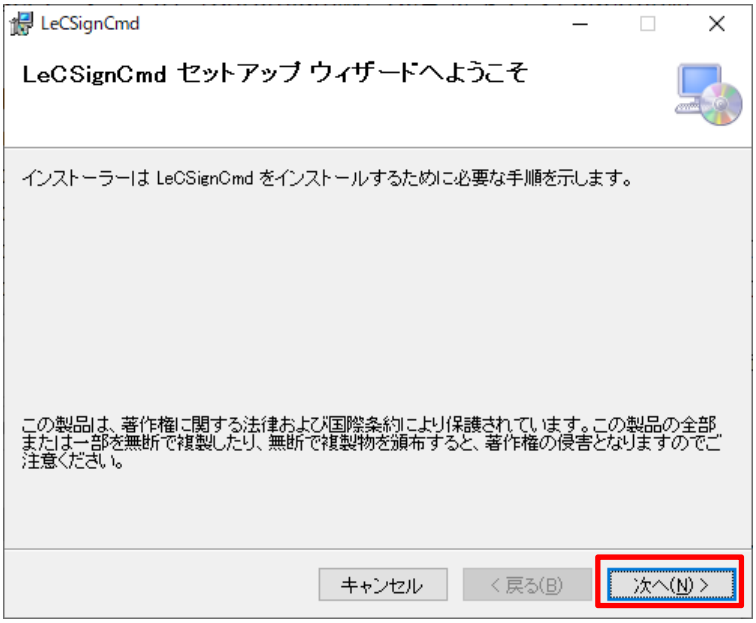


開かれたエクスプローラ画面

- 「常に許可する」を ON にした後にアンインストールした場合には CMD_NOT_REGIST は返らない。その場合は別途ダウンロードしてください。「3.2.3. 署名コマンド実行の判定」を参照。

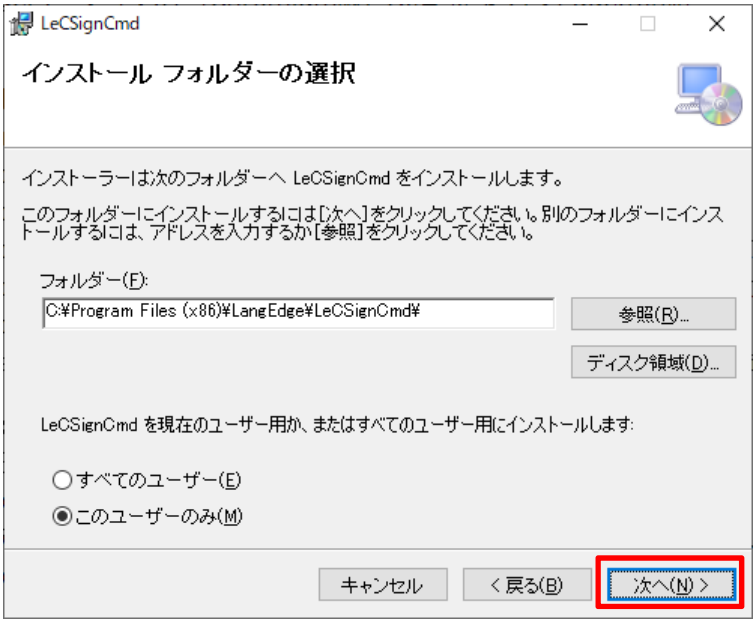
Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

3）インストーラが実行される。



インストーラ起動画面

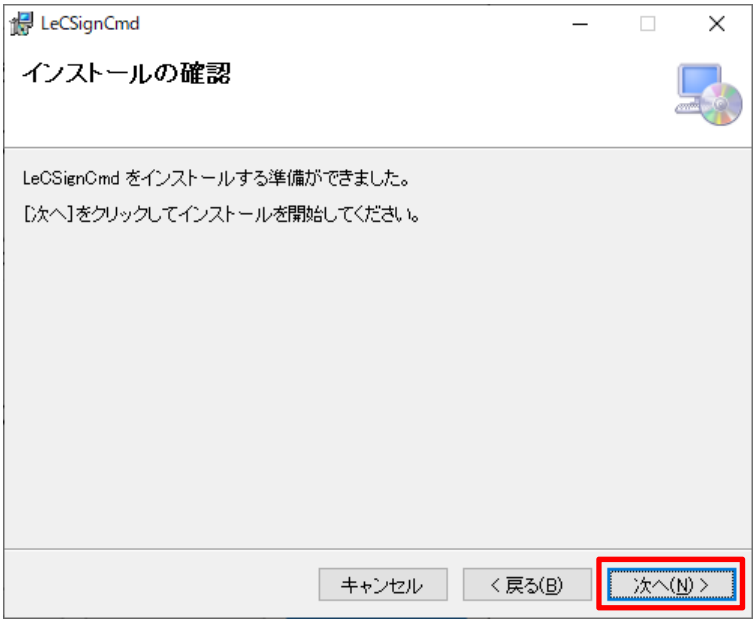
4）インストール フォルダーの選択 でフォルダを選択する。（推奨：標準のまま）



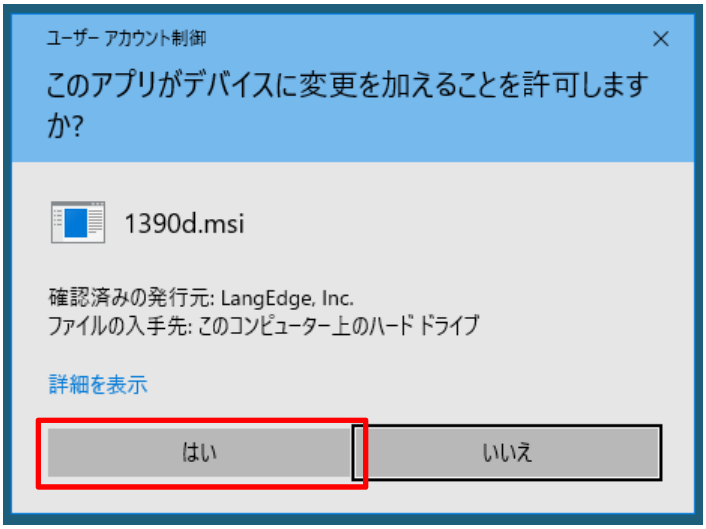
インストール フォルダーの選択 画面

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

５）インストールの確認をしてインストールを実行する。



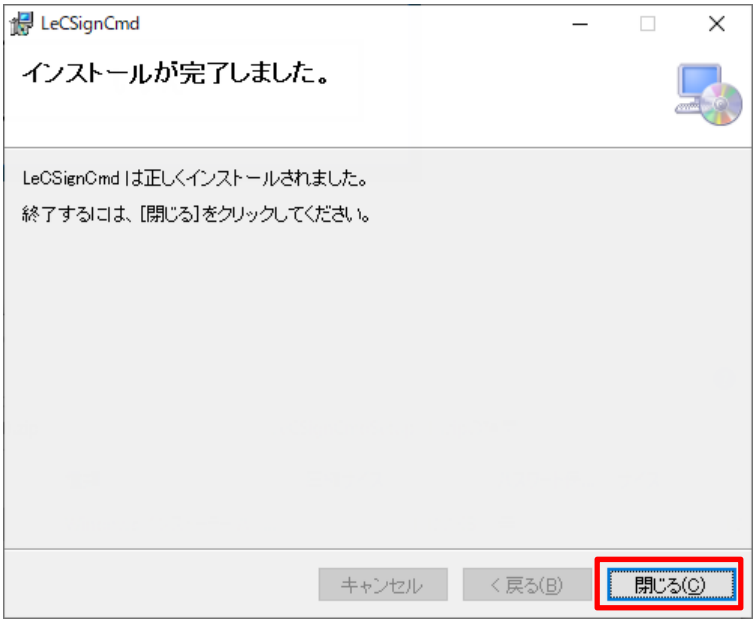
インストールの確認 画面



ユーザアカウント制御 のアクセス許可画面

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

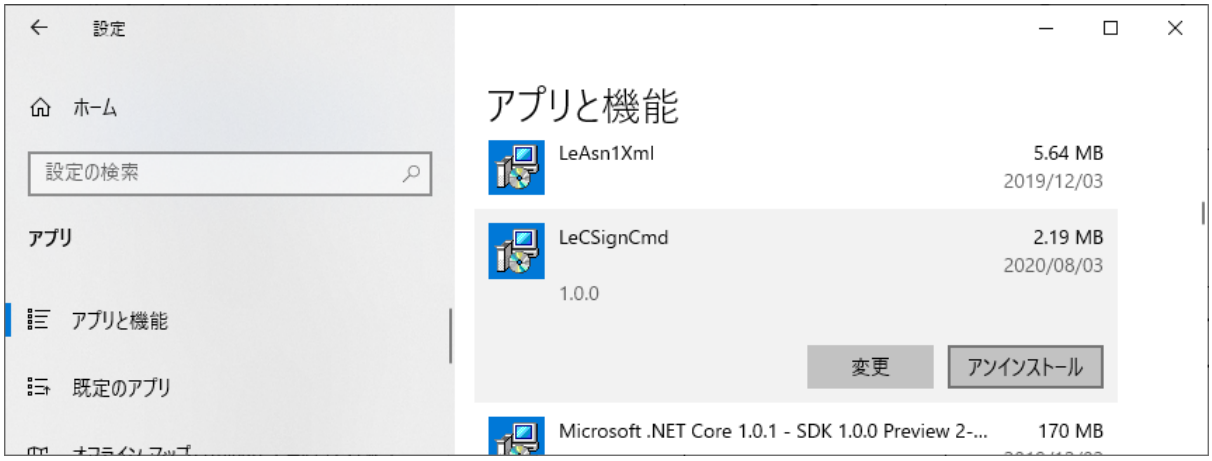
6）インストールが完了します。



インストールの完了 画面

3．1．3．署名コマンドのアンインストール手順

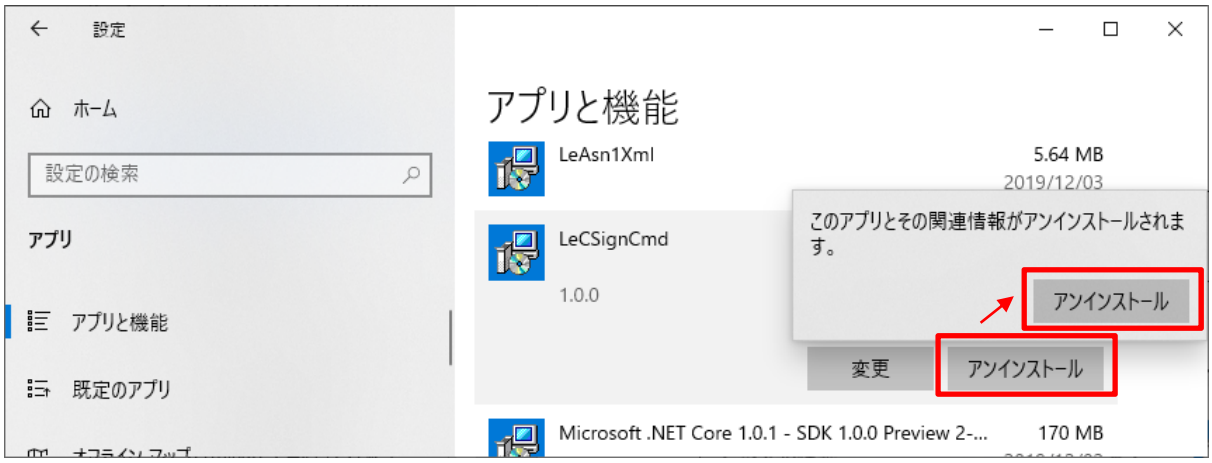
1）設定の「アプリと機能」画面を開いて「LeCSignCmd」を探す。



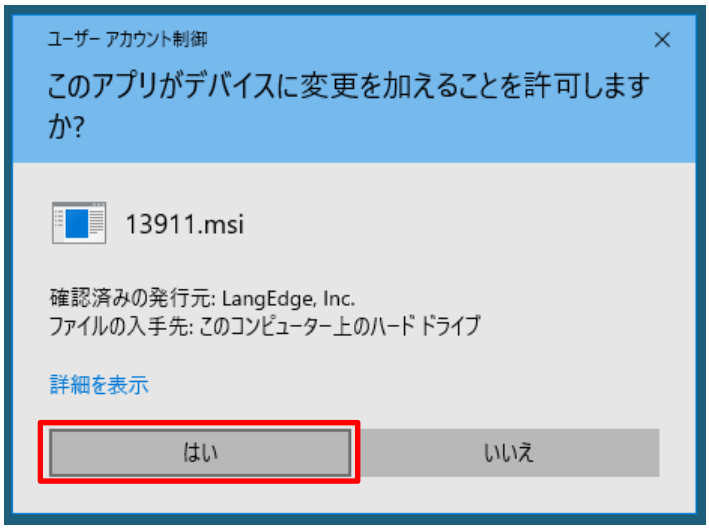
設定のアプリと機能 画面

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

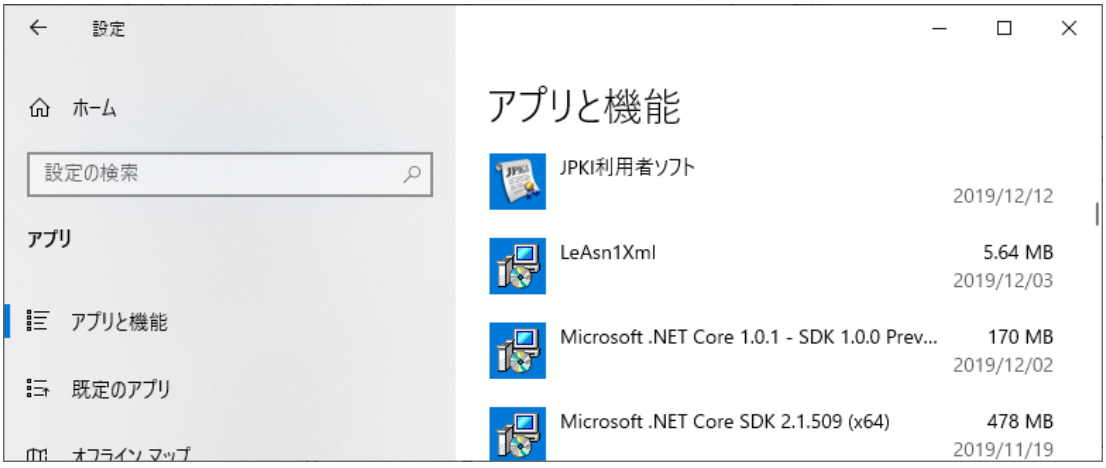
2）「LeCSignCmd」の「アンインストール」を実行する。



アプリと機能 画面からアンインストールを実行



ユーザアカウント制御 のアクセス許可画面



アンインストール後の アプリと機能 画面

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．２．署名コマンド利用シーケンス

署名コマンドでは実行時に自動インストールが行えない為に、事前にインストーラを使ってインストール（exe ファイルの配置とレジストリ登録）を行っておく必要がある。署名コマンドを利用するには JavaScript ライブラリ（LeCSignCmdWS.js）を経由する。LeCSignCmdWS クラスを生成/初期化して connect により署名コマンドを外部プロセスとして実行し、WebScoket 通信の接続を行う。それ以後は WebScoket 通信により同期的に API を実行できる。署名コマンドは実行時に最前面にあるブラウザを親ウィンドウとしてモーダルウィンドウを表示する。これにより実行中のブラウザ操作を禁止している。

署名コマンドでクライアント署名を利用するには大きく接続（実行と接続）・証明書取得（署名ハッシュ値計算）・署名値計算（署名値埋め込み）・終了の 4 処理が必要になるが、その間にサーバとの HTTP 通信と、XML 等による通信内容記述（セッション ID と情報の受け渡し）が必要となる。HTTP 通信と通信内容はカスタマイズが必要なケースもあるだろう。また本クライアント署名においては、HTTP 通信と XML 解析の API を個別提供しているので組み合わせて利用が可能となっている。また独自の通信 XML も用意しており、これら独自の通信手順を 1 つの一括実行 API で利用することも可能となっている。

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

サーバ連携の署名時の手順：

処理内容	個別 API	一括 API	処理詳細（一括）
実行と接続	**connect**		**署名コマンドの実行と接続（※必須）**
一括署名実行	↓	**exec**	**一括署名とサーバ通信の実行（※必須）**
証明書取得	**cert**	↓	**証明書の取得（※必須）**
証明書 HTTP 通信	(http)	↓	一括 API では独自 XML の送信/取得（オプション）
署名値計算	**sign**	↓	**署名値の計算（※必須）**
署名値 HTTP 通信	(http)	↓	一括 API では独自 XML の送信/取得（オプション） ※ 応答としてリダイレクト先の取得が可能。
終了	**terminate**		**署名コマンドの終了と接続開放（※必須）** ※ 本 API を実行しないと署名コマンドは終了しない。

クライアント側の処理手順概要（署名コマンド）

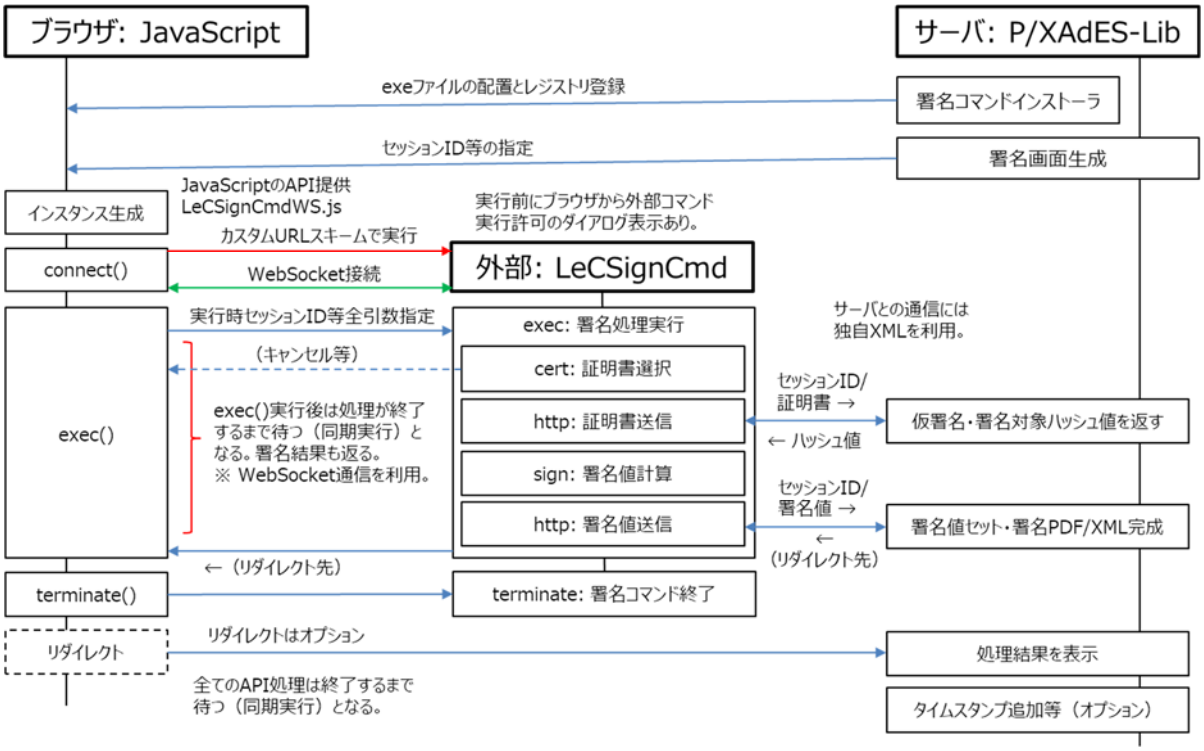
ローカルでの XAdES-BES 生成時の手順（Ver2.1 以降）：

処理内容	個別 API	処理詳細（一括）
実行と接続	**connect**	**署名コマンドの実行と接続（※必須）**
証明書取得	**cert**	**証明書の取得（※必須）**
証明書情報取得	(info)	証明書情報の取得（オプション）
XAdES-BES 対象追加	(xadd)	Detached/Enveloping 署名対象の追加（オプション）
XAdES-BES 署名	**xades**	**署名値の計算と XAdES-BES の生成（※必須）**
XAdES-BES 送信	(http)	署名済み XAdES-BES 情報の送信（オプション） ※ 別の方法で取得するなら通信は不要
終了	**teminate**	**署名コマンドの終了と接続開放（※必須）** ※ 本 API を実行しないと署名コマンドは終了しない。

クライアント側の処理手順概要（署名コマンド）

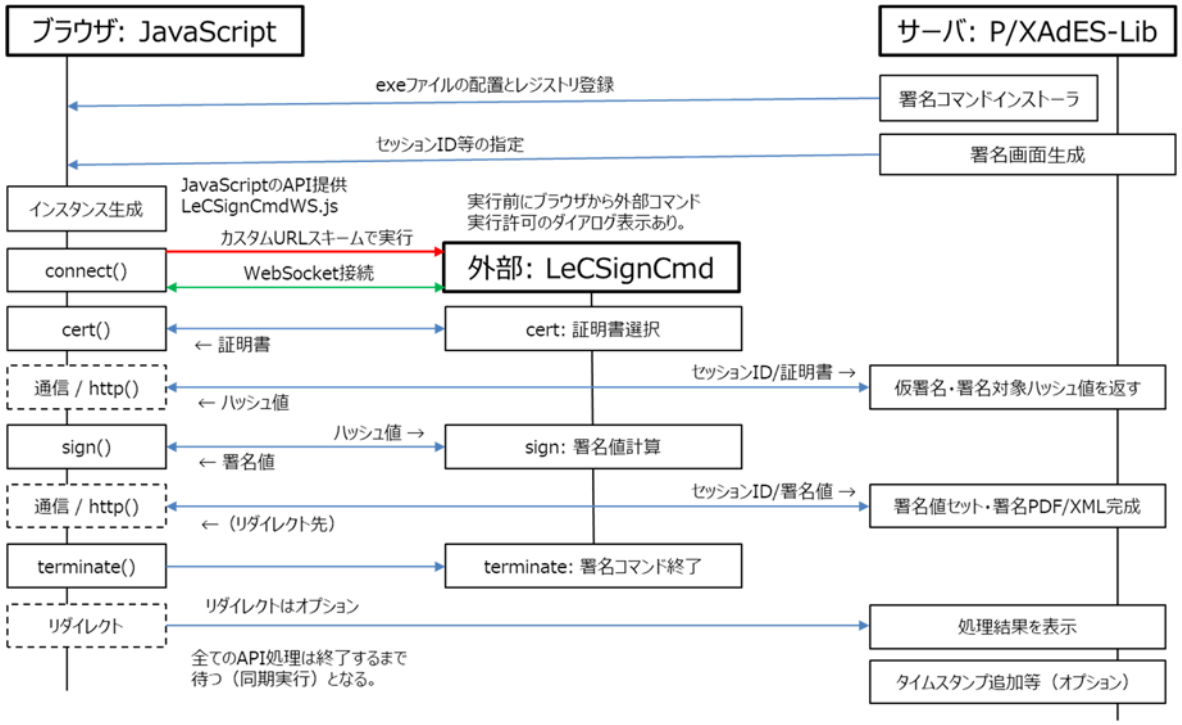
Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．２．１．一括 API 利用シーケンス



署名コマンド 一括 API 利用シーケンス図

３．２．２．個別 API 利用シーケンス



署名コマンド 個別 API 利用シーケンス図

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．２．３．署名コマンド実行の判定

署名コマンドの実行には connect の API を利用する。connect では「署名コマンド実行」と「実行された署名コマンドとの WebSocket 接続」の 2 つが実行される。

2020 年 10 月に Edge/Chrome のフォーカス移動の仕様変更があり、2020 年 8 月にリリースした署名コマンド実行の判定が使えなくなった。具体的な仕様変更の内容は、alert や実行確認時にはフォーカスアウトのイベントが送信されない仕様になった点となる。

新仕様においては、署名コマンドの実行に成功した場合のみフォーカスアウトによりインストール済みが確認できるが、署名コマンドが未インストール時と署名コマンドの実行エラーの区別が付かない。新仕様の対応は LeCSignCmdWS.js の入れ替えのみであり、署名コマンド自体はそのまま利用できる。

3.3.2. connect の実行時の状態		Ver 2.00.R2 2020 年 10 月以前	Ver 2.00.R2a 2020 年 10 月以降 新仕様
未インストール		すぐ CMD_NOT_REGIST	20 秒後に CMD_NOT_REGIST
インストール済みかつ実行確認ダイアログ表示	20 秒以内に OK をクリック	OK 後すぐ NO_ERR	OK 後すぐ NO_ERR
	キャンセルをクリックか、20 秒以上経過後に OK をクリック	20 秒後かクリック後に CMD_CONNECT_ERR	20 秒後か OK クリック後に **CMD_NOT_REGIST** ※ エラーと区別が付かない
インストール済みかつ常に許可済み		すぐ NO_ERR	すぐ NO_ERR

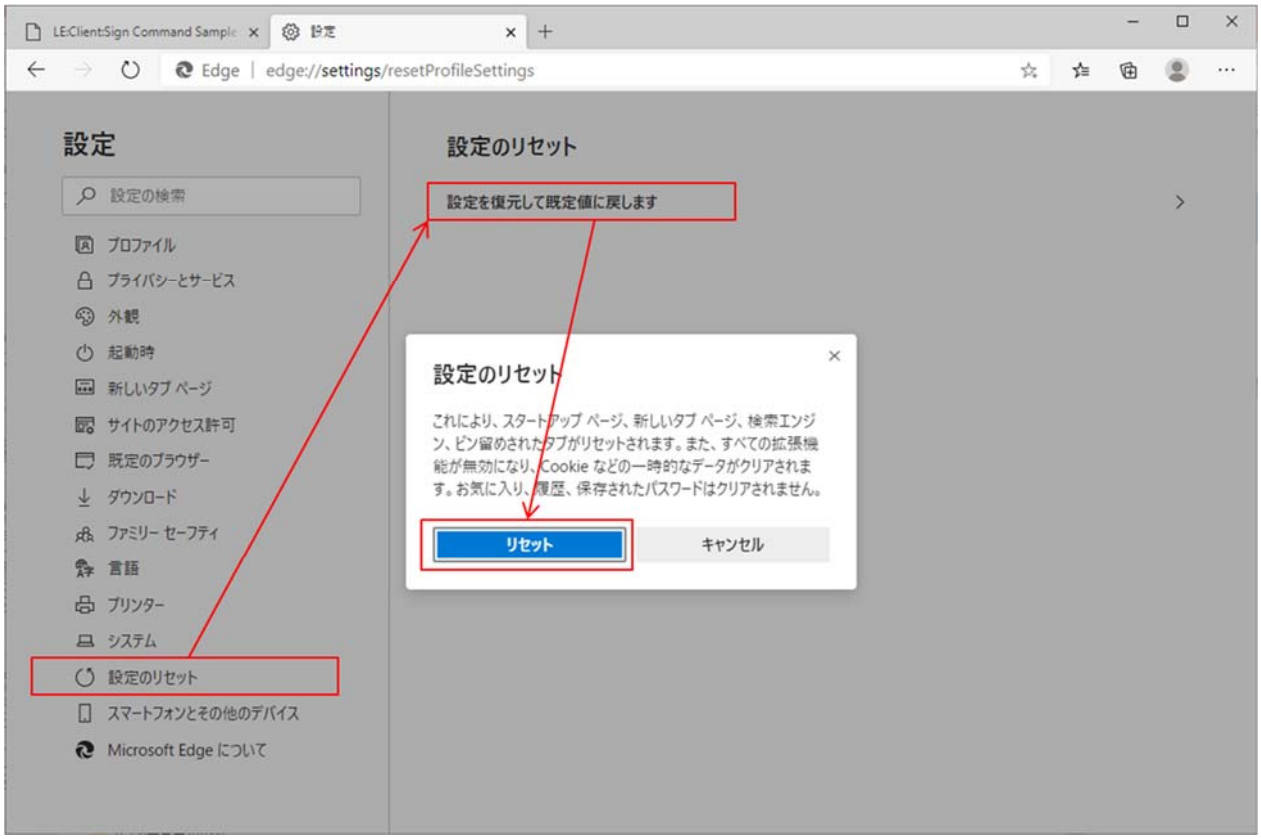
3.3.3. check の実行時の状態		Ver 2.00.R2 2020 年 10 月以前	Ver 2.00.R2a 2020 年 10 月以降 新仕様
未インストール		すぐ CMD_NOT_REGIST	5 秒後に CMD_NOT_REGIST
インストール済みかつ実行確認ダイアログ表示	5 秒以内に OK をクリック	OK 後すぐ NO_ERR	OK 後すぐ NO_ERR
	キャンセルをクリックか、5 秒以上経過後に OK をクリック	5 秒後かクリック後に CMD_CONNECT_ERR	5 秒後か OK クリック後に **CMD_NOT_REGIST** ※ エラーと区別が付かない
インストール済みかつ常に許可済み		すぐ NO_ERR	すぐ NO_ERR

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

「実行確認」時に「関連付けられたアプリでこの種類のリンクを開くことを常に許可する」チェックボックスを ON にして実行すると、次回から「実行確認」のダイアログは表示されず、署名コマンドの実行によりフォーカスアウトする。「常に許可する」状態の解除に関しては次ページにて。

「関連付けられたアプリでこの種類のリンクを開くことを常に許可する」を一度 ON にすると、その情報はブラウザ（Edge/Chrome）の設定ファイルに保存される。この為に署名コマンドのアンインストールではこの状態を解除することができない。以下に解除方法について説明する。

1）設定画面から解除する：設定画面からリセット



2）設定ファイルを編集して個別解除する

場所：　C:¥Users¥[ユーザ名]¥AppData¥Local¥Microsoft¥Edge¥User Data¥Default
ファイル名：　Preferences
変更：　テキストエディタでカスタム URL スキーム名を検索してフラグを false に変更

```
{"allowed_origin_protocol_pairs":{"file://":{"lcsign01":true}}}
```

↓以下に変更

```
{"allowed_origin_protocol_pairs":{"file://":{"lcsign01":false}}}
```

※ file:// の部分はサイト名となる。

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．３．署名コマンド利用 API

メソッド名	必須	署名コマンド	機能	説明/補足
constructor ()	○	起動前	生成/初期化	署名コマンド API クラス LeCSignCmdWS の生成と初期化設定を行う
connect ()	○	接続	署名コマンド実行と WebSocket 接続	署名コマンドを実行して WebSocket による通信を接続する
check ()	－	接続	署名コマンド実行のみを行う	署名コマンドを実行してインストール済みかどうかの確認を行う
cert ()	○	API 接続	証明書取得	証明書を選択して取得する
info ()	－	API 接続	証明書情報取得	証明書を解析して情報を取得する ※ Ver2.1 よりサポート
sign ()	△	API 接続	署名値計算	署名値を計算する ※ sign() 利用時には xades() は使わない
xades ()	△	API 接続	XAdES-BES 生成	指定された対象に XAdES-BES 署名を付与して生成する ※ xades() 利用時には sign() は使わない ※ Ver2.1 よりサポート
terminate ()	○	API 接続	署名コマンド終了	署名コマンドを終了する ※ terminate() を実行しないと終了しない
errorGet ()	－	利用しない（JS API）	エラー値取得	cert/sign 等 API のエラー値を取得する
status ()	－	API 接続	IC カード状態取得	IC カードの状態を取得する（接続を待つ）
version ()	－	API 接続	バージョン情報取得	署名コマンドのバージョン情報を取得する
http ()	－	API 接続	HTTP/HTTPS 通信	HTTP/HTTPS 通信を実行する
pxml ()	－	API 接続	独自 XML 解析	サーバ応答の独自 XML を解析する
exec ()	○	API 接続	一括実行	cert/sign と HTTP/HTTPS 通信を一括実行する

クライアント署名－署名コマンド API 一覧

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．３．１．constructor：生成/初期化

機能		constructor：LeCSignCmdWS クラスのコンストラクタ：初期設定の指定		
JSAPI 仕様		constructor (initflag, initopt1, initopt2, sessionid);		
引数	**1**	数値	initflag	初期化フラグ：署名コマンドに渡されるフラグ 0x00000000 LCSI_NO_FLAG オプション無し(CAPI) 0x00000001 LCSI_P11_USE PKCS#11 利用(initopt 必須) 0x00000004 LCSI_SET_RETHEX HEX 文字列で結果を返す 0x00000008 LCSI_SET_RETB64 Base64 文字列で結果を返す 0x00001000 LCSI_SET_DEBUG デバッグモード指定 0x00002000 LCSI_SET_LOGFILE ログファイル(initopt2) 0x00004000 LCSI_SET_LOGADD ログファイル追記 0x00008000 LCSI_SET_WININET WinInet 利用 (V2.2) ※ デフォルト設定では WinInet ではなく WinHTTP を利用
	2	文字列	initopt	オプション：指定しない場合は null で良い PKCS#11：優先利用の DLL ファイルのパスを指定可能 CAPI：追加の IC カード設定を指定可能「CAPI 補足」 ログファイル：指定時は initflag=0x00000 (CAPI 時のみ) ※ 複数指定時は改行コード[¥n]を利用して繋げる
	3	文字列	initopt2	initflag に LCSI_SET_LOGFILE セット時にログファイルのパスを指定する（省略時は null 指定）
	4	文字列	sessionid	独自 XML 通信（4.1 参照）用セッション ID を指定（省略時は null 指定）
戻り値		LeCSignCmdWS クラスのインタンスを返す		
PKCS#11 補足		初期化フラグ： 0x00000010 LCSI_P11_FILEDLG ファイルダイアログで dll 指定 オプション（PKCS#11 ドライバの DLL 指定）： ※ initopt に最低限１つ以上の DLL ファイルパスの指定が必要（NULL 不可）		
CAPI 補足		初期化フラグ： 0x00000100 LCSI_CAPI_ICCARD 標準 IC カード(JPKI/特定 CA 等)のチェックも行う 0x00000200 LCSI_CAPI_NOSTORE 個人 Windows 証明書ストアをチェックしない 0x00000400 LCSI_CAPI_CHKEKU 認証用と S/MIME 用を除外(拡張キー用途) ※ 0x00000800 LCSI_CAPI_NOCHECK 証明書チェック無し(LCSC_CAPI_ICCARD と利用) ※LCSI_CAPI_CHKEKU では以下の拡張キー用途の証明書を除外する（V2.1.R2 から） サーバ認証 ：OID=1.3.6.1.5.5.7.3.1 クライアント認証：OID=1.3.6.1.5.5.7.3.2 電子メールの保護：OID=1.3.6.1.5.5.7.3.4 (S/MIME 署名用) オプション（追加 IC カード設定の指定）： var cards = "24:-1:0:JPKI Crypto Service Provider for Sign";		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

V2.1 補足	➢ LCSI_SET_RETXML（0x00000002）は廃止されデフォルト設定が XML を返すようになった。LCSI_SET_RETXML は指定しても無視される。 ➢ LCSI_CAPI_NOCHECK（0x00000800）が追加され指定時には constructor では証明書チェックは一切行わない。その代わり cert にて LCSC_CAPI_ICCARD（0x00000100）を指定して IC カード（マイナンバーカードを含む標準 IC カード）のチェックを行う。これによりカード未挿入時の再チェックと初期化が可能となる。 ➢ V2.1.R2 において Windows 証明書ストアの CNG 形式の秘密鍵(署名鍵)にも対応した。
V2.2 補足	➢ LCSI_SET_WININET（0x00008000）が追加され指定時には HTTP/HTTPS 通信に WinInet を利用する。未指定時には WinHTTP を利用。プロキシ設定については「1.8. HTTP 通信のプロキシ設定」を参照。

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．３．２．connect：署名コマンド実行と接続

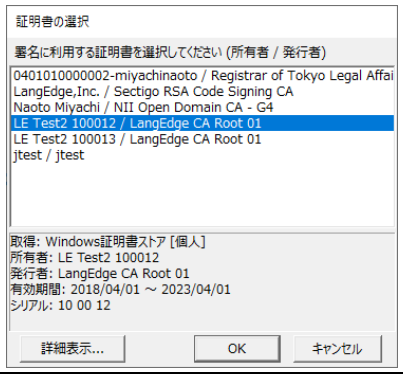
機能		connect：署名コマンド実行と接続：署名コマンドを実行して WebSocket 通信で接続		
JSAPI 仕様		(int) await connect(port, waitSec, cmdSec);		
引数	1	数値	port	署名コマンドとの WebSocket 通信に使うポート番号 ※ 0 以下ならデフォルト設定 50000 番となる
	2	数値	waitSec	署名コマンドとの JavaScript 側の接続最大待ち時間の指定 ※ 0 以下ならデフォルト設定 20 秒となる
	3	数値	cmdSec	署名コマンドとの署名コマンド側の接続最大待ち時間の指定 ※ 0 以下ならデフォルト設定 5 秒となる
戻り値		数値	ゼロ値： エラーは無かった / マイナス値： エラー値が返る。 CMD_NOT_REGIST (-1)： 署名コマンドが未登録の可能性あり(未インストール). CMD_CONNECT_ERR (-3)： 署名コマンドへの通信接続に失敗した	
備考		動的・プライベートに利用可能なポート番号は 49152～65535 となっている。		
備考		署名コマンドは実行されてから WebSocket 接続まで cmdSec 秒待ち、接続されなかった場合には未インストールとする。接続待ち中にはウィンドウが表示される。		
備考		CMD_CONNECT_ERR になるケースは「実行確認ダイアログをキャンセル」「実行確認ダイアログを waitSec 以上経過して開くを実行」「常に許可済み後にアンインストール」の 3 パターンがあります。詳しくは「3.2.3. 署名コマンド実行の判定」を参照。		

３．３．３．check：署名コマンドインストールチェック

機能		check：署名コマンドインストールチェック：署名コマンドを実行して確認する ※ connect を使うかわりに利用する。
JSAPI 仕様		(int) await check ();
引数		無し
戻り値	数値	ゼロ値： エラーは無かった / マイナス値： エラー値が返る。 CMD_NOT_REGIST (-1)： 署名コマンドが未登録の可能性あり(未インストール). EXEC_CANCEL (-2)： ブラウザの実行確認をキャンセルした
備考		署名コマンドは実行されるがインストール済みかつ常に許可済みの場合にはすぐに終了する。terminate の呼び出しは不要。
備考		JavaScript からレジストリのチェックが出来ない為に、現在は署名コマンドを実行しないとインストール済みチェックが行えない。
備考		Edge/Chrome の仕様変更により確実なインストール済みチェックは出来ない。

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．３．４．cert：証明書取得

<table>
<tr><td>機能</td><td colspan="3">cert：証明書取得：署名に使う証明書を選択する</td></tr>
<tr><td>JSAPI 仕様</td><td colspan="3">(string) await cert (cflag, copt);</td></tr>
<tr><td rowspan="2">引数</td><td>1</td><td>数値</td><td>cflag</td><td>証明書フラグ：共通部
0x00000000 LCSC_NO_FLAG オプション無し
0x00000001 LCSC_CONFIRM 証明書１つの場合も確認</td></tr>
<tr><td>2</td><td>文字列</td><td>copt</td><td>オプション：省略する場合は NULL 指定
利用する証明書を制限する（未サポート）</td></tr>
<tr><td>戻り値</td><td colspan="2">文字列</td><td colspan="2">正常時：
選択した証明書の X.509/DER 形式バイナリの Base64 文字列が返る
※ LCSC_RETXML 指定時は独自 XML 形式で返る（4.1.1.参照）
エラー時：
NULL が返る
別途 errorGet() の利用も可能
※ エラー時には terminate() が呼ばれ署名コマンドは終了する。</td></tr>
<tr><td>PKCS#11
補足</td><td colspan="4">証明書フラグ：
0x00000010 LCSC_P11_ALLCERT CA 証明書も含め全ての証明書を表示
0x00000020 LCSC_P11_USEPIN ログイン（PIN 必要）をして利用（JPKI 等）
オプション（LCSC_P11_USEPIN 指定時）：
opt に PIN 文字列の指定が可能（NULL 指定で独自ダイアログ）</td></tr>
<tr><td>CAPI 補足</td><td colspan="4">証明書フラグ：
0x00000100 LCSC_CAPI_ICCARD 標準 IC カード(JPKI/特定 CA 等)のチェックを行う
説明：
通常は初期化（contructor）時に証明書や IC カードのチェックを行うが、初期化時のフラグに LCSI_CAPI_NOCHECK（0x00000800）を指定することで、cert の証明書フラグ LCSC_CAPI_ICCARD（0x00000100）を指定することで IC カードのチェックを行える。LCSC_CAPI_ICCARD したのに LCS_NO_CERT_ERR（-11）となる場合には IC カードが挿入されていない。
証明書選択画面：
 

※ V2.10.R2 より詳細表示に拡張キー用途の情報があれば表示するようになった。</td></tr>
</table>

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．３．５．info：証明書情報取得

機能		info：証明書情報取得：証明書を解析して情報を取得する		
JSAPI 仕様		(string) await info (cert, infoflag);		
引数	1	文字列	cert	解析する証明書の X.509/DER 形式バイナリの Base64 文字列 ※ cert 戻り値の独自 XML 形式の指定も可能。つまり cert の戻り値をそのまま指定可能。
	2	数値	infoflag	情報フラグ： 0x00000000 LCSF_NO_FLAG オプション無し（JSON 返し） 0x00000001 LCSF_RETXML XML 形式で情報を返す
戻り値		文字列	正常時： 選択した証明書の情報が JSON 文字列で返る ※ LCSF_RETXML 指定時は独自 XML 形式で返る エラー時： NULL が返る 別途 errorGet() の利用も可能 ※ エラー時には terminate() が呼ばれ署名コマンドは終了する。	
補足		戻り値で返される情報の形式は「付録２．証明書情報一覧」を参照		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．３．６．sign：署名値計算

機能		sign：署名値計算：署名を実行して署名値を返す（直前の選択証明書を利用）		
JSAPI 仕様		(string) await sign (sflag, sopt, signAlg, data, passwd);		
引数	**1**	数値	sflag	署名フラグ：共通部 0x00000000　LCSS_DEFAULT　　data にハッシュ値を指定 0x00000001　LCSS_TDIRECT　　data に対象自体を指定
	2	文字列	sopt	オプション：省略する場合は NULL 指定 利用する証明書を指定する（未サポート）
	3	数値	signAlg	署名アルゴリズム： SHA256=0 / SHA384=1 / SHA512=2 / SHA1=10
	4	文字列	data	署名対象データ：Base64 文字列でバイナリ指定 LCSS_DEFAULT 時 = 署名対象のハッシュ値を指定 LCSS_TDIRECT 時 = 署名対象自体を指定（ハッシュ計算含） ※ 設定に関わらず 1 文字目が '<' であれば独自のハッシュ値 XML 形式（LCS_HASH：4.1.3 を参照）と判断する。
	5	文字列	passwd	PKCS#11： 利用パスワード指定（NULL の場合は通常エラー） ※ LCSS_P11_PINDLG 指定時は NULL 指定で独自ダイアログ CAPI： 未使用（IC カード利用時はドライバ側でダイアログ表示）
戻り値		文字列	正常時： 選択した署名値の Base64 文字列でバイナリが返る ※ LCSI_RETXML 指定時は独自 XML 形式で返る（4.1.3.参照） エラー時： NULL が返る 別途 errorGet()/errorApi() の利用も可能 ※ エラー時には terminate() が呼ばれ署名コマンドは終了する。	
PKCS#11 補足		署名フラグ： 0x00000010　LCSS_P11_NODLG　　passwd が NULL の場合にエラーにする		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．３．７．xadd：署名 XAdES-BES への署名対象追加

機能		xadd：署名 XAdES-BES への署名対象追加：Detached の追加 （複数回呼び出すことで複数の署名対象追加が可能）		
JSAPI 仕様		(int) await xadd (type, id, fname, data);		
引数	**1**	数値	type	XAdES 対象追加種別： 0x00000000 LCSA_DETACHED Detached 形式で対象を追加 0x00000001 LCSA_ENVELOPING Enveloping 形式で対象を追加 Enveloping の追加は現在未対応
	2	文字列	id	ID 文字列を指定、省略不可、例："DET01"
	3	文字列	fname	ファイル名を指定 Enveloping 時には data を指定すれば省略可能 Detached 指定時には省略不可、例："sample.docx" ※ フルパスを指定した場合にはファイル名と拡張子部のみが利用され、フルパスはデータの読み込みのみに利用される。
	4	文字列	data	署名対象バイナリを Base64 文字列化したデータ指定 ※ null 指定にて省略可能（null 時にはフルパス指定が必要）
戻り値		数値	正常時： LCS_OK が返る エラー時： LCS_OK 以外が返る（マイナス値）	
補足		※ 現在 Enveloping（内包）形式の署名対象追加は未サポート		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．３．８．xades：署名 XAdES-BES 生成

機能		xades：署名 XAdES-BES 生成：XAdES-BES 署名を実行して XAdES の XML を返す （直前の選択証明書を利用） ※ 本機能ではフルセットの XAdES では無く簡易な XAdES-BES のみ対応		
JSAPI 仕様		(string) await xades (xflag, signAlg, xml, id, xpath, passwd);		
引数	**1**	数値	xflag	XAdES フラグ：共通部 0x00000000 LCSX_DEFAULT 共通：指定無し
	2	数値	signAlg	署名アルゴリズム： SHA256=0 / SHA384=1 / SHA512=2 / SHA1=10
	3	文字列	xml	署名対象 XML： NULL の場合は Detached/Enveloping の指定が必要
	4	文字列	id	署名対象 id の指定：xml 中の署名対象となる id を指定 NULL の場合は Enveloped 形式となる
	5	文字列	xpath	署名対象の xpath の指定：オプション NULL にて省略可能
	6	文字列	passwd	PKCS#11： 利用パスワード指定（NULL の場合は通常エラー） ※ LCSX_P11_NODLG 未指定時は NULL 指定で独自ダイアログ CAPI： 未使用（IC カード利用時はドライバ側でダイアログ表示）
戻り値		文字列	正常時： 生成した XAdES-BES の XML が返される ※ 初期化に LCSI_SET_RETHEX 指定時は HEX 文字列で返される ※ 初期化に LCSI_SET_RETB64 指定時は Base64 文字列で返される エラー時： NULL が返る 別途 errorGet()/errorApi() の利用も可能 ※ エラー時には terminate() が呼ばれ署名コマンドは終了する。	
PKCS#11 補足		署名フラグ： 0x00000010 LCSX_P11_NODLG passwd が NULL の場合にエラーにする		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．３．９．terminate：署名コマンド終了

機能	terminate：署名コマンドの終了：実行して接続していた署名コマンドを終了する	
JSAPI 仕様	(void) await terminate ();	
引数	無し	
戻り値	無し	

３．３．１０．errorGet：エラー値取得

機能	errorGet：エラー値取得：直前に生じたエラー値を返す	
JSAPI 仕様	(int) errorGet ();	
引数	無し	
戻り値	数値	正常時： LCS_OK が返る エラー時： LCS_OK 以外が返る（マイナス値）
備考	本 API は JavaScript の API である為に、署名コマンドとの通信は行われない。	

３．３．１１．status：IC カード状態取得（PKCS#11 のみ）

機能	status：IC カード状態取得：IC カードの状態を返す（PKCS#11 のみ）	
JSAPI 仕様	(int) await status ();	
引数	無し	
戻り値	数値	正常時： IC カードがセット済みなら 1 が未セットなら 0 が返る エラー時： マイナス値が返る ※ エラー時には terminate() が呼ばれ署名コマンドは終了する。

２．３．１２．version：バージョン情報取得

機能		version：バージョン情報取得：クライアント署名のバージョン情報を返す	
ActiveX		(string) await version ();	
引数	1	無し	
戻り値		文字列	バージョン情報文字列が返る 例）"v1.00RC1"

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．３．１３．http：HTTP/HTTPS 通信

機能		http：HTTP/HTTPS 通信：POST メソッドにより HTTP/HTTPS 通信を行う		
JSAPI 仕様		(string) await http (url, send, head, hflag);		
引数	**1**	文字列	url	接続 URL を指定
	2	文字列	send	POST 方式で送信する BODY を指定
	3	文字列	head	Basic 認証等の HTTP ヘッダがあれば指定（null なら指定無し）
	4	数値	hflag	HTTP 通信フラグ ※ 現在未対応の為に 0 を指定
戻り値		文字列	非 null 値：サーバからの応答文字列が返る null 値：エラー（errorGet でエラー値取得可能） ※ エラー時には terminate() が呼ばれ署名コマンドは終了する。	

３．３．１４．pxml：独自 XML 応答の解析

機能		pxml：独自 XML 応答の解析：サーバから取得した独自 XML 応答（4.1 参照）を解析		
JSAPI 仕様		(string) await pxml (xml);		
引数	**1**	文字列	xml	独自 XML を指定
戻り値		文字列	非 null 値：サーバからの応答文字列が返る 　ハッシュ値 XML：ハッシュ値の HEX 文字列 　結果 XML：リダイレクト先（未指定の場合は null 値が返る） null 値：サーバからの応答文字列が無かった ※ エラー時には terminate() が呼ばれ署名コマンドは終了する。	
備考		解析エラー時には errorGet でエラー値が取得可能		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．３．１５．exec：一括実行

機能		exec：一括実行：独自 HTTP 通信/XML 形式にて署名処理を一括実行する		
JSAPI 仕様		(string) await exec (cflag, copt, curl, sflag, sopt, signAlg, surl, head, hflag);		
引数	**1**	数値	cflag	証明書フラグ：共通部 0x00000000　LCSC_NO_FLAG　　オプション無し 0x00000001　LCSC_CONFIRM　　証明書１つの場合も確認
	2	文字列	copt	オプション：省略する場合は NULL 指定 利用する証明書を制限する（未サポート）
	3	文字列	curl	証明書送信 URL の指定（必須：null 不可）
	4	数値	sflag	署名フラグ：共通部 0x00000000　LCSS_DEFAULT　　data にハッシュ値を指定 0x00000001　LCSS_TDIRECT　　data に対象自体を指定
	5	文字列	sopt	PKCS#11： 利用パスワード指定（NULL の場合は通常エラー） ※ LCSS_P11_PINDLG 指定時は NULL 指定で独自ダイアログ CAPI： 未使用（IC カード利用時はドライバ側でダイアログ表示）
	6	数値	signAlg	署名アルゴリズム： SHA256=0 / SHA384=1 / SHA512=2 / SHA1=10
	7	文字列	surl	署名値送信 URL の指定（必須：null 不可）
	8	文字列	head	Basic 認証等の HTTP ヘッダがあれば指定(null なら指定無し)
	9	数値	hflag	HTTP 通信フラグ ※ 現在未対応の為に 0 を指定
戻り値		文字列	正常時： リダイレクト先がサーバから指定された場合はリダイレクト先が返る エラー時： NULL が返る 別途 errorGet()/errorApi() の利用も可能 ※ エラー時には terminate() が呼ばれ署名コマンドは終了する。	

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．３．１６．LeCSignCmdWS クラス エラー値

署名コマンドクラスのエラー値は JavaScript クラス LeCSignCmdWS 内のエラー値であり、マイナス一桁の値となる。正常終了は 0 が返り、証明書選択ダイアログのキャンセルは 1 が返る。署名 LeCSignCmdWS.js に設定されている。

```
// エラー値設定
var LSCERR = {
        NO_ERR          :  0,   // 正常終了.
        CERT_DLG_CANCEL :  1,   // 証明書選択ダイアログがキャンセルされた.
        CMD_NOT_REGIST  : -1,   // 署名コマンドが未インストール(実行不可).
        EXEC_CANCEL     : -2,   // ブラウザにより実行がキャンセルされた
        CMD_CONNECT_ERR : -3,   // 署名コマンドの接続時エラーがあった
        INVALID_ARG     : -4,   // 関数の引数異常(引数のチェックが必要).
        NO_SETUP        : -5,   // 実行に必要な項目に未設定がある(必須項目設定不足).
        WS_ERROR        : -6,   // WebSocket エラー.
        INVALID_RESP    : -7,   // 応答の仕様異常.
        UNKNOWN_ERR     : -9    // 不明なエラー(ダミー:生じない)
};
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．４．署名コマンド利用実装例

CAPI と PKCS#11 のどちらでも HTML 部の実装は同じとなる。

HTML 部の例：

```
<!-- HTML BODY -->
<body>
<h2>署名コマンド試験：</h2>
<p>
<!-- ボタンクリックで署名コマンドを実行(セッション ID を指定) -->
<input type="button" id="start" value="署名実行" onclick="execSign('TEST01')">
</p>
<div id="message">no message.</div>
</body>
```

３．４．１．CAPI（CryptoAPI）利用

実装サンプル：bin_cmd/LeCSignCmdDemo.htm

実装サンプル：bin_cmd/LeCSignCmdWS.htm

JavaScript 署名実行の例：

```
<!-- LE:署名コマンド JS (※先に指定が必要) -->
<script src="LeCSignCmdWS.js"></script>

<!-- 署名個別実装 JS -->
<script type="text/javascript">

  // 署名処理実行(参考用の実装例).
  async function execSign(sid){

    // 署名コマンド生成/初期化(必須).
    var initflag = 0x00000000;    // 初期化フラグ(デフォルト CAPI)
//  initflag    |= 0x00000001;    // LCSI_USE_P11 PKCS#11 利用(initopt1 必須)
    initflag    |= 0x00000002;    // LCSI_RET_XML cert/sign の結果を独自 XML 形式で返す
    initflag    |= 0x00000100;    // LCSI_CAPI_ICCARD  標準 IC カード利用
    var initopt1 = null;          // 初期化オプション 1(PKCS#11 の DLL パス等)
    var initopt2 = null;          // 初期化オプション 2(ログファイルパス等)
    var sessionid = sid;          // 連携セッション ID
    var cscmd = new LeCSignCmdWS(initflag, initopt1, initopt2, sessionid);

    // 署名コマンドの起動と接続
    var port = -1;       // 接続ポート番号指定(-1 ならデフォルト設定 50000 を利用)
    var waitSec = -1;    // JS 側接続待ち時間指定(-1 ならデフォルト設定 20 秒 を利用)
    var cmdSec = -1;     // CMD 側接続待ち時間指定(-1 ならデフォルト設定 5 秒 を利用)    rc =
await cscmd.connect(port, waitSec, cmdSec);
    if(rc != 0) {
      mesgElement.innerHTML = "Error: SignCmd connect.";
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
      if(rc == LSCERR.CMD_NOT_REGIST) {
        // 署名コマンドが未登録(未インストール).
      } else if(rc == LSCERR.CMD_CONNECT_ERR) {
        // 署名コマンド接続がエラーだった.
      } else {
        // その他のエラー
      }
      delete cscmd;
      return;
    }

    // 証明書設定
    var certflag   = 0x00000000;  // 証明書選択フラグ(デフォルト=0x00000000)
    certflag      |= 0x00000001;  // LCSC_CONFIRM/証明書1つの場合も確認
    var certopt    = null;        // 証明書選択オプション(null 指定)
    // 署名値設定
    var signflag   = 0x00000000;  // signtarget にハッシュ値(HEX)か独自 XML を指定
    var signalg    = 0;           // 署名アルゴリズム(SHA256=0/SHA384=1/SHA512=2/SHA1=10)
    var signopt    = null;        // 署名時オプション指定(PKCS#11 利用時に PIN 指定)
    // HTTP 通信設定
    certurl = hostname + "https://dev.langedge.jp/tom/lcsign/pcert";  // 証明書送信 URL
    signurl = hostname + "https://dev.langedge.jp/tom/lcsign/psign";  // 署名値送信 URL
    var head       = null;        // オプションヘッダ指定
    var hflag      = 0x00000001;  // HTTP 通信フラグ(フルパス指定=0x00000001)

    // 一括実行コマンド実行
    var redirect = await cscmd.exec(certflag, certopt, certurl,
                                    signflag, signopt, signalg, signurl, head, hflag);
    rc = cscmd.errorGet();
    if(rc != 0 || redirect == null) {
        return;
    }

    // 署名コマンド正常終了(解放)
    await cscmd.terminate();
    delete cscmd;

    // 指定された場合はリダイレクトを別ウィンドウで開く
    if(redirect) {
      alert(redirect);
      window.open(redirect);
    }

  }

</script>
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．４．２．PKCS#11 利用

実装サンプル：bin_cmd/LeCSignCmdWS.htm

JavaScript 署名実行の例：

```
<!-- LE:署名コマンド JS (※先に指定が必要) -->
<script src="LeCSignCmdWS.js"></script>

<!-- 署名個別実装 JS -->
<script type="text/javascript">

  // 署名処理実行(参考用の実装例).
  async function execSign(sid){

    // 署名コマンド生成/初期化(必須).
    var initflag = 0x00000000;    // 初期化フラグ(デフォルト CAPI)
    initflag    |= 0x00000001;    // LCSI_USE_P11 PKCS#11 利用(initopt1 必須)
    initflag    |= 0x00000002;    // LCSI_RET_XML cert/sign の結果を独自 XML 形式で返す
    var initopt1 = "pkcs11.dll";  // 初期化オプション 1(PKCS#11 の DLL パス等)
    var initopt2 = null;          // 初期化オプション 2(ログファイルパス等)
    var sessionid = sid;          // 連携セッション ID
    var cscmd = new LeCSignCmdWS(initflag, initopt1, initopt2, sessionid);

    // ※ 以下略 (CAPI と同じ)

  }

</script>
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．４．３．JavaScript の async/await 補足

署名コマンドは WebSocket を利用した非同期（async）の動作をする。このままでは API が全て非同期実行されてしまい、API を順番に呼ぶことが困難となる。この為に署名コマンドの API では、Edge/Chrome の JavaScript において非同期の async function を同期的に実行する await 呼び出しを推奨している。しかしながら await を利用するファンクション自体が async ファンクションである必要がある制約があり、呼び出し元を全て非同期の async function にする必要が出てしまう。

これを避ける為には async function の戻りである Promise を使って .then() で終了を待ち受けることで、元ファンクションを非 async function にする方法がある。これにより async function 階層の連鎖を止めることができる。ただし await と異なり基本的には非同期の処理となり、結果を .then() により待つ点が異なるので、最上位では非同期処理が必要となる。

async/await と Promise による同期（sync）ファンクションの例：

```
<script type="text/javascript">

  // 同期(sync)ファンクションによる async 同期的実行の実装例
  function main(){
    console.log("main 開始");
    // 同期処理開始
    afunc1().then(async function() {
       // afunc1 終了(全処理終了)
       console.log("main 全処理終了");
    });
    // 処理終了を待たずに実行される(非同期)
    console.log("main 終了");
  }

  // 非同期(async)ファンクション 1 による async 同期的実行の実装例
  async function afunc1(){
    console.log("afunc1 開始");
    // 同期処理開始
    await afunc2();                                  // afunc2 を同期実行
    // afunc2 処理終了を待ってから次へ(同期)
    console.log("afunc1 終了");
  }

  // 非同期(async)ファンクション 2(主処理の実装例)
  async function afunc2() {
    console.log("afunc2 開始");
    await 0;                                         // 非同期にする為に await を実行
    console.log("afunc2 終了");
  }

</script>
<body><input type="button" id="start" value="試験実行" onclick="main()"></body>
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

またIE11のActiveXの実行と、Edge/Chromeの署名コマンドの実行を行う場合に、IE11ではasyncファンクションがエラーとなる。これを避けるには PHP 等によりソースの場合分けを行う方法がある。以下に IE11 と Edge/Chrom のソース分けの例を示す。なお PHP が使える環境が前提となる。

IE11 と Edge/Chrome のソース分けの例：

```
<?php if ( isIe11() ): ?>
  <script> // IE11 実行
    function SignMethod() { alert("IE11 exec."); };
  </script>
<?php else: ?>
  <script> // IE11 以外での実行
    async function SignMethod() { alert("async exec."); };
  </script>
<?php endif; ?>
```

※ isIe11 は JavaScript による IE11 判定関数。

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．４．４．XAdES と証明書情報の利用（V2.1 新機能）

実装サンプル：bin_cmd/LeCSignCmdXades.htm

JavaScript 署名実行の例：

```
<!-- LE:署名コマンド JS (※先に指定が必要) -->
<script src="LeCSignCmdWS.js"></script>

<!-- 署名個別実装 JS -->
<script type="text/javascript">

  // ============================================================
  // 署名処理実行(参考用の実装例).
  async function execSign(){

    // ------------------------------------------------------------
    // オプション１：マイナンバーカード指定チェック
    var jpkitest  = false;
    const jpki = document.getElementById("jpki");
    if(jpki.checked) {
      console.log("試験：マイナンバーカード指定");
      jpkitest = true;
    }

    // ------------------------------------------------------------
    // 処理開始.
    var rc = 0;
    console.log('署名処理開始:');
    // 出力要素取得
    var CTYPE = document.getElementById("CTYPE");
    var CTIME = document.getElementById("CTIME");
    var COPT1 = document.getElementById("COPT1");
    var COPT2 = document.getElementById("COPT2");
    var COPT3 = document.getElementById("COPT3");
    var COPT4 = document.getElementById("COPT4");
    CTYPE.innerHTML = "";
    CTIME.innerHTML = "";
    COPT1.innerHTML = "";
    COPT2.innerHTML = "";
    COPT3.innerHTML = "";
    COPT4.innerHTML = "";

    // ------------------------------------------------------------
    // ブラウザ確認(現在 Chrome と新 Edge のみ利用可能).
    var agent = window.navigator.userAgent.toLowerCase();
    var target = (agent.indexOf('edg') !== -1) || (agent.indexOf('chrome') !== -1);
    if(target != true) {
      console.log(agent);
      console.log('未サポートのブラウザです。');
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
      alert('未サポートのブラウザです。¥nEdge または Chrome を使ってください。');
      return;
    }
    console.log('ブラウザ確認 OK');

    // ---------------------------------------------------------
    // 署名コマンド生成/初期化(必須).
    var initflag = LCSI_FLAG.LCSI_NO_FLAG;        // 初期化フラグ(デフォルト CAPI)
    if(jpkitest == true) {
      // マイナンバーカード
      initflag  |= LCSI_FLAG.LCSI_CAPI_NOCHECK;  // CAPI:証明書チェック無し(LCSC_CAPI_JPKI
利用の為)
    } else {
      // 証明書選択
      initflag  |= LCSI_FLAG.LCSI_CAPI_ICCARD;    // CAPI:標準 IC カード利用
    }
//  initflag    |= LCSI_FLAG.LCSI_SET_DEBUG;      // デバッグモード指定
    var initopt1 = null;                // 初期化オプション 1(PKCS#11 の DLL パス等)
    var initopt2 = null;                // 初期化オプション 2(ログファイルパス等)
    var sessionid = "DUMMY";            // 連携セッション ID(本試験では使わないのでダミー)
    var cscmd = new LeCSignCmdWS(initflag, initopt1, initopt2, sessionid);
    if(cscmd == null) {
      alert('エラー:署名コマンドクラスが初期化できませんでした。');
      return;
    }

    // ---------------------------------------------------------
    // 署名コマンドの実行と接続
    var port = -1;       // 接続ポート番号指定(-1 ならデフォルト設定 50000 を利用)
    var waitSec = -1;    // JS 側接続待ち時間指定(-1 ならデフォルト設定 20 秒 を利用)
    var cmdSec = -1;     // CMD 側接続待ち時間指定(-1 ならデフォルト設定 5 秒 を利用)
    rc = await cscmd.connect(port, waitSec, cmdSec);
    if(rc != LSCERR.NO_ERR) {
      // 署名コマンドの実行か接続に失敗した
      if(rc == LSCERR.CMD_NOT_REGIST) {
        // 署名コマンドが未登録(未インストール).
        var chk = confirm('エラー(' + rc + '):署名コマンドが未インストールの可能性があり
ます。¥n インストーラをダウンロードしますか?');
        if(chk == true) {
                window.location.href = "./LeCSignCmdSetup.zip";
                alert('インストーラをダウンロードしています。¥n ダウンロード後にファイルを
開いてインストールしてください。');
        }
      } else if(rc == LSCERR.CMD_CONNECT_ERR) {
        // 署名コマンドがエラーだった.
        alert('エラー(' + rc + '):署名コマンドの接続エラーがありました。');
      } else {
        // その他のエラー
        alert('エラー(' + rc + '):署名コマンドの接続異常です。');
      }
      delete cscmd;
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
      return;
    }

    // ======================================================
    // 署名コマンド接続完了(API 実行)

    // ------------------------------------------------------
    // 証明書設定
    var certdata  = null;                    // 証明書データ
    var certopt   = null;                    // 証明書選択オプション(省略時 null 指定)
    var certflag  = LCSC_FLAG.LCSC_NO_FLAG;  // 証明書選択フラグ(デフォルト=0x00000000)
//  certflag     |= LCSC_FLAG.LCSC_CONFIRM;  // デバッグ：証明書 1 つの場合も確認
    if(jpkitest == true) {

      // マイナンバーカード（標準 IC カードにマイナンバーカードも含まれる）
      certflag    |= LCSC_FLAG.LCSC_CAPI_ICCARD;  // CAPI:標準 IC カードチェック
      alert('IC カードをセットしてください。¥n ご注意：PIN 入力画面が裏に表示される場合があ
ります。');
      while(true) {
        // 証明書取得コマンド(cert)実行：IC カードのチェックと取得
        certdata = await cscmd.cert(certflag, certopt);
        rc = cscmd.errorGet();
        if(rc == LSCERR.NO_ERR && certdata != null)
          break;                             // 取得できたのでループを終了
        // 取得できなかった
        if(rc == -11) {
          // IC カードが見つからない（繰り返し）
          var res = confirm('IC カードをセットしてください。');
          if(res == false) {
            // キャンセル
            alert('IC カードチェックがキャンセルされました。');
            await cscmd.terminate();     // ※エラーではないので terminate()が必須
            delete cscmd;
            return;
          }
        } else {
          // その他エラー
          alert('エラー(' + rc + ')：証明書取得エラーです。');
          delete cscmd;
          return;
        }
      }

    } else {

      // 証明書取得コマンド(cert)実行：通常の証明書選択
      certdata = await cscmd.cert(certflag, certopt);
      rc = cscmd.errorGet();
      if(rc != LSCERR.NO_ERR) {
        // エラー
        if(rc == LSCERR.CERT_DLG_CANCEL) {
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
          alert('証明書選択ダイアログがキャンセルされました。');
        } else {
          // キャンセル
          if(certdata == null)
            alert('エラー(' + rc + '):証明書取得エラーです。');
          else
            alert('エラー(' + rc + '):' + certdata);
        }
        delete cscmd;
        return;
      }

    }
    console.log("証明書取得 OK");

    // ---------------------------------------------------------
    // 証明書情報取得設定
    var infoflag   = LCSF_FLAG.LCSF_NO_FLAG;  // 結果を JSON 形式で返す(デフォルト)
//  infoflag      |= LCSF_FLAG.LCSF_RETXML;   // 結果を XML 形式で返す
    // 証明書情報取得(info)実行
    var certinfo = await cscmd.info(certdata, infoflag);
    rc = cscmd.errorGet();
    if(rc < LSCERR.NO_ERR) {
      if(certinfo == null)
        alert('エラー(' + rc + '):証明書情報取得エラーです。');
      else
        alert('エラー(' + rc + '):' + certinfo);
      delete cscmd;
      return;
    }
    console.log("証明書情報取得 OK");

    // ---------------------------------------------------------
    // 結果の解析
    var certobj = JSON.parse(certinfo);
    // 期間
    var ctime = "期間:";
    if(certobj.from != null)
      ctime += certobj.from;
    ctime += "～";
    if(certobj.to != null)
      ctime += certobj.to;
    // 種別
    if(certobj.jpki != null) {
      CTYPE.innerHTML = "種別:マイナンバーカード(JPKI)";
      CTIME.innerHTML = ctime;
      COPT1.innerHTML = "氏名:" + certobj.jpki.name;
      COPT2.innerHTML = "性別:" + certobj.jpki.gender;
      COPT3.innerHTML = "生年月日:" + certobj.jpki.birth;
      COPT4.innerHTML = "住所:" + certobj.jpki.address;
    } else if(certobj.regist != null) {
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
      CTYPE.innerHTML = "種別：商業登記証明書";
      CTIME.innerHTML = ctime;
      COPT1.innerHTML = "法人名：" + certobj.regist.corp;
      COPT2.innerHTML = "代表者：" + certobj.regist.name;
      COPT3.innerHTML = "肩書：" + certobj.regist.title;
      COPT4.innerHTML = "住所：" + certobj.regist.address;
    } else {
      CTYPE.innerHTML = "種別：一般証明書";
      CTIME.innerHTML = ctime;
      var info = "発行：";
      if(certobj.issuer.cn != null)
        info += certobj.issuer.cn;
      else if(certobj.issuer.ou != null)
        info += certobj.issuer.ou;
      else if(certobj.issuer.o != null)
        info += certobj.issuer.o;
      else
        info += "CN/OU/O 無し";
      COPT1.innerHTML = info;
      COPT2.innerHTML = "";
      COPT3.innerHTML = "";
      COPT4.innerHTML = "";
    }
    console.log("証明書情報セット OK");

    // ------------------------------------------------------
    // 署名設定
    var orgxml  = "<LeTest><Contents Id=¥"TEST01¥">";
        orgxml     += "<Data>Sign Data</Data>";
        orgxml     += "<日本語>試験です</日本語>";
        orgxml     += "</Contents></LeTest>";   // 署名対象 XML
    var xflag        = 0x00000000; // XAdES フラグを指定(デフォルト)
    var signAlg      = 0;          // 署名アルゴリズム(SHA256=0/SHA384=1/SHA512=2/SHA1=10)
    var xml          = orgxml;     // 署名対象 XML
    var id           = "TEST01";   // 署名対象を ID 指定(Enveloped の場合は null 指定)
    var xpath        = null;       // 署名対象の XPATH 指定(不要時には null 指定)
    var passwd       = null;       // 署名時パスワード指定(PKCS#11 利用時に PIN 指定、省略時
null 指定)
    // XAdES-BES 生成コマンド(xades)実行
    var xadesXml = await cscmd.xades(xflag, signAlg, xml, id, xpath, passwd);
    rc = cscmd.errorGet();
    if(rc != LSCERR.NO_ERR) {
      if(xadesXml == null)
        alert('エラー(' + rc + ')：XAdES-BES 生成エラーです。');
      else
        alert('エラー(' + rc + ')：' + xadesXml);
      delete cscmd;
      return;
    }
    console.log("XAdES 生成 OK");
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
    // ------------------------------------------------------------
    // 結果のアップロード(/signcmd/up は送信した内容をそのまま返す)
    var upurl = "https://dev.langedge.jp/tom/signcmd/up";      // 連携ホスト.
    var head  = null;          // オプションヘッダ指定
    var hflag = 0x00000000;  // HTTP 通信フラグ
    var respXml = await cscmd.http(upurl, xadesXml, head, hflag);
    rc = cscmd.errorGet();
    if(rc != LSCERR.NO_ERR) {
      if(respXml == null)
        alert('エラー(' + rc + '):結果の HTTP 通信エラーです。');
      else
        alert('エラー(' + rc + '):' + respXml);
      delete cscmd;
      return;
    }
    console.log("XAdES アップロード OK");

    // ------------------------------------------------------------
    // 結果の保存 (表示)
    saveXml("xades.xml", respXml);

    // ============================================================
    // 署名コマンド正常終了(解放)
    await cscmd.terminate();
    delete cscmd;

    // ------------------------------------------------------------
    // 処理正常終了
    console.log('署名処理終了: OK');

  }

  // ------------------------------------------------------------
  // 試験用:ファイル表示
  function saveXml(filename, text) {
        let blob = new Blob([text],{type:"application/xml"});
        let link = document.createElement('a');
        link.href = URL.createObjectURL(blob);
        /* ファイル保存(コメントを外す)
        link.download = filename;
        link.click();
        console.log("saveFile: " + filename);
        /* */
        window.open(link.href);          // 別ウィンドウで表示
    // 後始末
        URL.revokeObjectURL(link.href);
  }

</script>
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．５．商業登記リモート署名

商業登記リモート署名を利用するには、商業登記ポータルサイトにて配布されている「商業登記リモート署名ドライバソフト」が必要となる。詳しくは商業登記ポータルを参照。

商業登記リモート署名では署名時のエラーとして、GetLastError の値を返す。このために V2.2.R2 ではログファイルのエラー出力として GetLastError の値を出力するようになった。以下例。

```
2026-06-29T18:24:53 sign / error(-52) GetLastError = 0xE0020005
2026-06-29T18:24:53 exec.sign / error(-52) CAPI の署名実行エラー(-52)
```

商業登記リモート署名ドライバソフトの署名時（sign 呼び出し）にエラーがあった場合に、GetLastError にて得られるエラーコードは以下の通りとなる。

値	意味
0xE0010001	セッション情報[アクセストークン、リフレッシュトークン]の取得に失敗した。
0xE0010002	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。
0xE0020001	G ビズ ID の認証エラーが発生した。
0xE0020002	認可コードの生成に失敗した。
0xE0020003	認可の要求に失敗した[鍵パスワードエラー等]。
0xE0020004	認可の要求に対するチェックが失敗した。
0xE0020005	鍵パスワード入力画面で「キャンセル」された。
0xE0020006	認可コード表示画面で「キャンセル」された。
0xE0020007	G ビズ ID アプリで認可が「拒否」された。
0xE0030001	G ビズ ID との通信に失敗した。
0xE0030002	リモート署名システムとの通信に失敗した。
0xE0040001	SAD の延長に失敗した。
0xE0050001	署名値の取得に失敗した。
0xE0050002	取得した署名値のリスト返却に失敗した。
0xE0060001	G ビズ ID ログイン中に操作タイムアウト[10 分]が発生した。

商業登記リモート署名の利用時以外にも IC カードや USB トークンの利用時に GetLastError の値がログ出力される可能性がある。その場合の値の意味は Windows 標準のエラーとなる。詳細は以下の URL 等で確認が可能となっている。

https://learn.microsoft.com/ja-jp/windows/win32/com/com-error-codes-4

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

４．サーバ側実装（LE:XAdES:Lib/LE:PAdES:LIb）

４．１．独自通信 XML プロトコル（LE:XAdES:Lib/LE:PAdES:Lib 共通）

ActiveX プラグインも署名コマンドも同じ独自通信 XML プロトコルを利用する。セッション ID 等の初期化時の引数やパラメーターはサーバ側が生成する HTML/JavaScript へ埋め込まれる形で提供されるが、その後最低 2 回の通信が必要となる。1 回目は C→S で証明書を送信し C←S でハッシュ値を返す。2 回目は C→S で署名値を送信し C←S で最終結果を返す。なお PAdES ライブラリでは PdaClientXml クラスにて通信 XML の解析や応答 XML の生成が可能になっているが、XAdES ライブラリでは独自に実装が必要となっている。

クライアント	通信	サーバ	独自定義	主内容
証明書 XML 生成 （cert で生成可能）	→(要求)→	証明書 XML 解析 XAdES：独自解析 PAdES：PdaClientXml	LCS_CERT (4.1.1 参照)	証明書
ハッシュ値 XML 解析 （xmlGet で解析可）	←(結果)←	ハッシュ値 XML 生成 XAdES：独自生成 PAdES：PdaClientXml	LCS_HASH (4.1.2 参照)	ハッシュ値
署名値 XML 生成 （sign で生成可能）	→(要求)→	署名値 XML 解析 XAdES：独自解析 PAdES：PdaClientXml	LCS_SIGN (4.1.3 参照)	署名値
結果 XML 解析 （xmlGet で解析可）	←(結果)←	結果 XML 生成 XAdES：独自生成 PAdES：PdaClientXml	LCS_RSLT (4.1.4 参照)	結果

通信に利用する XML の概要

```
// クライアント側（LCS_CERT）：証明書選択 > 証明書 XML
$ LpaCmd.sh -client -cert p12 LeTest.p12 test -sid T01 > CERT.xml
// サーバ側（前準備）：署名フィールド生成
$ LpaCmd.sh -field -in input.pdf -out field.pdf
// サーバ側（LCS_HASH）：仮署名 < 証明書 XML > ハッシュ値 XML
$ LpaCmd.sh -server -in field.pdf -out make.pdf < CERT.xml > HASH.xml
// クライアント側（LCS_SIGN）：署名値計算択 < ハッシュ値 XML > 署名値 XML
$ LpaCmd.sh -client -cert p12 LeTest.p12 test -sid T01 < HASH.xml > SIGN.xml
// サーバ側（LCS_RSLT）：署名値埋め込み < 署名値 XML > 結果 XML
$ LpaCmd.sh -server -in make.pdf -out sign.pdf < SIGN.xml > RSLT.xml
// 結果確認（本来はクライアント側で確認）
$ cat RSLT.xml
```

LpaCmd を使ったクライアント署名の利用例（クライアント側はダミー実行）

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

４．１．１．証明書 XML（LCS_CERT：クライアント→サーバ：step 1）

◎ 証明書 XML（クライアント側で生成）

項目名			種別	必須	説明
LcsRequest			要素	○	XML のルート要素
	sid		属性	○	セッション ID（識別用の接続毎に異なる任意の文字列）
	type		属性	○	通信種別 LCS_CERT を指定。
	time		属性	○	生成日時を YYYYMMDDhhmmssZ 形式で指定
	cert		要素	○	署名証明書の情報
		テキスト要素	要素	○	X.509 形式の証明書を HEX に変換した文字列を格納。

```
<LcsRequest type="LCS_CERT" time="20220819094105Z" sid="U000137984">
  <cert>3082461f06092a864886f70d010702a082461030824…701a0820b6130820329308202110208 5</cert>
</LcsRequest>
```

XML 生成方法	利用例
署名コマンド	var certxml = await cscmd.cert(certflag, certopt);
ActiveX	var certxml = csign.cert(cflag, copt);
LpaCmd 試験	LpaCmd -client -cert x509 LeTest.cer -sid T01 > CERT.xml

４．１．２．ハッシュ値 XML（LCS_HASH：サーバ→クライアント：step 2）

◎ ハッシュ値 XML（サーバ側で生成）

項目名			種別	必須	説明
LcsResponse			要素	○	XML のルート要素
	result		属性	○	処理結果を返す。”OK”かエラーメッセージ（UTF-8）。
	type		属性	○	通信種別 LCS_HASH か LCS_DATA を指定。
	sid		属性	×	セッション ID（確認及びテスト用）
	hashType		属性	×	署名ハッシュ方式 ※ "SHA256" / "SHA384" / "SHA512"、省略時 "SHA256"
	hash / data		要素	○	result が"OK"の場合は署名対象のハッシュ値が返される
		テキスト要素	要素	○	ハッシュ値を HEX に変換した文字列を格納。

```
<LcsResponse result="OK" type="LCS_HASH" sid="U000137984" hashType="SHA256">
  <hash>e5a30c33e718862185dba5f7e204e353a0133f8d</hash>
</LcsResponse >
```

XML 生成方法	利用例
API（C++）	PdaClientXml clientXml; BINARY hashxml = clientXml.createCertResponse(target, hashType);

LpaCmd	// 署名フィールド生成
	LpaCmd -field -in input.pdf -out field.pdf
	// 仮署名実行
	LpaCmd -server -in field.pdf -out make.pdf < CERT.xml > HASH.xml

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

４．１．３．署名値 XML（LCS_SIGN：クライアント→サーバ：step 3）

◎ 署名値 XML（クライアント側で生成）

項目名			種別	必須	説明
LcsRequest			要素	○	XML のルート要素
	sid		属性	○	セッション ID（識別用の接続毎に異なる任意の文字列）
	type		属性	○	通信種別 LCS_SIGN を指定。
	time		属性	○	生成日時を YYYYMMDDhhmmssZ 形式で指定
	sign		要素	○	署名値の情報　通常数キロバイト
		テキスト要素	要素	○	署名値を HEX に変換した文字列を格納。

```
<LcsRequest type="LCS_SIGN" time="20220819094106Z" sid="U000137984">
  <sign>3082460c020101310f300d0609608…060355040313125365637572655369676e20526f6f7443</sign>
</LcsRequest>
```

XML 生成方法	利用例
署名コマンド	var signxml = await cscmd.sign(signflag, signtarget, signalg, signopt);
ActiveX	var signxml = csign.sign(sflag, sopt, signAlg, hash, passwd);
LpaCmd 試験	LpaCmd -client -cert p12 LeTest.p12 test -sid T01 < HASH.xml > SIGN.xml

４．１．４．結果 XML（LCS_RSLT：サーバ→クライアント：step 4）

◎ 結果 XML（サーバ側で生成）

項目名		種別	必須	説明
LcsResponse		要素	○	XML のルート要素
	result	属性	○	処理結果を返す。"OK"かエラーメッセージ（UTF-8）。
	type	属性	○	通信種別 LCS_RSLT を指定。
	sid	属性	×	セッション ID（確認及びテスト用）
	redirect	属性	×	正常終了時のみ指定可能。指定されると処理完了後に移動。 未指定の場合には署名処理完了のメッセージが表示さえる。

```
<LcsResponse result="OK" type="LCS_RSLT" sid="U000137984"
 redirect="http://xxxxx.jp/ok_display"/>
```

XML 生成方法	利用例
API（C++）	PdaClientXml clientXml; BINARY rsltxml = createSignResponse(redirect_url);
LpaCmd	// 署名値の埋め込み LpaCmd -server -in make.pdf -out sign.pdf < SIGN.xml > RSLT.xml

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

４．２．XML 長期署名ライブラリ LE:XAdES:Lib

４．２．１．XAdES サーバ試験コマンド LcsXAdES.exe

サーバを使わず、ローカルのみで試験を行うデバッグ用の実行サーバ LcsXAdES.exe を提供している。ソースコード LeClientSign¥ServerTest¥LcsXAdES¥LcsXAdES.cpp を参照することで、証明書取得時の仮署名と、署名値取得時の署名値埋め込みの、実装例として利用できる。C++/CLI にて実装されているので C# からの利用の参考にもなる。

証明書取得時の仮署名の例：

```
// ------------------------------------------------------------------------
// 証明書取得対応処理
String^ cert(String^ inXml, String^ dir)
{
        XmlNodeList^ list = nullptr;
        XmlElement^ elmt = nullptr;
        XmlElement^ elmt2 = nullptr;
        String^ outXml = nullptr;
        le::LeXAdES^ xades = nullptr;

        // リクエスト XML の解析
        XmlDocument^ doc = gcnew XmlDocument();
        doc->LoadXml(inXml);
        // セッション ID/属性から
        String^ sessionId;
        list = doc->SelectNodes( "/LcsRequest/@sid" );
        if (list->Count <= 0 || list[0] == nullptr)
                return outXml;
        elmt = static_cast<XmlElement^>(list[0]);
        sessionId = elmt->InnerText;
        if (sessionId == nullptr)
                return outXml;
        // 証明書 cert 要素
        array<Byte>^ cert;
        list = doc->SelectNodes( "/LcsRequest/cert" );
        if (list->Count <= 0 || list[0] == nullptr)
                return outXml;
        elmt = static_cast<XmlElement^>(list[0]);
        cert = hex2bin(elmt->InnerText);
        if(cert == nullptr)
                return outXml;

        // XAdES 生成
        String^ idStr = "LcsTest";
        String^ xadesFile = dir + "/" + sessionId + ".xml";
        // インスタンス生成
        xades = gcnew le::LeXAdES(idStr, nullptr, nullptr);
        // 試験用の Enveloping 指定（実務では必要な参照を追加する）
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
        int rc = xades->addEnvelopingXml( "<LangEdge><Debug>XAdES</Debug></LangEdge>" );
        if(rc < 0)
                goto EXIT;
        // 仮署名実行
        unsigned int flags = le::SIGN_FLAG::SIGN_NO_SIGN;
        rc = xades->sign(cert, le::HASH_TYPE::HASH_SHA256, false, flags);
        if(rc < 0)
                goto EXIT;
        // 仮署名結果の保存
        rc = xades->saveXml(xadesFile);
        if(rc < 0)
                goto EXIT;
        // 署名対象の取得
        array<Byte>^ sinfo = xades->getSignedInfo();
        if(sinfo == nullptr)
                goto EXIT;
        // 署名対象のハッシュ値計算
        HashAlgorithm^ hashAlg = gcnew SHA256Managed();
        array<Byte>^ hash = hashAlg->ComputeHash(sinfo);
        if(hash == nullptr)
                goto EXIT;

        // レスポンス XML の生成
        doc = gcnew XmlDocument();
        elmt = doc->CreateElement("LcsResponse");       // ルート要素
        elmt->SetAttribute("type", "LCS_HASH");         // 属性：種別
        elmt->SetAttribute("result", "OK");             // 属性：結果正常
        elmt->SetAttribute("hathType", "SHA256");       // 属性：ハッシュ方式：省略可能
        elmt->SetAttribute("sid", sessionId);           // 属性：セッション ID：省略可能
        elmt2 = doc->CreateElement("hash");             // hash 子要素
        elmt2->InnerText = bin2hex(hash);               // ハッシュ値をセット
        elmt->AppendChild(elmt2);
        doc->AppendChild(elmt);
        outXml = doc->OuterXml;

EXIT:
        if(xades != nullptr)
                delete xades;
        xades = nullptr;
        return outXml;
}
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

署名値取得時の署名値埋め込みの例：

```
// ------------------------------------------------------------------------
// 署名値計算対応処理
String^ sign(String^ inXml, String^ dir)
{
        XmlNodeList^ list = nullptr;
        XmlElement^ elmt = nullptr;
        String^ outXml = nullptr;
        le::LeXAdES^ xades = nullptr;

        // リクエスト XML の解析
        XmlDocument^ doc = gcnew XmlDocument();
        doc->LoadXml(inXml);
        // セッション ID/属性から
        String^ sessionId;
        list = doc->SelectNodes( "/LcsRequest/@sid" );
        if (list->Count <= 0 || list[0] == nullptr)
                return outXml;
        elmt = static_cast<XmlElement^>(list[0]);
        sessionId = elmt->InnerText;
        if (sessionId == nullptr)
                return outXml;
        // 署名値 sign 要素
        array<Byte>^ value;
        list = doc->SelectNodes( "/LcsRequest/sign" );
        if (list->Count <= 0 || list[0] == nullptr)
                return outXml;
        elmt = static_cast<XmlElement^>(list[0]);
        value = hex2bin(elmt->InnerText);
        if(value == nullptr)
                return outXml;

        // XAdES 生成
        String^ xadesFile = dir + "/" + sessionId + ".xml";
        String^ xadesUrl = "file://" + xadesFile;
        // インスタンス生成
        xades = gcnew le::LeXAdES(nullptr, nullptr, nullptr);
        // 仮署名 XAdES の読み込み
        int rc = xades->loadXml(xadesFile);
        if(rc < 0)
                goto EXIT;
        // 署名値のセット
        rc = xades->setSignatureValue(value);
        if(rc < 0)
                goto EXIT;
        // 署名結果の保存
        rc = xades->saveXml(xadesFile);         // 上書き
        if(rc < 0)
                goto EXIT;
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
        // レスポンス XML の生成
        doc = gcnew XmlDocument();
        elmt = doc->CreateElement("LcsResponse");       // ルート要素
        elmt->SetAttribute("type", "LCS_RSLT");         // 属性：種別
        elmt->SetAttribute("result", "OK");             // 属性：結果正常
        elmt->SetAttribute("sid", sessionId);           // 属性：セッション ID：省略可能
        elmt->SetAttribute("redirect", xadesUrl);       // 属性：リダイレクト：省略可能
        doc->AppendChild(elmt);
        outXml = doc->OuterXml;

EXIT:
        if(xades != nullptr)
                delete xades;
        xades = nullptr;
        return outXml;
}
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

４．３．PDF 長期署名ライブラリ LE:PAdES:Lib

PAdES を利用する場合には、LE:PAdES:Lib または LE:PAdES-Basic:Lib の V1.05.R2（2017 年 5 月リリース）以上が必要となるので注意が必要である。

４．３．１．PAdES サーバ試験コマンド LcsPAdES.exe

サーバを使わず、ローカルのみで試験を行うデバッグ用の実行サーバ LcsPAdES.exe を提供している。ソースコード LeClientSign¥ServerTest¥LcsPAdES¥LcsPAdES.cpp を参照することで、証明書取得時の仮署名と、署名値取得時の署名値埋め込みの、実装例として利用できる。C++の API にて実装されている。

証明書取得時の仮署名の例：

```
// ------------------------------------------------------------------------
// 証明書取得対応処理
std::string cert(le::PdaClientXml& inXml, std::string& dir)
{
        int rc = 0;
        std::string outXml, inputPdf, fieldName, outputPdf;
        le::LpkCert cert;
        le::PdaField field;
        le::LePAdES pades;
        le::BINARY certBin, hash, outBin;

        // 試験設定
        fieldName = "LCS1";
        inputPdf = dir + "input.pdf";
        outputPdf = dir + inXml.getSessionId().c_str() + "-temp.pdf";

        // 署名証明書の準備
        certBin = inXml.getData();
        rc = cert.setBin(certBin, false);
        if(rc < 0)
                return outXml;

        // 元 PDF ファイルの読み込み
        rc = pades.loadFile(inputPdf.c_str(), NULL, NULL);
        if(rc < 0)
                return outXml;

        // 署名フィールドの指定
        field.setName(fieldName.c_str());
        rc = pades.addField(field);
        if(rc < 0)
                return outXml;

        // V2 クライアント署名用の仮署名
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
        rc = pades.makeEnhanced2(fieldName.c_str(), cert, hash);
        if(rc < 0)
                return outXml;

        // 仮署名 PDF ファイルの保存
        rc = pades.saveFile(outputPdf.c_str());
        if(rc < 0)
                return outXml;

        // 応答 XML の生成
        outBin = inXml.createCertResponse(hash);
        if(outBin.empty())
                return outXml;
        outXml.assign((const char*)&outBin[0], outBin.size());

        return outXml;
}
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

４．４．サーバ試験コマンド LcsXAdES/LcsPAdES（オプション）

試験用のサーバを用意するのは大変なので、ローカルでサーバの動作をするサーバ試験コマンドを提供している。GUI を持たないコマンドプログラムになっており、引数として証明書を取得して仮署名を行う（-cert）か、仮署名ファイルに署名値を埋め込み署名ファイルを完成させる（-sign）の種別が選べる。標準入力にクライアント生成の独自 XML をセットすると標準出力にサーバ生成の独自 XML が返される。

XAdES 用（LcsXAdES.exe）と PAdES 用（LcsPAdES.exe）の 2 種類が提供される。どちらも必要となる DLL 群と共に ServerTest/bin の下にあり、同じフォルダ内に仮署名された中間ファイルと最終的な署名ファイルが生成される。仮署名ファイルと署名ファイルのファイル名にはセッション ID 名が使われる。

例：LcsXAdES.exe -[cert/sign] < [入力 XML] > [出力 XML]
例：LcsPAdES.exe -[cert/sign] < [入力 XML] > [出力 XML]

これらサーバ試験コマンドの実装ソースが、ServerTest/LcsXAdES と ServerTest/LcsPAdES の下にあるので、サーバ構築時の参考に利用可能となっている。

サーバ試験コマンドを利用するには独自 HTTP 通信の httpSet 引数のホスト名において、"debug://LcsXAdES.exe" または "debug://LcsPAdES.exe" を指定し、httpSend 引数のパス指定に "△-cert" または "△-sign " を指定する。△は空白文字を示す。

サーバ試験コマンド	利用ライブラリ	概要	章
LcsXAdES.exe	LE:XAdES:Lib	Enveloping 1 つ参照した XAdES 署名生成	3.1.1
LcsPAdES.exe	LE:PAdES:Lib	input.pdf に対してい PAdES 署名付与	3.2.1

サーバ試験コマンドの種類

なおクライアント側は XAdES/PAdES で共通化されているので、1 つのモジュール（ActiveX 等）で両方に対応が可能である。XAdES を利用するか PAdES を利用するかはサーバ側実装にて選択される。

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

署名値取得時の署名値埋め込みの例：

```
// ------------------------------------------------------------------------
// 署名値計算対応処理
std::string sign(le::PdaClientXml& inXml, std::string& dir)
{
        int rc = 0;
        std::string outXml, inputPdf, fieldName, outputPdf, outputUrl;
        le::LePAdES pades;
        le::LpkCades cades;
        le::BINARY value, cadesBin, outBin;

        // 試験設定
        fieldName = "LCS1";
        inputPdf = dir + inXml.getSessionId().c_str() + "-temp.pdf";
        outputPdf = dir + inXml.getSessionId().c_str() + ".pdf";
        outputUrl = "file://" + outputPdf;

        // 署名値の取得
        value = inXml.getData();

        // 仮署名 PDF ファイルの読み込み
        rc = pades.loadFile(inputPdf.c_str(), NULL, NULL);
        if(rc < 0)
                return outXml;

        // 署名値の埋め込み
        rc = pades.setSignValue(fieldName.c_str(), &value[0], value.size());
        if(rc < 0)
                return outXml;

        // 署名済み PDF ファイルの保存
        rc = pades.saveFile(outputPdf.c_str());
        if(rc < 0)
                return outXml;

        // 応答 XML の生成
        outBin = inXml.createSignResponse(outputUrl.c_str());
        if(outBin.empty())
                return outXml;
        outXml.assign((const char*)&outBin[0], outBin.size());

        return outXml;
}
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

付録１．エラーコード一覧

付録１．１．LCS_ERR：基本エラーコード

番号	定義名	意味
0	LCS_OK	エラー無し（正常終了・初期値）
1	LCS_CANCEL	キャンセル（ダイアログ等でキャンセルが選択された）
共通エラー		
-10	LCS_ARG_ERR	引数エラー
-11	LCS_NO_CERT_ERR	証明書が見つからない
-12	LCS_NO_PKEY_ERR	秘密鍵が見つからない
-13	LCS_INVALID_CERT_ERR	証明書が正しくない
-14	LCS_UNKN_HASH_ERR	指定されたハッシュアルゴリズムが未サポートか異常
-15	LCS_INVALID_HASH_ERR	指定されたハッシュ値の異常
-16	LCS_CALC_HASH_ERR	ハッシュ値の計算エラー
-17	LCS_INVALID_DATA_ERR	指定された署名対象データの異常
-18	LCS_INVALID_HEX_ERR	指定された HEX データの異常
-20	LCS_INET_HTTP_ERR	HTTP 通信エラー
-21	LCS_XML_GEN_ERR	XML 生成エラー
-22	LCS_DEBUG_EXE_ERR	デバッグコマンド実行エラー
-23	LCS_HEX_GEN_ERR	HEX 文字列生成エラー
-24	LCS_B64_GEN_ERR	Base64 文字列生成エラー
PKCS#11 エラー		
-30	LCSP_INIT_ERR	PKCS#11 の初期化エラー（ロードエラー以外の理由）
-31	LCSP_LOAD_ERR	PKCS#11 の DLL ファイルのロードエラー
-32	LCSP_GET_FUNC_ERR	PKCS#11 の C_GetFunctionList のポインタ取得に失敗
-33	LCSP_NULL_FUNC_ERR	PKCS#11 のファンクションが NULL 定義だった
-34	LCSP_PASSWD_ERR	PKCS#11 の利用パスワードエラー
-35	LCSP_UNKN_ALG_ERR	PKCS#11 の暗号アルゴリズムが未サポートか異常
-36	LCSP_REMOVED_ERR	PKCS#11 の IC カード未挿入状態エラー
CAPI エラー		
-50	LCSC_INIT_ERR	CAPI の初期化エラー
-51	LCSC_INVALID_PROV_ERR	CAPI のプロバイダエラー
-52	LCSC_CALC_SIGN_ERR	CAPI の署名実行エラー
-53	LCSC_SELECT_ERR	CAPI の証明書選択ダイアログエラー
-54	LCSC_STORE_ACCESS_ERR	CAPI の証明書ストアアクセスエラー ※ IE が保護モードの可能性あり
署名コマンドエラー		

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

-60	LCSS_OPEN_ERR	署名コマンドの通信初期化エラー
-61	LCSS_LOOP_ERR	署名コマンドの通信取得エラー
-62	LCSS_RESPONSE_ERR	署名コマンドの通信送信エラー
ActiveX エラー		
-70	LCSA_INIT_ERR	ActiveX の初期化エラー
-71	LCSA_REDIRECT_ERR	ActiveX のリダイレクトエラー
システムエラー		
-90	LCS_NO_SUPPORTED	現在未サポートの機能を使った
-91	LCS_VERSION_ERR	サーバ側ライブラリとのバージョン不一致
-97	LCS_SYSTEM_ERR	不明なシステムエラー
-98	LCS_EXCEPTION_ERR	不明な例外
-99	LCS_UNKNOWN_ERR	その他不明なエラー

付録１．２．LCSX_ERR：XAdES エラーコード

番号	定義名	意味
0	LCSX_NOERROR	エラー無し（正常終了・初期値）
Detached/Enveloping エラー		
-110	LCSX_DETACH_OPEN	ファイルが存在しないか、添付ファイルの読込みでエラーが発生した
-111	LCSX_DETACH_ARG	添付ファイル名の先頭が「#」だった
-112	LCSX_DETACH_EXIST	添付ファイルに同名のファイルか ID が指定された
-113	LCSX_DETACH_EMPTY	添付ファイルに指定されたファイルが 0 バイトだった
XAdES 署名エラー		
-130	LCSX_DATA_ARG	署名対象のエラー
-131	LCSX_HASH_ALG	未サポートのハッシュ方式が指定された
-132	LCSX_SIGN_CERT	署名証明書が開けない
-133	LCSX_C14N_TRANS	C14N 正規化に失敗した
-134	LCSX_DETACHED	Detached 指定のエラー
-135	LCSX_XMLDSIG	XML 署名生成エラー
-136	LCSX_HASH_SHA1	SHA-1 署名された証明書が指定された
-137	LCSX_XML_DOC	DOM の XML ドキュメントエラー
-138	LCSX_XML_NO_SET	対象 XML が未設定だった
-139	LCSX_XML_ADD_SIGN	Signature 要素の追加エラー
-140	LCSX_XML_OUT	結果 XML 取得エラー
-141	LCSX_ENVELOPED	Enveloped 指定のエラー
-142	LCSX_SINGED_XML	署名済みのエラー
-143	LCSX_NO_SIGN_EXEC	連続署名時に致命的エラーの為に署名実行されなかった

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

-144	LCSX_ENVELOPING	Enveloping 指定のエラー
XAdES 検証エラー（現在未サポート）		
-150	LCSX_VERIFY_SIGN	署名値の不一致（改ざん）
-151	LCSX_VERIFY_DT_HASH	参照情報のハッシュ不一致（改ざん）
-152	LCSX_VERIFY_DT_NONE	参照情報が見つからない
-153	LCSX_VERIFY_CERT	署名証明書が見つからないか異常がある
-154	LCSX_VERIFY_PATH	署名証明書の認証パスが生成できない
-155	LCSX_VERIFY_TIME	署名証明書の有効期限切れ
-156	LCSX_VERIFY_FORMAT	署名フォーマットが異常（必要な要素が無い等）
-157	LCSX_VERIFY_NOSIGN	未署名のファイルだった

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

付録２．証明書情報一覧（info() が返す JSON/XML 情報）

タグ名			説明
(CERT)			ルートタグ名、JSON では省略
	type		種別：‘CERT’(一般証明書) / ‘ROOT’(ルート) / ‘JPKI’ / ‘REGIST’(商業登記)
	serial		シリアル番号 HEX 文字列
	from		有効期間開始 形式：yyyy-mm-ddThh:mm:ss（ローカル時刻）
	to		有効期間終了 形式：yyyy-mm-ddThh:mm:ss（ローカル時刻）
	issuer		発行者：X.500 識別名
		cn	共通名 "2.5.4.3"
		o	組織名 "2.5.4.10"
		ou	部門名 "2.5.4.11"
		l	局所性(都道府県/区) "2.5.4.7"
		st	州または地域 "2.5.4.8"
		sn	苗字(姓) "2.5.4.4"
		givenname	名前(名) "2.5.4.42"
		street	通り "2.5.4.9"
		title	名称 "2.5.4.12"
		serialnumber	シリアル番号 "2.5.4.5"
		uid	ユニーク ID "2.5.4.45"
		c	国 "2.5.4.6"
		oidX.X.X.X	その他の OID 例："oid2.5.4.17"
	subject		発行先：X.500 識別名
		---	issuer と同じ内容
	jpki		JPKI：拡張部(マイナンバーカードの時のみ)
		name	氏名：JIS 第 1 水準、第 2 水準、補助漢字以外の文字は代替文字に変換
		address	住所：JIS 第 1 水準、第 2 水準、補助漢字以外の文字は代替文字に変換
		birth	生年月日 EYYYYMMDD: E(年号コード) 1:明治/2:大正/3:昭和/4:平成/0:不明
		gender	性別 1：男、2：女、3：不明
		scname	氏名：代替文字使用位置を示す数字の文字列: 0 代替文字でない / 1 代替文字
		scaddress	住所：代替文字使用位置を示す数字の文字列: 0 代替文字でない / 1 代替文字
	regist		REGIST：拡張部(商業登記証明書の時のみ)
		corp	法人名（商号）
		address	本店住所
		name	代表者名
		title	肩書
		number	会社法人等番号（12 桁：法人番号は先頭にチェックデジット 1 桁追加）
		issue	管轄登記所

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

※ 一般の証明書情報の JSON 例

```
{
  "type":"CERT",
  "serial":"0133XXXXXXXXXXX",
  "from":"20XX-10-26T00:00:00",
  "to":"20XX-10-25T23:59:59",
  "issuer":{
    "c":"JP",
    "o":"e-Xxxxxxxx CA",
    "ou":"e-Xxxxxxxx PS"
  },
  "subject":{
    "c":"JP",
    "cn":"Xxxxxx Xxxxxx"
  }
}
```

※ 商業登記証明書情報の JSON 例

```
{
  "type":"REGIST",
  "serial":"072C75FXXXXXXX",
  "from":"20XX-12-13T10:38:31",
  "to":"20XX-03-13T23:59:59",
  "issuer":{
    "c":"JP",
    "o":"Japanese Government",
    "ou":"Ministry of Justice",
    "cn":"Registrar of Tokyo Legal Affairs Bureau"
  },
  "subject":{
    "c":"JP",
    "o":"MOJ No.0100XXXXXXXX",
    "cn":"040101XXXXXXX-kaisataro"
  },
  "regist":{
    "number":"0100XXXXXXXX",
    "issue":"東京法務局",
    "corp":"株式会社〇〇〇〇",
    "address":"東京都千代田区〇〇町三丁目１番地１号",
    "name":"会社太郎",
    "title":"代表取締役"
  }
}
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

※ 公的個人証明書情報の JSON 例

```
{
  "type":"JPKI",
  "serial":"703XXX",
  "from":"20XX-03-15T03:38:08",
  "to":"20XX-04-01T23:59:59",
  "issuer":{
    "c":"JP",
    "o":"JPKI",
    "ou":"JPKI for digital signature",
    "ou":"Japan Agency for Local Authority Information Systems"
  },
  "subject":{
    "c":"JP",
    "l":"Tokyo-to",
    "l":"Edogawa-ku",
    "cn":"20XXXXXXXXXXXXXXXXXXXXXX03A"
  },
  "jpki":{
    "name":"○○　○○",
    "scname":"00000",
    "address":"東京都江戸川区○○町１丁目２３番４５号",
    "scaddress":"0000000000000000000",
    "birth":"319810401",
    "gender":"1"
  }
}
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

※ 一般の証明書情報の XML 例

```
<?xml version="1.0" encoding="utf-8"?>
<CERT>
  <type>CERT</type>
  <serial>0133XXXXXXXXXXX</serial>
  <from>20XX-10-26T00:00:00</from>
  <to>20XX-10-25T23:59:59</to>
  <issuer>
    <c>JP</c>
    <o>e-Xxxxxxxx CA</o>
    <ou>e-Xxxxxxxx PS</ou>
  </issuer>
  <subject>
    <c>JP</c>
    <cn>Xxxxxx Xxxxxx</cn>
  </subject>
</CERT>
```

※ 商業登記証明書情報の XML 例

```
<?xml version="1.0" encoding="utf-8"?>
<CERT>
  <type>REGIST</type>
  <serial>072C75F6C75B3F</serial>
  <from>20XX-12-13T10:38:31</from>
  <to>20XX-03-13T23:59:59</to>
  <issuer>
    <c>JP</c>
    <o>Japanese Government</o>
    <ou>Ministry of Justice</ou>
    <cn>Registrar of Tokyo Legal Affairs Bureau</cn>
  </issuer>
  <subject>
    <c>JP</c>
    <o>MOJ No.0100XXXXXXXX</o>
    <cn>040101XXXXXXX-kaisataro</cn>
  </subject>
  <regist>
    <number>0100XXXXXXXX</number>
    <issue>東京法務局</issue>
    <corp>株式会社〇〇〇〇</corp>
    <address>東京都千代田区〇〇町三丁目１番地１号</address>
    <name>会社太郎</name>
    <title>代表取締役</title>
  </regist>
</CERT>
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

※ 公的個人証明書情報の XML 例

```
<?xml version="1.0" encoding="utf-8"?>
<CERT>
  <type>JPKI</type>
  <serial>703XXX</serial>
  <from>20XX-03-15T03:38:08</from>
  <to>20XX-04-01T23:59:59</to>
  <issuer>
    <c>JP</c>
    <o>JPKI</o>
    <ou>JPKI for digital signature</ou>
    <ou>Japan Agency for Local Authority Information Systems</ou>
  </issuer>
  <subject>
    <c>JP</c>
    <l>Tokyo-to</l>
    <l>Edogawa-ku</l>
    <cn>20XXXXXXXXXXXXXXXXXXXXXXX03A</cn>
  </subject>
  <jpki>
    <name>○○　○○</name>
    <scname>00000</scname>
    <address>東京都江戸川区○○町１丁目２３番４５号</address>
    <scaddress>00000000000000000000</scaddress>
    <birth>319810401</birth>
    <gender>1</gender>
  </jpki>
</CERT>
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

付録３．XAdES-BES 仕様（xades() が返す XML 情報）

XML 要素名	主な属性/項目 ※補足
<Signature/>	Id="LEXDS01" xmlns="xmldsig#"
<SignedInfo/>	※ 署名値の対象
<CanonicalizationMethod/>	Algorithm="REC-xml-c14n-20010315"
<SignatureMethod/>	Algorithm="xmldsig-more#rsa-sha256"
<Reference/>	※ 署名対象の指定（必ず１つはある）
<Transforms/>	※ 変形指定（オプション）
<Transform/>	Algorithm="REC-xml-c14n-20010315" Algorithm="REC-xpath-19991116" 等
<Xpath/>	※Xpath 指定（オプション）
<DigestMethod/>	Algorithm="xmlenc#sha256" 等指定アルゴリズム
<DigestValue/>	※ Reference 先のハッシュ値
<Reference/>	※ Detached/Enveloping の指定（オプション）
<Reference/>	Id="LEXDS01-xades-ref" URI="#LEXDS01-xades" Type="#SignedProperties"
<Transforms/>	※ 変形指定（C14N）
<Transform/>	Algorithm="REC-xml-c14n-20010315"
<DigestMethod/>	Algorithm="xmlenc#sha256" 等指定アルゴリズム
<DigestValue/>	※ XAdES 用 Reference 先のハッシュ値
<SignatureValue/>	※ <SignedInfo/>の署名値
<KeyInfo/>	※ 鍵情報
<X509Data/>	※ X.509 形式の情報
<X509Certificate/>	※ 署名証明書
<X509Certificate/>	※ 認証パス証明書（ルート等がある場合のみ）
<Object/>	※長期署名用オブジェクト
<QualifyingProperties/>	xmlns="01903/v1.3.2#" ※XAdES 要素
<SignedProperties/>	Id="LEXDS01-xades" ※XAdES 署名要素
<SignedSignatureProperties/>	※ 署名プロパティ群
<SigningTime/>	※ 署名時刻
<SigningCertificateV2/>	※ 署名証明書 V2
<Cert/>	※ 証明書情報
<CertDigest/>	※ 署名証明書のハッシュ値情報
<DigestMethod/>	Algorithm="xmlenc#sha256" 等指定アルゴリズム
<DigestValue/>	※ 署名証明書のハッシュ値
<IssuerSerialV2/>	※ 発行者の DER 情報とシリアル番号を連結

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

※ 生成される XAdES-BES の XML 例

```
<?xml version="1.0" encoding="UTF-8"?>
<TestXml>
 <Contents Id="TEST01">
  <参考>日本語</参考>
  <PID>WA7H</PID>
 </Contents>
 <Signature xmlns="http://www.w3.org/2000/09/xmldsig#" Id="LEXDS01">
  <SignedInfo>
   <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
   <SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256"/>
   <Reference URI="#TEST01" Id="LEXDS01-ref-0">
    <Transforms>
     <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    </Transforms>
    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
    <DigestValue>9UX1cZ6bvqvKLC8jzS3UCB3x5rLm+Ms1NO/Y5mE+vSg=</DigestValue>
   </Reference>
   <Reference URI="#LEXDS01-xades" Id="LEXDS01-xades-ref"
              Type="http://uri.etsi.org/01903#SignedProperties">
    <Transforms>
     <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    </Transforms>
    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
    <DigestValue>MoAwXOYjkmOofIzTN4nFokgGdeKsBPYFAva5NCEt+po=</DigestValue>
   </Reference>
  </SignedInfo>
  <SignatureValue>Ulwoiss(略)JgV+4k1Q==</SignatureValue>
  <KeyInfo>
   <X509Data>
    <X509Certificate>MIIGvTCC(略)umk7Knza23TQ==</X509Certificate>
    <X509Certificate>MIIECzCC(略)BDWO6QsEwd9A==</X509Certificate>
   </X509Data>
  </KeyInfo>
  <Object Id="LEXDS01-xades-obj">
   <QualifyingProperties xmlns="http://uri.etsi.org/01903/v1.3.2#" Target="#LEXDS01">
    <SignedProperties Id="LEXDS01-xades">
     <SignedSignatureProperties>
      <SigningTime>2020-11-30T12:36:49Z</SigningTime>
      <SigningCertificateV2>
       <Cert>
        <CertDigest>
         <DigestMethod xmlns="http://www.w3.org/2000/09/xmldsig#"
                       Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
         <DigestValue xmlns="http://www.w3.org/2000/09/xmldsig#">
          bOedXAvORY49arH4Syo/g/GpsKxFhAOcg971igSaow8=
         </DigestValue>
        </CertDigest>
        <IssuerSerialV2>MIGKMH+kfT(略)gcHLHX2x1s/</IssuerSerialV2>
       </Cert>
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

```
      </SigningCertificateV2>
     </SignedSignatureProperties>
    </SignedProperties>
   </QualifyingProperties>
  </Object>
 </Signature>
</TestXml>
```

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

◆ 製品についてのお問合せ窓口
support@langedge.jp / TEL 03-3862-2268 / 担当：宮地

有限会社ラング・エッジ

〒101-0032 東京都千代田区岩本町２-１４-１２ 宮崎ビル２Ｆ
TEL 03-3862-2268 web http://www.langedge.jp/

以上

Copyright © 2012-2026 LangEdge,Inc. All rights reserved.