ラング・エッジ XML 長期署名ライブラリ

LE:XAdES:Lib V3 マニュアル

２０２６年７月１日版

□ 目次

□ 目次 ……1
○ 履歴 ……3
１．概要……4
１．１．全体構成と長期署名 ……6
１．２．署名対象の指定 ……8
１．３．API 環境 ……14
１．４．評価版 ……15
１．５．動作環境 ……16
１．６．動作に必要なファイル ……17
１．７．Windows 環境のインストールとソースビルド ……18
１．８．Linux 環境のインストールとソースビルド ……20
１．９．内部プロジェクト構成 ……24
１．１０．利用外部ライブラリ ……25
１．１１．フォルダ構成 ……28
１．１２．文字コードと LeString クラス ……29
２．ＡＰＩ利用方法 ……30
２．１．API リファレンス ……30
２．２．Windows C++ 利用の場合……41
２．３．Linux C++ 利用の場合 ……42
２．４．Java 利用の場合 ……43
２．５．.NET API 利用の場合 ……45
２．６．Lx3Cmd コマンドの利用例……47
３．ＸＭＬ電子署名とＸＡｄＥＳ長期署名 ……53
３．１．XML 署名（XmlDsig） ……53
３．２．XAdES 長期署名……54
３．３．XAdES オプション要素……59
３．４．Manifest 要素の指定 ……63
３．５．複数署名 ……65
３．６．署名検証 ……67
３．７．長期署名の運用 ……70
３．８．XML スキーマチェック ……71
４．ＰＫＩ要素と電子署名付与 ……73
４．１．独自証明書ストア ……73
４．２．署名鍵の置き場所による分類 ……74
４．３．仮署名（HSM の利用等） ……76
４．４．証明書検証サーバー（CVS）の利用：GPKI/LGPKI 等……78
４．５．検証結果 XML 仕様（XdaVerifyXml） ……84
４．６．タイムスタンプ取得エラーのデバッグ ……89

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

５．クライアント署名（Ver2.1） ……94
５．１．ブラウザ（ActiveX プラグインと署名コマンド）の選択 ……95
５．２．CAPI（CryptoAPI）と PKCS#11 の選択 ……96
５．３．署名サーバー連携 ……97
５．４．署名サーバー実装例 ……98
５．５．LE:Client:Sign フォルダ構成 ……99
付録Ａ．エラーコード …… 100
Ａ．１．LeUtil エラーコード（src/LeCommon/LeUtil.h）-1000 ～ -1099 …… 100
Ａ．２．Lx3Cmd エラーコード（src/LeXAdES3/Lx3Cmd/Lx3Cmd.h）-1200 ～ -1399 …… 100
Ａ．３．LePKI エラーコード（include/LePKI/LePKI.h）-3000 ～ -3999 …… 101
Ａ．４．LeXAdES3 エラーコード（include/LeXAdES3/LexError.h）-5000 ～ -5999 …… 103
付録Ｂ．適合宣言（Declaration of Conformity） …… 106
Ｂ．１．Version number of ETSI EN 319 132-2 to be referenced …… 106
Ｂ．２．Scope of profile implementation …… 106
Ｂ．３．Conformity to the XAdES-T profile …… 106
Ｂ．４．Conformity to the XAdES-X-Long form …… 108
Ｂ．５．Conformity to the XAdES-A profile …… 109
Ｂ．６．Specifications to be referenced by elements "Conditional" …… 109
付録Ｃ．制限事項・履歴 ……110
付録Ｄ．コピーライト表示 ……111

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

○ 履歴

バージョン	日付	主な修正内容
Ver 3.02.R1	2026/07/01	Windows 開発環境に Visual Studio 2026 を追加して、2015 を削除。 LE:PKI:Lib にて商業登記リモート署名に対応。
Ver 3.01.R3	2025/08/27	「1. 概要」に「e シール（電子シール）の利用について」と「ECDSA（楕円曲線）暗号方式の利用について」を追加。 「4.5. 検証結果 XML 仕様」へ "error"要素のメッセージを追加。 「4.6. タイムスタンプ取得エラーのデバッグ」を追加。
Ver 3.01.R2	2025/01/27	「1. 概要」に LE:PKI:Lib Ver 1.09.R2 の注意点を追加
Ver 3.00.R2	2024/01/22	「3.3. XAdES オプション属性」の記載漏れ部分を執筆 「3.8. XML スキーマチェック」の追加 「4.5. 検証結果 XML 仕様（XdaVerifyXml)」の追加
Ver 3.00.R1	2023/08/30	LE:XAdES:Lib V3 マニュアルの第 1 版

- 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。最新情報はラング・エッジの製品ページで確認できます。
- 製品ページ　http://www.langedge.jp/biz/LeXAdES/index.html

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

１．概要

XAdES（ETSI EN 319 132・ETSI TS 101 903）とは長期保管に対応した XML 電子署名（W3C XML-DSig）の仕様名称であり、ISO 14533-2:2021（XAdES V1.4.1）および JIS X5093 の 2 つのプロファイル（XAdES V1.3.2、仕様としては古いので注意）が存在する。本書はラング・エッジの XML 長期署名ライブラリである、「XML 長期署名ライブラリ LE:XAdES:Lib V3（以後 LE:XAdES:Lib V3）」の製品マニュアルである。なお LE:XAdES:Lib は過去に V1/V2（.NET 利用 Windows 版のみ）があったが、V3 では PDF 長期署名ライブラリ LE:PAdES:Lib で利用している PKI ライブラリ LE:PKI:Lib（Windows 版/Linux 版）を利用している。V3 では API 仕様を見直し整理した為に、V1 および V2 と V3 の間では API の互換性は無い。また LE:PKI:Lib は LE:PAdES:Lib と共有しているので PKI 部に関しては LE:PAdES:Lib との API の互換性がある。

LE:XAdES:Lib	概要	サポート
V3.0	LE:PKI:Lib（OpenSSL）を利用し Windows /Linux の両方に対応（ダイナミックライブラリ、DLL/so）。**V1 および V2 との API 互換性は無い。**本書の対象バージョン。 ※ Windows 版と Linux 版、Java/.NET/C++/コマンド	今後新規機能は V3 に対して追加される。メンテナンスサポートも行われる。
V2.0	一部 OpenSSL を利用し Windows ダイナミックライブラリ（DLL）化、V1 との API 互換性あり。 ※ Windows 版のみ、.NET/C++/コマンド	**今後はメンテナンスサポートのみ**の予定（サポート継続）。
V1.0	.NET 利用の Windows スタティックライブラリ。 ※ Windows 版のみ、.NET/コマンド	**現在未サポート。**（V2 移行推奨）

ラング・エッジ XML 署名製品履歴

1	V3：Windows 版に加えて Linux 版も提供	V2：Windows 版のみ
2	V3：.NET とコマンド以外に Java と C++ の API も提供	V2：.NET と C++/CLI のみ
3	V3：同一 XML ファイル内の複数署名に対応	V2：1 ファイル 1 署名のみ
4	V3：多階層の Manifest 構造に対応	V2：1 階層のみ対応
5	V3：PKI 機能に LE:PKI:Lib を利用（LE:PAdES:Lib 共通化）	V2：独自 PKI 実装（.NET）

V3 の V2 からの大きな変更点

※ XAdES のレベルは、ISO 14533-2:2021 で定義している XAdES-BES/ XAdES-T/ XAdES-X-Long/ XAdES-A にのみ対応している。XAdES-C/XAdES-X 等はほぼ利用されていない仕様の為。

これまでの V1/V2 およびラング・エッジの既存製品である「PDF 長期署名ライブラリ LE:PAdES:Lib」と同じく、LE:XAdES:Lib V3 も、保守目的で利用可能な全てのソースコードとプロジェクトのファイルが付属する。ただしライセンスはオープンソースライセンスでは無く、商用ライセンスでの提供である。なお保守目的の場合には修正が可能。

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

○ e シール（電子シール）の利用について

「XML 長期署名ライブラリ LE:XAdES:Lib V3」では、e シール（eSeal）をサポートしている。e シールとは署名証明書の発行先が自然人では無く組織になっている電子証明書によるデジタル署名となる。デジタル署名を利用するために技術的には電子署名と全く同じであるので、本製品を利用して e シールを付与することおよび検証をすることが可能となっており、また e シールを使った長期署名にも対応している。

○ ECDSA（楕円曲線）暗号方式の利用について

V3.01.R2 より RSA 署名アルゴリズム以外に ECDSA 署名アルゴリズムに対応した。対応しているカーブ（曲線）は、NIST P-256/P-384/P-521 となる。

○ LE:PKI:Lib Ver1.09.R2 の仕様変更に関する注意点

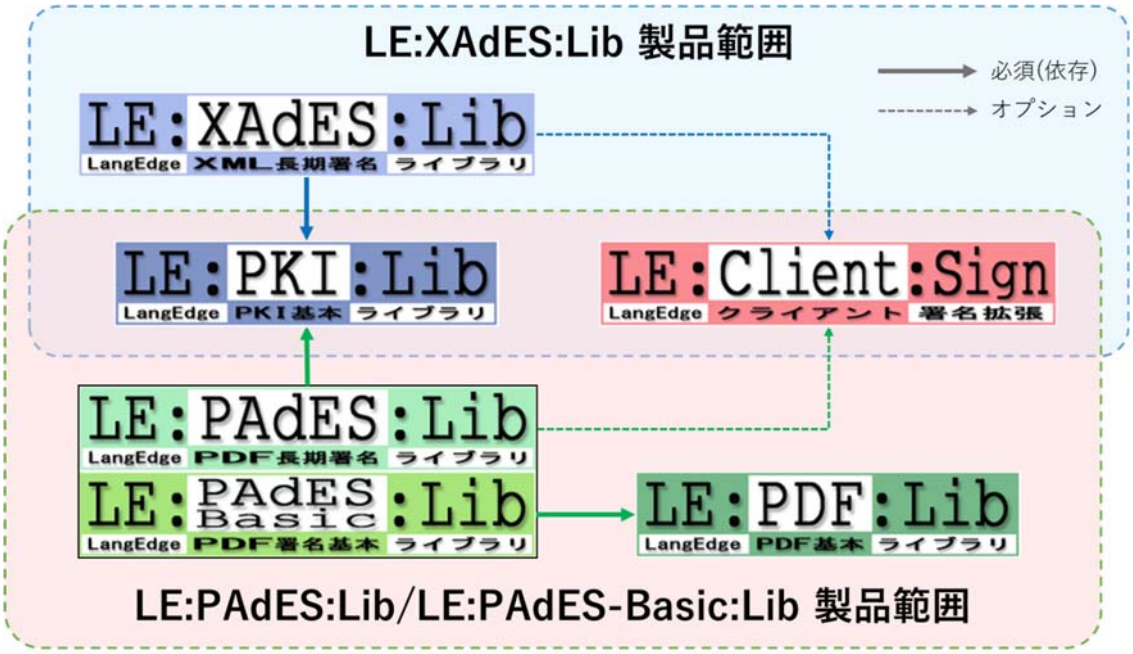
LE:PKI:Lib の Ver1.09.R2（2025 年 1 月リリース）では Windows 環境の HTTP 通信において、proxy.ini 設定ファイルの指定が可能となった。詳しくは LE:PKI:Lib マニュアル（LePKI-manual.pdf）の「４．２．HTTP 通信のプロキシ設定」の章を参照。

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

１．１．全体構成と長期署名

XML 長期署名ライブラリ LE:XAdES:Lib V3 は、大きく分けて XML 長期署名を行う LE:XAdES:Lib と、暗号と PKI の操作を行う LE:PKI:Lib から構成される。またオプション製品としてサーバー上の LE:XAdES:Lib と連携してクライアント（Windows）上のブラウザ（Edge/Chrome）と連携して署名鍵/証明書を利用するクライアント署名拡張 LE:Clinet:Sign の利用も可能となっている。

ラング・エッジの電子署名製品としては PDF 長期署名ライブラリ LE:PAdES:Lib も提供されている。LE:XAdES:Lib V3 以降は、LE:PKI:Lib と LE:Client:Sign は LE:PAdES:Lib と共通して利用される。



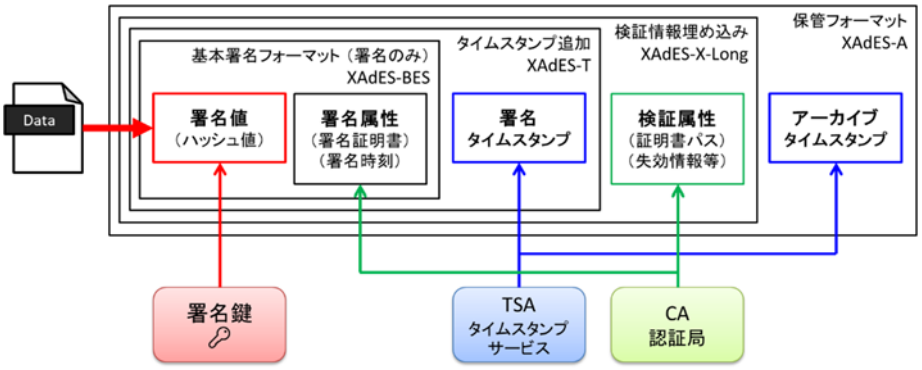
機能と全体の構成

ライブラリ名	ドキュメント（マニュアル）	内容
LE:XAdES:Lib	LeXAdES-manual.pdf （本書）	XML 署名の付与手順や検証の方法やオプション設定等の説明
LE:PKI:Lib（単体利用可能） ※ タイムスタンプ取得のみ等であれば LE:PKI:Lib だけで可能	LePKI-manual.pdf	署名方法や、証明書検証やタイムスタンプ利用の方法やオプション等の説明
LE:Client:Sign（単体利用不可） ※ 利用には LE:PAdES:Lib または LE:XAdES:Lib が必要	LeClientSign2-manual.pdf	ブラウザ（Edge/Chrome）への署名機能の JavaScript 組み込み方法等の説明

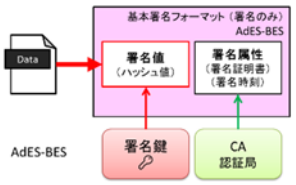
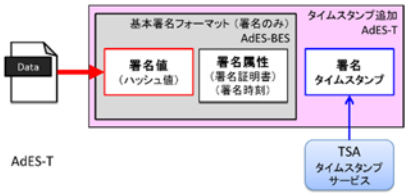
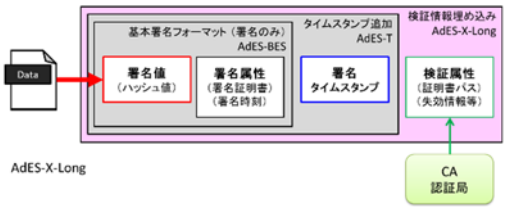
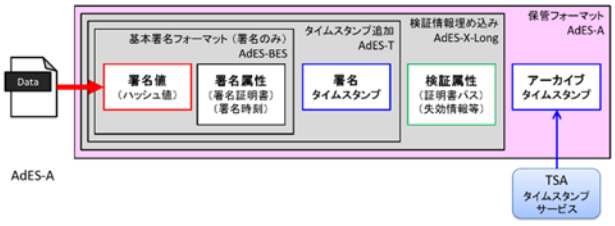
機能とドキュメント構成

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

長期署名フォーマットは、署名データ＋タイムスタンプトークン＋検証情報（証明書＋失効情報）で構成される。これらを順に追加して行くことで生成して行くことになる。



一般的な長期署名構造

手順	構造	説明
Step.1 XAdES-BES 署名付与 API: addReference() ＋ sign()	 ※ 署名者が保証された状態	任意の署名対象（複数可）を対象として署名証明書や署名時刻を付け署名データを生成する。
Step.2 XAdES-T 署名タイムスタンプ API: addEsT()	 ※ 署名者と署名時刻が保証された状態	署名時刻を保証するタイムスタンプを追加する。 署名値が対象なので署名証明書の保証はされない。
Step.3 XAdES-X-Long 検証情報埋め込み API: addEsXL()	 ※ 全証明書の検証情報が XAdES 内に埋め込まれている状態	署名証明書と TSA 証明書を検証し、認証パス証明書群と失効情報群を埋め込んで追加する。
Step.4 XAdES-A アーカイブ タイムスタンプ API: addEsA()	 ※ 外側のアーカイブタイムスタンプを除き全て保証した状態	全体を保護する為のアーカイブタイムスタンプを追加する。
Step.5 延長	---	Step.3 と 4 を繰り返す。

長期署名の作成手順

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

１．２．署名対象の指定

XAdES のベースとなっている XML 署名では大きく分けて Detached（外包）/Enveloping（内包）/Enveloped（埋め込み）/Manifest（間接指定）の 4 種類の署名対象指定方法がある。LE:XAdES:Lib では全てをサポートしている。また Enveloped を除いた Detached と Enveloping と Manifest は混在可能であり複数の署名対象を一度に指定することができる。

指定方法	概要
Detached	外部ファイルまたは Id を URI 指定することでそのまま署名対象とする。 外部 Detached：外部ファイルを示す URI を指定。 内部 Detached：同じ XML データの中の別要素を Id で指定。
Enveloping	外部ファイルを Base64 化等して Signatre 要素下の Object 要素として内包する。
Enveloped	XML データの中に署名の Signature 要素を埋め込む。 内部 Detached に近い指定が可能だが URI が空文字 "" になる点が異なる。
Manifest	Manifest 要素を参照して署名し、Manifest 内から更に URI で署名対象を指定する。 Enveloping Manifest：Manifest を Signatre 要素下の Object 要素として内包する。 Detached Manifest：Manifest を外部ファイルとして指定する。

署名対象の指定方式

LE:XadES:Lib V3 では LeReference クラスに参照をセットする API により各種参照方式に対応している。

方式	種類	LeReference API	概要
分離 1	外部 Detached	setDetachedFile()	ファイルパスで外部ファイルを指定
		setDetachedData()	データと URI で外部ファイルを指定
		setDetachedHash()	ハッシュ値と URI で外部ファイルを指定
分離 2	内部 Detached	setDetachedId()	同じ XML ファイル内の Id を指定 ※ 事前に LeXAdES3.setXml() が必要
内包	Enveloping	setEnvelopingFile()	ファイルパスで読み込みデータを指定
		setEnvelopingData()	データ指定で読み込みデータを指定
外包	Enveloped	setEnveloped()	同じ XML を Enveloping 指定 ※ 事前に LeXAdES3.setXml() が必要
間接	Manifest	setManifestFile()	ファイルパスで Manifest を指定 ※ enveloping 引数で Envelping 指定可能
		setManifestData()	データと URI で Manifest を指定 ※ enveloping 引数で Envelping 指定可能

LeReference による参照指定

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

◆ LeReference クラス対象セット API(C++)

```
// ------------------------------------------------------------------
// 分離 1 : 外部 Detached セット

// 外部 Detached 参照のファイルセット
INT setDetachedFile(
       const WCHAR* uri,
       const WCHAR* rootDir = NULL,
       const CHAR*  mimeType = NULL);

// 外部 Detached 参照のバイナリセット
INT setDetachedData(
       const BYTE*  refData,
       SIZE         dataLen,
       const WCHAR* uri,
       const CHAR*  mimeType = NULL);

// 外部 Detached 参照のハッシュセット
INT setDetachedHash(
       LPK_HASH     hashType,
       const BYTE*  hashData,
       SIZE         hashLen,
       const WCHAR* uri,
       const CHAR*  mimeType = NULL);
```

```
// ------------------------------------------------------------------
// 分離 2 : 内部 Detached セット
// ※ 事前に LeXAdES3.setXml() による XML セットが必要

// 内部 Detached 参照の Id(URI)セット
INT setDetachedId(
       const WCHAR* refId,
       LEX_TRANS_TYPE c14n = XR_TRANS_NONE,
       const CHAR*  mimeType = NULL);
```

```
// ------------------------------------------------------------------
// 内包 : Enveloping セット

// Enveloping 参照のファイルセット
INT setEnvelopingFile(
       const WCHAR* fpath,
       const CHAR*  mimeType = NULL);

// Enveloping 参照のバイナリセット
INT setEnvelopingData(
       const BYTE*  refData,
       SIZE         dataLen,
       const CHAR*  mimeType = NULL);
```

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

```
// ------------------------------------------------------------------
// 外包：Enveloped セット
// ※ 事前に LeXAdES3.setXml() による XML セットが必要

// 外包 Enveloped のセット
INT setEnveloped(
       LEX_TRANS_TYPE c14n = XR_TRANS_NONE,
       const CHAR* xpath = NULL);
```

```
// ------------------------------------------------------------------
// 間接：Manifest セット

// Manifest ファイル外部参照の追加（Manifest 階層化）
INT setManifestFile(
       const WCHAR* file,
       const WCHAR* rootDir = NULL,
       bool enveloping = false,
       LEX_TRANS_TYPE c14n = XR_TRANS_NONE);

// Manifest ファイル外部参照の追加（Manifest 階層化）
INT setManifestData(
       const BYTE* refData,
       SIZE dataLen,
       const WCHAR* uri,
       bool enveloping = false,
       LEX_TRANS_TYPE c14n = XR_TRANS_NONE);
```

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

1-A）外部 Detached 指定 - 分離 1:署名対象は外部ファイルを参照

署名ファイル名：[任意]	
<Signature> <SignedInfo> <Reference URI="Target.xml"> <DigestValue>…(Base64)…</DigestValue> </Reference> </SignedInfo> </Signature>	URI で外部ファイルを指定 署名対象のハッシュ値

対象ファイル名：Target.xml	
<MyData> <data>…</data> </MyData>	※ XML ファイル以外に何でも指定可能 例：PDF ファイル / TIFF ファイル等

1-B）内部 Detached 指定 - 分離 2:署名対象は同一ファイルから任意の Id 指定

署名ファイル名：[任意]	
<MyData> <data Id="signTarget">…</data> <Signature> <SignedInfo> <Reference URI="#signTarget"> <DigestValue>…(Base64)…</DigestValue> </Reference> </SignedInfo> </Signature> </MyData>	URI に**空文字列**を指定すると自分自身の先頭から<Signature>要素を省いた範囲が署名対象となる

2）Enveloping 指定 - 内包:署名対象は同一ファイル内 Object 要素の Id を指定

署名ファイル名：[任意]	
<Signature> <SignedInfo> <Reference URI="#MyObjId"> <DigestValue>…(Base64)…</DigestValue> </Reference> </SignedInfo> <Object Id="MyObjId"> <MyData> <data>…</data> </MyData> </Object> </Signature>	URI で同一ファイル内 Id を指定 署名対象のハッシュ値

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

3） Enveloped 指定 - 外包:署名対象は同一ファイルの先頭から署名を除いた範囲

署名ファイル名：[任意]

```
<MyData>
  <data>…</data>
  <Signature>
    <SignedInfo>
      <Reference URI="">
        <DigestValue>…(Base64)…</DigestValue>
      </Reference>
    </SignedInfo>
  </Signature>
</MyData>
```

＝

```
<MyData>
  <data>…</data>
</MyData>
```

URI に**空文字列**を指定すると自分自身の先頭から<Signature>要素を省いた範囲が署名対象となる

4-A） Enveloping Manifest 指定 - 間接 1:署名対象は Manifest 要素から参照

ファイル名：[任意]

```
<Signature>
  <SignedInfo>
    <Reference URI="#MyManifestId"
     Type="http://www.w3.org/2000/09/xmldsig#Manifest">
      <DigestValue>…(Base64)…</DigestValue>
    </Reference>
  </SignedInfo>
  <Object>
    <Manifest Id="MyManifestId">
      <Reference URI="Target.xml">
        <DigestValue>…(Base64)…</DigestValue>
      </Reference>
    </Manifest>
  </Object>
</Signature>
```

URI で同一ファイル内 Id を指定
Manifest 要素のハッシュ値

URI で外部ファイルを指定
署名対象のハッシュ値

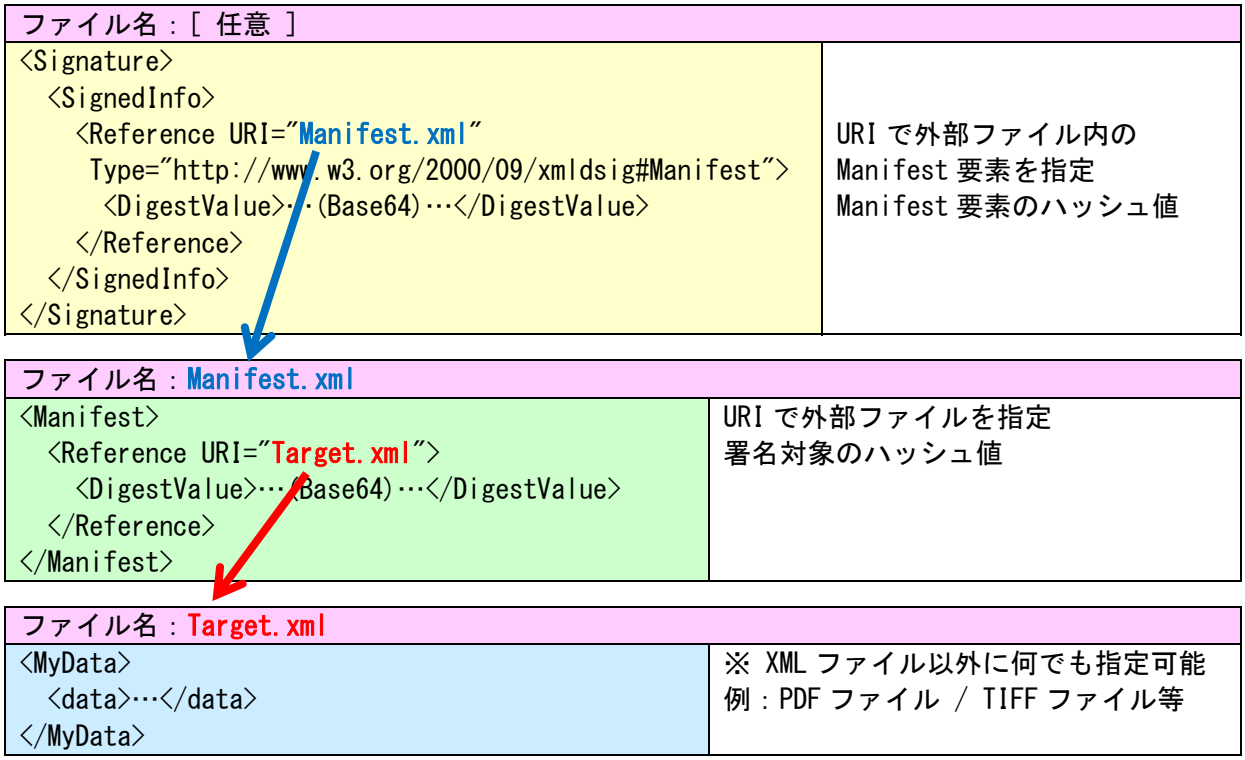
ファイル名：Target.xml

```
<MyData>
  <data>…</data>
</MyData>
```

※ XML ファイル以外に何でも指定可能
例：PDF ファイル / TIFF ファイル等

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

4-B）Detached Manifest 指定 - 間接 2:署名対象は Manifest 要素から参照



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

１．３．API 環境

LE:XAdES:Lib V3 は C++で開発されており、インターフェイスとして C++ / Java / .NET / コマンドの 4 種類が提供される。Java 環境に関しては JNI を使っている為にピュア Java では無い、また.NET 環境に関しても C++/CLI を使っている為にピュアなマネージコードでは無いので注意が必要である。

API 環境	C++	Java	.NET	コマンド
補足	.DLL/.so 呼び出し	JNI 経由により C++呼び出し	ラッパ経由により C++呼び出し	コマンド引数にてオプション指定
利用例	アプリ組み込み等	Web サービスへの組み込み等	ASP/.NET 等にて C#・VB 等で利用	バッチファイルからの利用等
動作環境	VisualStudio/stdc ランタイムが必要	Java8(1.8)以上	VisualStudio ランタイムが必要 **※ Windows のみ**	VisualStudio/stdc ランタイムが必要

C++アプリ	Java アプリ	.NET アプリ	バッチ(シェル)ファイル
C++ヘッダファイル	JNI インターフェイス	ラッパクラス(C++/CLI)	Lx3Cmd コマンド
C++モジュール（.DLL/.so ファイル）			

１）C++の API

ビルド時に include ファイルとリンクライブラリが必要です。
Include/LeXAdES3/LeXAdES3.h と include/LePKI/LePKI.h を参照ください。

２）Java の API

JNI を使って内部的には C++の API を呼び出しています。
クラス構成はほぼ C++ のクラス構成と同じですが、戻り値が異なります。

３）.NET の API

C++/CLI のラッパクラスを使って内部的には C++の API を呼び出しています。
クラス構成はほぼ Java のクラス構成と同じです。

４）コマンドによる API

簡易に利用ができるように Lx3Cmd が提供されます。
コマンドの引数によりオプション指定することで各種機能が利用可能です。

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

１．４．評価版

評価版には以下の制限項目がある。

No	制限項目
1	評価版で署名時または Manifest 生成時に追加可能（addReference）な**参照は最大 4 つまで**となります。4 つ以上の参照を追加した場合には、エラーコード XDA_ERR_EVALUATION（-5992）が返ります。なお自動追加される XAdES オブジェクトの参照はカウントしません。
2	評価版で付与する電子署名の署名 Id（ds:Signature 要素の Id 属性）には**常に「LeEval-」が挿入**されます。署名 Id を指定しても必ず最初に挿入されます。
3	評価版にソースコードは付属しません。

評価版の確認は Lx3Cmd を実行することで確認可能です。

・Windows 版の実行方法

> bin_win/Release/Lx3Cmd.exe

・Linux 版の実行方法

$ bin_linux/Lx3Cmd.sh

○ 以下“のように "[EVAL]" が表示された場合は評価版です。

```
Lx3Cmd : LeXAdES(V3.0X.RX)/LePKI(V1.0X.RX) [EVAL] Langedge XAdES V3 Command.
      :
```

○ 以下のように表示された場合は製品版です。

```
Lx3Cmd : LeXAdES(V3.0X.RX)/LePKI(V1.0X.RX) Langedge XAdES V3 Command.
      :
```

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

１．５．動作環境

LE:XAdES:Lib V3 の動作環境としては Windows と Linux が対象となる。詳細な環境としては以下となる。開発に使っている Linux ディストリビューションは CentOS 系であるが、Debian と RedHat でも動作確認されている。

Windows 環境	・Windows 10 以降／Windows Server 2016 以降 ・32bit と 64bit 用を同梱 ・Visual Studio C++ 2017 と 2022 でビルド（C++ランタイムが必要） → リビルドすることで VS2019 / VS2026 に対応可能 ・OpenSSL、OpenLDAP、libxml2、zlib（全て同梱） ・PKCS#12 形式証明書と Windows 証明書ストアと PKCS#11 形式 IC カード
Linux 環境	・Linux x86（64bit） ・GCC 4.1/4.3 でビルド（Makefile 利用） ・libc-2.5.so 以上、libstdc++.6.0.so 以上 ・OpenSSL、OpenLDAP、libxml2、zlib（全て同梱）、(iconv)モジュール ・PKCS#12 形式証明書（Java 証明書ストア利用不可）
クライアント署名 (オプション利用)	・Windows 10 以降 ・ブラウザ：Edge／Chrome ・Windows 証明書ストア（IC カードも利用可能）と PKCS#11 形式 IC カード

動作環境

リリース種別	種別	OS 環境	動作環境
LE:XAdES:Lib V3 Windows 版	LE:XAdES:Lib （製品版）	Windows	32bit/64bit 同梱
LE:XAdES:Lib V3 Linux 64bit 版		Linux	64bit ※
LE:XAdES:Lib V3 Windows 評価版	LE:XAdES:Lib Eval （評価版）	Windows	32bit/64bit 同梱
LE:XAdES:Lib V3 Linux 64bit 評価版		Linux	64bit ※
※ 各 Linux 32bit 版のご提供は未対応です。どうしても必要な場合はご連絡ください。			

リリース種別

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

１．６．動作に必要なファイル

LE:XAdES:Lib V3 の動作に必要なファイルは大きく分けて LeXAdES3 と LePKI に分かれる。なお Java より利用する場合にはクラスパッケージと JNI 用のモジュールが必要となる。コマンドを使うにはコマンド実行ファイルが必要となる。

Windows 環境 [bin_win]の下	Linux 環境 [lib_linux]の下	概要
LeXAdES3.dll	libLeXAdES3.so	◎ 必須モジュール XAdES 操作他を行うメインモジュール、LePKI に依存
LeXAdES3jni.dll	libLeXAdES3jni.so	○ Java 環境利用時のみ必要なモジュール Java 用の LeXAdES3 クラスパッケージとモジュール
lexades-3.0.X.jar [java/lib]の下		
LeXAdES3dnet.dll	（未サポート）	○ .NET 環境利用時のみ必要なモジュール
LePKI.dll	libLePKI.so	◎ 必須モジュール PKI 操作他を行うモジュール
LePKIjni.dll	libLePKIjni.so	○ Java 環境利用時のみ必要なモジュール Java 用の LePKI クラスパッケージとモジュール
lepki-1.0.X.jar [java/lib]の下		
LePKIdnet.dll	（未サポート）	○ .NET 環境利用時のみ必要なモジュール
Lx3Cmd.exe [bin_win]の下	Lx3Cmd Lx3Cmd.sh [bin_linux]の下	○ XAdES コマンド環境利用時のみ必要なモジュール Linux 版では Lx3Cmd.sh の利用を推奨
LpkCmd.exe [bin_win]の下	LpkCmd LpkCmd.sh [bin_linux]の下	○ PKI コマンド環境利用時のみ必要なモジュール Linux 版では LpkCmd.sh の利用を推奨 ※ PKI 操作のみ必要な場合に利用

動作に必要な本体モジュール

Windows 環境	C++ランタイムコンポーネント （再頒布可能パッケージ）	全てスタティックリンクで利用 ※ 依存する DLL は全てシステム用
Linux 環境	特に無し	全てスタティックリンクで利用

動作に必要な外部モジュール

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

１．７．Windows 環境のインストールとソースビルド

１） Windows 環境のインストール

1-0） LeXAdES-3.XX.RX.zip を Unzip 展開する。

1-1） 展開したフォルダ（LeXAdES-3.XX.RX）の中にある bin_win¥Release64（64bits）または bin_win¥Release（32bits）ディレクトリを環境変数 Path に加えるか、bin_win¥Release64 または bin_win¥Release ディレクトリの下に入っている DLL ファイルと Lx3Cmd.exe を、環境変数 Path に含まれているディレクトリ下にコピーする。以下が実行に必要なファイル。

LE:XAdES:Lib V3 ファイル：
LeXAdES3.dll、LeXAdES3jni.dll、LeXAdES3dnet.dl、
LePKI.dll、LePKIjni.dll、LePKIdnet.dll、Lx3Cmd.exe

※ 以下の LE:XAdES:Lib V2 ファイルとは共存可能です。
LeXAdESdnet.dl、LxaCmd.exe、libeay32L.dll（OpenSSL）

1-2） Microsoft Visual C++ の再頒布可能パッケージが必要です。必要であれば以下アドレスから取得する。

最新のサポートされる Visual C++ のダウンロード（マイクロソフト・サポート）
https://support.microsoft.com/ja-jp/help/2977003/the-latest-supported-visual-c-downloads

1-3） DOS 窓を開き、Lx3Cmd.exe が実行できることを確認する。

```
> Lx3Cmd.exe
Lx3Cmd : LeXAdES(V3.0X.RX)/LePKI(V1.0X.RX) Langedge XAdES V3 Command.
Copyright (c) 2012-202X LangEdge,Inc. all rights reserved.
   :
>
```

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

2） Windows 環境のソースビルド

2-1） LE:XAdES:Lib V3 の src フォルダ中にある LeXAdES3-2022.sln を VisualStudio 2022 で開く。開いたらスタートアッププロジェクトとして "Lx3Cmd" を指定する。

※：VisualStudio 2022 以外でもビルド可能。

開発環境バージョン	ビルドプロジェクト （ソリューションファイル）
Visual Studio 2015	src¥LeXAdES3-2015.sln
Visual Studio 2017	src¥LeXAdES3-2017.sln **※ 製品リリースのビルドに利用**
Visual Studio 2019	src¥LeXAdES3-2019.sln
Visual Studio 2022	src¥LeXAdES3-2022.sln **※ 製品リリースのビルドに利用**
Visual Studio 2026	src¥LeXAdES3-2026.sln

※ Ver3.02 より Visuasl Studio 2015 は非サポートとなった。

2-2） ソリューション構成［Release］を選択する。
プラットフォームは 32bit の場合［Win32］を、64bit の場合［x64］を選択する。
全ての構成をセット後に［ソリューションのリビルド］を実行する。

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

１．８．Linux 環境のインストールとソースビルド

１）Linux 環境のインストール

1-0） LeXAdES-3.XX.RX.tar.gz を任意のディレクトリで tar 展開する。

1-1） 展開したフォルダ（LeXAdES-3.XX.RX）下にある bin_linux/Lx3Cmd.sh を引数無しで実行する。エラーが表示されなければソースビルドは不要でこのまま利用できる。

例：$ LeXAdES-3.XX.RX/bin_linux/Lx3Cmd.sh
Lx3Cmd : LeXAdES(V3.XX.RX)/LePKI(V1.XX.RX) LangEdge XAdES V3 Command.
Copyright (c) 2012-202X LangEdge,Inc. all rights reserved.
（以下略）

1-2） Lx3Cmd/C++/Java 各 API の動作基本テストを行い、エラーが無いか確認する。

※ Java は Java のコンパイルと実行環境が必要。
※ C++は gcc/g++のコンパイル環境が必要。
※ タイムスタンプや検証情報の取得の為にインターネット接続が必要です。

例：$ make test

※ コマンド（Lx3Cmd.sh）だけを使うだけなら以上でインストール終了となる。
C+++/Java の API を利用する場合には次ページの 1-4A）または 1-4B）の手順を実行する。

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

○ Linux 環境のインストール（続き：C++/Java の API を使う場合）

1-4A) **※ LD_LIBRARY_PATH 環境変数をセットして利用する場合：**

展開したディレクトリ（LeXAdES-3.XX.RX）下にある lib_linux ディレクトリを実行時に LD_LIBRARY_PATH 環境変数に追加する。

```
-----(.bashrc 等シェルの設定例：$HOME 下にインストール時)---------
LD_LIBRARY_PATH=$HOME/LeXAdES-3.XX.RX/lib_linux:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
-----------
```

1-4B) **※ Linux 標準のライブラリディレクトリ（例：/usr/lib 等）にインストールする場合：**

展開したディレクトリ（LeXAdES-3.XX.RX）下にある lib_linux ディレクトリの中にあるファイルをコピーする。/usr/lib の下であれば make install や make install32 でも良い。このインストール方法ではルート権限が必要です。

例 1(64bit 環境)：

```
# cd LeXAdES-3.XX.RX; make install
      or
# cp LeXAdES-3.XX.RX/lib_linux/*.so /usr/lib64
```

例 2(32bit 環境)：

```
# cd LeXAdES-3.XX.RX; make install32
      or
# cp LeXAdES-3.XX.RX/lib_linux/*.so /usr/lib
```

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

2） Linux 環境のソースビルド

2-1） ソースビルドには Java の API の為に JNI を使うので.bashrc 等で環境変数 JAVA_HOME のセットが必要。以下は Java8 の場合の設定例。

```
JAVA_HOME=/usr/lib/jvm/java-8-sun
export JAVA_HOME
```

2-2） 展開したディレクトリ（LeXAdES-3.XX.RX）下で make コマンド（クリーンとビルド）を実行する。

例 1 (64bit 環境) :

```
$ make clean; make all
```

例 2 (32bit 環境) :

```
$ make clean; make all32
```

※：iconv に関しては最近の Linux では glibc にて標準で組み込まれている。
標準でサポートされていれば /usr/include/iconv.h が提供されているはず。
iconv.h が開けないエラーになる場合には、LE:XAdES:Lib の local/src/iconv の下にある iconv.h を /usr/include 等にコピーして試す。

※：undefined r‘ference to 'l’biconv_open' 等のエラーになる場合にのみ libiconv のインストールを試すことを推奨。

※：その他ビルド時にエラーが出る場合は弊社まで連絡ください。

2-3) CentOS/RHE 等で SELinux をオンにしている場合には SELinux 設定を行う必要がある場合がある。2-2)までの手順でうまく動作しない場合に試してください。実行にはルート権限が必要です。

例：`$ make selinux`

2-4） 以後手順は「Linux 環境のバイナリインストール」の 1-1）以降と同じ手順で動作確認とインストールを行う。

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

3） Linux 環境の make 利用方法

LE:XAdES:Lib V3 のルートディレクトリ直下にある Makefile を利用する方法を示す。

コマンド	説明
$ make clean	一括クリーン（中間オブジェクトや生成物の削除）
$ make	一括ビルド(64bit 版) ※ $ make all や $make xades でも同じ
$ make all32	一括ビルド(32bit 版)
$ make test	全テスト(sample)実行 ※ $make test.xades でも同じ
# make install	インストール(/usr/lib64 へインストール)実行(64bit 版) ※ルート権限が必要
# make install32	インストール(/usr/lib へインストール)実行(32bit 版) ※ルート権限が必要

■ Linux 環境の注意事項

Windows 版とのソース互換を保つためにソースファイルの日本語コードは SHIFT-JIS を利用している。

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

１．９．内部プロジェクト構成

内部プロジェクトは XAdES 操作の LeXAdES3 モジュールと PKI 操作の LePKI モジュールに分かれる。全プロジェクトは保守目的で利用可能ライセンスとして全ソースが付属する。

モジュール	プロジェクト	概要	補足
LeXAdES3.dll (libLeXAdES3.so)	LeXAdES3	XAdES 署名操作を行う ・署名対象の指定 ・長期署名化	---
	LePKI	PKI 操作を行う ・証明書の利用	証明書クラス LpkCert の利用が必要な為に依存している。
	LeCommon	共通低レベル機能	ユーティリティと通信
LePKI.dll (libLePKI.so)	LePKI	PKI 操作を行う ・署名データ生成と検証 ・証明書の操作と検証	ラング・エッジの他の長期署名製品（例：LE:XAdES:Lib）と共通利用される。
	LeBerXml	ASN.1/BER 操作機能	ASN.1/BER 形式の操作用。LpkCades 等で利用。
	LeCommon	共通低レベル機能	ユーティリティと通信
Lx3Cmd.exe (Lx3Cmd)	Lx3Cmd	XAdES 操作のコマンド	XAdES/PKI 操作
	[LeXAdES3]	XAdES 署名操作を行う	---
	[LePKI]	PKI 操作を行う	---
LpkCmd.exe (Lpkmd)	LpkCmd	PKI 操作のコマンド ・署名や証明書の検証	PKI 操作のみ
	[LePKI]	PKI 操作を行う	---
LeXAdES3jni.dll (LeXAdES3jni.so)	LeXAdES3jni	LeXAdES V3 Java API 用	Java JNI プロジェクト
	[LeXAdES3]	XAdES 署名操作を行う	---
LePKIjni.dll (LePKIjni.so)	LePKIjni	LePKI Java API 用	Java JNI プロジェクト
	[LePKI]	PKI 操作を行う	---
LeXAdES3dnet.dll (Windows のみ)	LeXAdES3dnet	LeXAdES V3 .NET API 用	C++/CLI (C#)プロジェクト
	[LeXAdES3]	XAdES 署名操作を行う	---
LePKIdnet.dll (Windows のみ)	LePKIdnet	LePKI .NET API 用	C++/CLI (C#)プロジェクト
	[LePKI]	PKI 操作を行う	---

LE:XAdES:Lib V3 の内部プロジェクト構成（src フォルダの下にある）

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

１．１０．利用外部ライブラリ

外部ライブラリは Linux にも対応したマルチプラットホームのプロジェクトを利用している。ただしクライアント署名や Windows 環境で必要な機能は、Windows 標準のモジュールを利用している。外部ライブラリはビルドしたソースが local/src ディレクトリの下に格納されている。特に Windows 環境用はビルドの為に少し変更している部分もある。詳しくは各ディレクトリ下を参照。

プロジェクト	Ver	利用範囲
OpenSSL	3.1.1	・RSA や SHA-1/2、証明書、CRL 等の基本ライブラリとして利用 ・PKCS#7 や RFC3161 タイムスタンプ等のライブラリとして利用 ・タイムスタンプ属性証明書サポートの為に**ソースを一部修正** ● Windows は libcrypto_static.lib / libssl_static.lib をスタティックリンク ○ Linux は libcrypto.a / libssl.a をスタティックリンク
OpenLDAP	2.4.32	・LDAP 通信に利用（CRL や証明書の取得） ◎ Windows / Linux 共に libldap.a / liblber.a をスタティックリンク
libxml2	2.7.8 (2.9.3)	・XML 作成と解析に利用（設定ファイルやクライアント署名の通信用） ◎ Windows は libxml2.lib、Linux は libxml2.a をスタティックリンク
zlib	1.2.5	・libxml2 から利用（libxml2 の為に V1.2.5 以上が必須） ◎ Windows は libz.lib、Linux は libz.a をスタティックリンク
iconv	---	・UTF-8/SJIS 等のコード変換（通常 GNU の C ライブラリに付属） × Windows では利用しない △ Linux では通常 glibc に含まれるので不要（必要なら用意する）

オープンソースのライブラリ（local フォルダの下にある）

モジュール	利用範囲
CryptoAPI	Windows 証明書ストアの利用と LDAP 通信（予備）
WinHTTP	HTTP/HTTPS 通信（Windows 環境デフォルト利用）
WinSock	HTTP/HTTPS（OpenSSL 利用）通信（Windows 環境予備、オプション指定可能）
WinInet	HTTP/HTTPS 通信（Windows 環境過去互換用、オプション指定可能）
WinLdap	LDAP 通信（Windows 環境メイン）

Windows 環境の標準ライブラリ

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

※ 外部利用ライブラリのコピーライト表示

ライブラリ	Windows	Linux	ライセンス（下段はコピーライト表示）
OpenSSL	○	○	Apache License 2.0（**コピーライト表示義務あり**）
			Copyright 1998-2022 The OpenSSL Project Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.
OpenLDAP	○	○	OpenLDAP Public License – BSD 系（**コピーライト表示義務あり**）
			Copyright 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved.
libxml2	○	○	MIT License（**コピーライト表示義務あり**）
			Copyright (C) 1998-2003 Daniel Veillard. All Rights Reserved.
zlib	○	○	zlib License（コピーライト表示義務無し）
			Copyright (C) 1995-2012 Jean-loup Gailly and Mark Adler.
iconv	×	○	LGPL License（**コピーライト表示義務あり**）
			Copyright (C) 1999-2003 Free Software Foundation, Inc.

ライブラリ毎のライセンス種類とコピーライト表示

各ライブラリのライセンスファイルが license フォルダ下に格納されている。LE:XAdES:Lib V3 を組み込んだ製品をリリースする際には license フォルダ下のファイルを含める事を推奨する。

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

コピーライト表示が必要なライブラリについてはアバウト画面等でコピーライト表示する事を推奨する。コピーライト表示例を以下に示す。

```
[OpenSSL License]
Copyright 2002-2020 The OpenSSL Project

Licensed under the Apache License, Version 2.0 (the “License”);
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is
distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and limitations under the License.

 [OpenLDAP License]
This product includes softwares developed by:
 Copyright 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA.  All Rights Reserved.
 http://www.openldap.org/

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

[zlib License]
This product incluI softwares developed by:
 Copyright (C) 1995-2012 Jean-loup Gailly and Mark Adler.
 http://www.zlib.net/
```

3． Windows 用コピーライト表示 iconv のコピーライトが無い)

※ zlib のコピーライト表示は省略可能

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

１．１１．フォルダ構成

フォルダ/ファイル	説明	補足
[bin_linux]	Linux 環境用の実行モジュール用フォルダ。	Lx3Cmd 利用時必要
[bin_win]	Windows 環境用の実行モジュール用フォルダ。 Debug / Release / Debug64 / Release64 に分かれる。	実行時必要 (Windows/Java)
[lib_linux]	Linux 環境用の実行ライブラリ用フォルダ。 ダイナミックリンクファイル(.so)が格納されている。	実行時必要 (Linux/Java)
[lib_win]	Windows 環境用のリンクライブラリ用フォルダ。 Debug / Release / Debug64 / Release64 に分かれる。	C++ API 利用時必要 (Windows)
[java]	Java API 用のソースとライブラリ用フォルダ。 [lib] 実行とコンパイルに必要な jar ファイル格納。 [LeXAdES3][LePKI] Java 用ソース/プロジェクト。	実行時必要(Java) Java API 利用時必要
[include]	C++利用時に必要となるインクルードフォルダ。 LeCommon / LeXAdES3 / LePKI に分かれている。	C++ API 利用時必要
[clientV2]	クライアント署名 V2 用プロジェクトフォルダ。 [bin_activex] V2 ActiveX モジュール格納 (IE11 用)。 [bin_cmd] V2 署名コマンド格納 (Edge/Chrome 用)。 [doc] クライアント署名 V2 マニュアル [ServerTest] ローカル試験用ツールのプロジェクト。 [ServerTest/bin] ローカル試験時に PATH を通す。	クライアント署名 V2 利用時必要
[doc]	C++/Java/.NET の API ドキュメント他のフォルダ。 [LeXAdES-manual.pdf] 製品マニュアル [LeXAdES3/cpp/] C++ API リファレンス [LeXAdES3/java/] Java API リファレンス [LeXAdES3/dotnet/] .NET API リファレンス [LeXAdES3/cmd/Lx3Cmd-Help.txt] Lx3Cmd ヘルプ	リファレンス用
readme-1st.txt	ラング・エッジ製品全体に関する説明ファイル。	お読みください
readme-LeXAdES3.txt	LE:XAdES:Lib V3 に関する説明ファイル。	お読みください
[sample]	[LeXAdES3/cmd] Lx3Cmd(コマンド)利用サンプル [LeXAdES3/cpp] C++利用サンプル [LeXAdES3/java] java 利用サンプル [LeXAdES3/cs] C#(.NET)利用サンプル [LePKI-server] 検証プロキシサーバ Tomcat 実装例	利用方法学習用
Makefile	Linux 環境用のメイン Makefile。	保守用
[src]	内部ライブラリ（LeXAdES3/LePKI 等）用のフォルダ [src/LeXAdES3-2019.sln] VS2019 用ソリューション	保守用
[dotnet]	.NET のラッパクラスのソース用フォルダ	保守用
[local]	外部ライブラリ（OpenSSL 等）用のフォルダ。	保守用
EULA-LeXAdES3.txt	ソフトウェア製品使用許諾契約書。	著作権
[license]	外部ライブラリ用のライセンスフォルダ。	著作権

ルート下にあるフォルダとファイル

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

１．１２．文字コードと LeString クラス

LE:XAdES:Lib V3 では Windows 版と Linux 版のソースコードを共通化する為に内部文字コードとしてシフト JIS を利用している。また動作環境による差異を避ける為に独自の文字列クラス LeString を提供している。LeString クラスではユニコード（STL の std::wstring）や UTF-8 への変換 API が提供されている。

C++環境ではシフト JIS で文字列が返されるので、必要に応じて LeString クラスにより文字コードを変換して利用が可能。

LeString のメソッド	説明
const CHAR* c_str(); std::string getStr();	シフト JIS 形式で文字列を返す。
std::wstring getWStr();	ユニコード形式で文字列を返す。 ユニコードは Windows 環境(UCS2)と Linux 環境(UCS4)では異なる。
BINARY getUTF8();	UTF-8 形式で文字列を返す。

独自文字列クラス LeString の文字列取得/変換メソッド

Java 環境では文字列は java.lang.String クラスを利用しているので、内部コードはユニコードとなる。String は Java 標準の文字列クラスなので特に注意する必要はない。

Lx3Cmd は、Windows 版の場合には標準でシフト JIS 形式にて出力し、Linux 版の場合には標準で英語（ASCII 形式）にて出力する。ただし Lx3Cmd ではオプション指定により英語（ASCII 形式）とシフト JIS 形式と UTF-8 形式の選択が可能である。

Lx3Cmd 引数	説明
（指定無し）	Windows 環境では日本語メッセージをシフト JIS 形式で表示する。 Linux 環境では英語メッセージを ASCII 形式で表示する。
-eng	英語メッセージを ASCII 形式で表示する。（Linux 環境デフォルト設定）
-sjis	日本語メッセージをシフト JIS 形式で表示する。（Windows 環境デフォルト設定）
-utf8	日本語メッセージを UTF-8 形式で表示する。

Lx3Cmd の出力メッセージのオプション指定

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

２．ＡＰＩ利用方法

２．１．API リファレンス

以下に LE:XAdES:Lib V3 のクラス構成を示す。

LeXAdES3 クラス	XAdES 長期署名を行うメインとなるクラス
LeReference クラス	署名参照情報クラス（署名時に利用）
LeManifest クラス	間接指定情報クラス（署名時に利用）
XdaVerifyXml クラス	署名検証結果クラス（検証時に利用）
XdaClientXml クラス	クライアント署名クラス（LE:Client:Sign 連携時に利用）

LeXAdES3 クラス構成

LePKI クラス	PKI 操作メインクラス（認証パスの構築や検証等）
LpkCert クラス	X.509 証明書クラス
LpkCrl クラス	証明書失効リスト（CRL）クラス
LpkOcsp クラス	オンライン証明書状態プロトコル（OCSP）クラス
LpkPkcs7 クラス	PKCS#7 署名クラス
LpkCades クラス	CAdES 署名クラス
LpkTimestampToken クラス	タイムスタンプトークン クラス
LpkTimestamp クラス	タイムスタンプ情報ベースクラス
LpkTs3161 クラス	RFC3161 タイムスタンプ情報クラス
LpkTsAmano クラス	アマノタイムスタンプ情報クラス
LpkUtil クラス	PKI 補助クラス（CRL/OCSP/証明書のネット取得等）
LpkCrypto クラス	PKI 暗号クラス（ハッシュ計算や暗号化）
LeString クラス	独自文字列クラス

LePKI クラス構成

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

C++と Java の API リファレンスは、Doxygen により HTML 形式で自動生成されて提供される。
コマンドは -help 引数により利用方法の説明が表示される。

1）C++の API リファレンス doc/LeXAdES3/cpp/index.html をブラウザで参照。
2）Java の API リファレンス doc/LeXAdES3/java/index.html をブラウザで参照。
3）.NET の API リファレンス doc/LeXAdES3/dotnet/index.html をブラウザで参照。
4）コマンド（Lx3Cmd）の引数リファレンス doc/LeXAdES3/cmd/Lx3Cmd-Help.txt を参照。

```
＞ Lx3Cmd

Lx3Cmd : LeXAdES(V3.0X.RX)/LePKI(V1.0X.RX) LangEdge XAdES V3 Command.
         Copyright (c) 2006-202X LangEdge,Inc. all rights reserved.

  ＞ Lx3Cmd.exe -help          : ヘルプ表示

  ＞ Lx3Cmd.exe -help -sign    : 署名付与(XAdES-BES[T])ヘルプ表示
  ＞ Lx3Cmd.exe -help -time    : タイムスタンプ付与(XAdES-T/XAdES-A)ヘルプ表示
  ＞ Lx3Cmd.exe -help -verify  : 検証実行と検証情報の埋め込み(XAdES-XL)ヘルプ表示

  ＞ Lx3Cmd.exe -help -schema  : XML スキーマチェックヘルプ表示
  ＞ Lx3Cmd.exe -help -info    : XAdES 情報操作ヘルプ表示
  ＞ Lx3Cmd.exe -help -refs    : Manifest ファイル生成ヘルプ表示

  ＞ Lx3Cmd.exe -help -server  : 仮署名付与/署名データ埋込(サーバ機能) ヘルプ表示
  ＞ Lx3Cmd.exe -help -client  : 署名データ生成(クライアント機能試験) ヘルプ表示

  ＞ Lx3Cmd.exe -help -all     : 全ヘルプ一括表示

  options: ヘルプオプション
    -eng / -sjis / -utf8 : メッセージを英語(eng)又は指定文字コードで表示

  xades-levelup-step)
    1> Lx3Cmd.exe -sign -cert ... -out xades-bes.xml
    2> Lx3Cmd.exe -time -ts ... -in xades-bes.xml -out xades-t.xml
    3> Lx3Cmd.exe -verify -embed ... -in xades-t.xml -out xades-xl.xml
    4> Lx3Cmd.exe -time -ts ... -in xades-xl.xml -out xades-a.xml
    5> Lx3Cmd.exe -verify -embed ... -in xades-a.xml -out xades-a-xl.xml-verify -embed ...¥
        -in xades-a.xml -out xades-a-xl.xml
```

Lx3Cmd の usage

```
Lx3Cmd : LeXAdES(V3.02.R1)/LePKI(V1.10.R1) LangEdge XAdES V3 Command.
         Copyright (c) 2006-2024 LangEdge,Inc. all rights reserved.

  ＞ Lx3Cmd.exe -command [-options]
    -command : メインコマンド(必須)
    -options : オプション指定

  command: 主操作を指定
    -sign    : 署名生成 (XAdES-BES)
```

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

```
  -time   : タイムスタンプ付与 (XAdES-T/XAdES-A)
  -verify : XAdES/Manifest 検証実行と XAdES 検証情報の埋め込み(XAdES-XL)
  -schema : XML スキーマチェック
  -info   : XAdES 情報操作
  -refs   : Manifest ファイル生成
  -server : 仮署名付与/署名データ埋込(サーバ機能)
  -client : 署名データ生成(クライアント機能試験)
  -help   : ヘルプ表示(sign/time/verify/info/refs/server/client と組み合わせ可)


> Lx3Cmd.exe -sign [-options] -det/eping/... -out signed.xml [-in base.xml]

options: 署名対象の指定オプション

 // 参照(ファイルまたは Id を参照する形式): Detached options.
  -det filepath [opts] : 外部 Detached(ファイル指定)形式の署名対象追加
  -det #Id [opts]      : 内部 Detached(Id 指定)形式の署名対象追加
  -detdir dirpath      : ディレクトリ下全ファイルの Detached 署名対象追加
  -dlist dlistfile     : Detached 形式の署名対象をリストファイルから追加
 // 内包(署名の中に対象を埋め込む形式): Enveloping options.
  -eping filepath [opts] : Enveloping 形式の署名対象追加
  -elist elistfile       : Enveloping 形式の署名対象をリストファイルから追加
     ※ 拡張子が.xml か MimeType が xml の場合以外ではバイナリとして追加
 // 外包(対象の中に署名を埋め込む形式): Enveloped options.
  -eped [xpath] : Enveloped 形式の署名対象追加、1 つのみ指定可
     ※ [xpath] に XPath オプションを指定可能
 // 間接(対象群を含む Manifest を参照する形式): Manifest options.
  -dmani mfilepath  : Manifest ファイルの Detached(ファイル指定)追加
  -dmlist mlistfile : Manifest ファイルをリストファイルから Detached 追加
  -emani mfilepath  : Manifest を読み込み Enveloping(埋め込み)追加
  -emlist mlistfile : Manifest ファイルをリストファイルから Enveloping 追加
  // 共通オプション: Common options.
  -dir dirpath : 外部ファイル参照時のホームディレクトリ指定(-detdir 不可)
 // 補足: Supplements.
  ※ dlist/elist/mlist のリストファイルは 1 行 1 対象ファイルパスで記述
  ※ det #Id/eped は -in オプションの同時指定が必須
  ※ [opts] には以下の引数が指定可能(-det #Id は Description のみ)
     mime:<MimeType>     : Mime-Type を指定する
       例) "mime:text/plain"
     encd:<Encoding>     : Encoding を指定する
       例) "encd:http://www.ietf.org/rfc/rfc2279.txt"
     desc:<Description> : Description(説明)を指定する
       例) "desc:This is test."

 // 例) 複数ファイル指定の Detached
  > Lx3Cmd -sign -det test.txt -det test2.xml ... -out sign.xml
 // 例) Id 指定の Detached
  > Lx3Cmd -sign -det #data1 ... -in base.xml -out sign.xml
 // 例) Enveloping 指定(MimeType 付)
  > Lx3Cmd -sign -eping test.txt mime:text/plain ... -out sign.xml
 // 例) Enveloped 指定
  > Lx3Cmd -sign -eped ... -in base.xml -out sign.xml
 // 例) 外部 Manifest 指定
```

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

```
   > Lx3Cmd -sign -dmani manifest.xml ... -out sign.xml
  // 例) 内部(Enveloping)Manifest 指定
   > Lx3Cmd -sign -emani manifest.xml ... -out sign.xml
  // 例) 複数指定
   > Lx3Cmd -sign -eped -det test.txt -det #data1 -eping test2.xml ¥
            ... -in base.xml -out sign.xml

 options: 署名(XAdES-BES)の生成オプション
   -id name : Sinature 要素の Id 属性名の指定
   -xpath xpath : -in 指定時の署名挿入位置の XPath 指定
      ※ xpath は名前空間やプレフィックスを無視して指定
   -cert <p12/x509/finger/select/card/p11> : 署名証明書指定 (署名時必須)
      p12 filepath passwd : PKCS#12 指定 P12 ファイルとパスワードが必要
      x509 filepath       : 指定証明書をファイルから指定(仮署名用)
      finger HEX          : Windows 証明書ストアから指紋(HEX 文字列)で指定
      select              : Windows 証明書ストアから選択して取得
      card type [numb]    : IC カード利用 (種別指定)
        1 : JPKI Sign : 'JPKI Crypto Service Provider for Sign'
        2 : JPKI Auth : 'JPKI Crypto Service Provider for Auth'
        3 : GPKI/JPKI(Old) : 'JPKI Crypto Service Provider'
        4 : LGPKI/DIACERT : 'Melco Standard-9 Enhanced Cryptographic SP'
        5 : NDN(GoSign/AOSign) : 'NEC Secure Ware AES Cryptographic Provider'
        6 : e-Probatio/ToiNX : 'DNP Standard-9 Cryptographic SP'
        7 : CRPKI(CAPI) : 'CRPKI Crypto Service Provider for Remote Sign'
        8 : Pentio/TDB/LGPKI : MS_SCARD_PROV_A (MINI DRIVER)
      card cng [numb]     : CNG の IC カード利用 (種別指定)
        1 : CRPKI(CNG) : 'CRPKI Key Storage Provider for Remote Sign'
      p11 filepath passwd : PKCS#11 指定 P11 の DLL ファイルとパスワードが必要
   -hash <s256/s384/s512/sha1> : 署名と参照のハッシュ方式 省略時=s256
      s256 : SHA-2 (256 ビット)
      s384 : SHA-2 (384 ビット)
      s512 : SHA-2 (512 ビット)
      sha1 : SHA-1 (非推奨:160 ビット)
   -c14n <10/10wC/EX/EXwC/11/11wC/NOT> : XML の正規化方法指定 省略時=10
      10   : C14N 1.0
      10wC : C14N 1.0 with Comment
      EX   : C14N 1.0 Exclusive
      EXwC : C14N 1.0 Exclusive with Comment
      11   : C14N 1.1
      11wC : C14N 1.1 with Comment
      NOT  : (NOT USE)
   -kopt <path|noval> : KeyInfo オプション指定
      path  : ds:KeyInfo に認証パス証明書全てを入れる
      noval : ds:KeyInfo に ds:KeyValue を含めない
   -xopt <nostime|nocert2|usedof> : XAdES 要素オプション指定
      nostime : SigningTime 要素を追加しない
      nocert2 : SigningCertificateV2 要素を使わず V1 を利用
      usedof  : SignedDataObjectProperties/DataObjectFormat 要素を追加する
   -prefix ds [xs] : prefix 利用と指定 省略時=prefix 無し
   -xmldsig : XML 署名(XAdES Object を追加しない) 省略時=XAdES 生成
   -make       : 仮署名 (署名値は生成しない)
   -value hash : 仮署名 (試験用:ハッシュ値から署名値を計算)
   -setval val : 仮署名 署名値埋め込み (make 済みファイルに署名値埋め込み)
```

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

```
options: 検証オプション(署名時)
  -vopt <nochk|full|post|nonet|pcrl> : 署名時検証オプション指定
     nochk : 署名前に証明書のチェック(証明書チェーンの確認等)をしない
     full  : 署名前に PKI 検証もおこなう(正常検証時のみ署名実行)
     post  : 署名後に検証する
     nonet : 検証時ネット接続しない(full 指定時に有効)
     pcrl  : CRL 優先検証(full 指定時に有効)
  -sflag <none/org|win> : 証明書ストア指定フラグ
     none : 証明書ストア無し(非推奨)
     org  : 独自証明書ストア利用
     win  : Windows 証明書ストア利用(Windows 版のみ)
  -store dirpath  : 独自証明書ストアの指定(※証明書/CRL/OCSP が設定可能)
  -repository url : ディレクトリサーバの指定
  -ocsp url       : RFC 6960 OCSP レスポンダの指定
  -revoproxy url  : LE 独自検証プロキシサーバの指定
  -http <http|inet|sock> : HTTP 通信 API 種別の指定 (省略時は http)
     http : WinHTTP 利用 (推奨/省略時設定)
     inet : WinInet 利用 (非推奨/過去互換)
     sock : 独自 Socket 利用 (OpenSSL 利用)

options: 署名タイムスタンプ(XAdES-T)追加オプション(署名時)
  -ts <3161/amano> : タイムスタンプ指定
     3161 url [hash] [id] [passwd] : RFC3161 (id/passwd 指定で Basic 認証対応)
        ※ [hash] には <s256/s384/s512/sha1> が指定可能
     amano url licensefile passwd  : AMANO タイムスタンプサービス(有償)
        ※ URL/ライセンスファイル/パスワードが必要
  -topt <full> : タイムスタンプ付与時オプション指定
     full : Encoding 等のオプション属性を全て付ける(省略時は必要最小限)
  -http <http|inet|sock> : HTTP 通信 API 種別の指定 (省略時は http)
     http : WinHTTP 利用 (推奨/省略時設定)
     inet : WinInet 利用 (非推奨/過去互換)
     sock : 独自 Socket 利用 (OpenSSL 利用)
  -tserr [dir] : タイムスタンプ応答エラーの出力フォルダ指定 (省略時は".")
  ※ 署名タイムスタンプは後から追加も可能

options: 出力オプション
  -out filepath : 出力 XAdES ファイル(必須)
  -in xmlfile   : 署名するベース XML を指定(任意)
  -debug dir    : 署名時のデバッグ情報出力指定
  -v            : 処理を表示する


> Lx3Cmd.exe -time [-options] -in input.xml -out output.xml

options: タイムスタンプ(XAdES-T/XAdES-A)追加
  -ts <3161/amano> : タイムスタンプ指定(必須)
     3161 url [hash] [id] [passwd] : RFC3161 (id/passwd 指定で Basic 認証対応)
        ※ [hash] には <s256/s384/s512/sha1> が指定可能
     amano url licensefile passwd  : AMANO タイムスタンプサービス(有償)
        ※ URL/ライセンスファイル/パスワードが必要
  -topt <full|a132> : タイムスタンプ付与時オプション指定
     full : Encoding 等のオプション属性を全て付ける(省略時は必要最小限)
```

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

```
      a132 : ArchiveTimeStamp V1.3.2 を利用(新規非推奨)
  -http <http|inet|sock> : HTTP 通信 API 種別の指定 (省略時は http)
      http : WinHTTP 利用 (推奨/省略時設定)
      inet : WinInet 利用 (非推奨/過去互換)
      sock : 独自 Socket 利用 (OpenSSL 利用)
  -tserr [dir] : タイムスタンプ応答エラーの出力フォルダ指定 (省略時は".")
  -snum num : 複数署名時の対象署名番号指定(省略時は最初の署名)
  -prefix xs141 : ArchiveTimeStamp の prefix 指定 省略時=prefix 無し
  -def filepath : URI 未指定の参照デフォルト設定(XAdES-A)

options: 入出力オプション
  -in filepath  : 入力 XAdES ファイル(必須)
  -out filepath : 出力 XAdES ファイル(必須、入力と同じファイルを指定可能)
  -debug dir : タイムスタンプ取得時のデバッグ情報出力指定


> Lx3Cmd.exe -verify [-options] -in input.xml [-out output.xml]

  ※ XAdES/XML 署名/Manifest フォーマットの検証が可能

options: 検証オプション
  -sflag <none/org|win> : 証明書ストア指定フラグ
      none : 証明書ストア無し(非推奨)
      org  : 独自証明書ストア利用
      win  : Windows 証明書ストア利用(Windows 版のみ)
  -store dirpath  : 独自証明書ストアの指定(※証明書/CRL/OCSP が設定可能)
  -repository url : ディレクトリサーバの指定
  -ocsp url       : RFC 6960 OCSP レスポンダの指定
  -revoproxy url  : LE 独自検証プロキシサーバの指定
  -http <http|inet|sock> : HTTP 通信 API 種別の指定 (省略時は http)
      http : WinHTTP 利用 (推奨/省略時設定)
      inet : WinInet 利用 (非推奨/過去互換)
      sock : 独自 Socket 利用 (OpenSSL 利用)
  -dir dirpath : 外部ファイル参照時のホームディレクトリ指定
  -ocsp url : OCSP(RFC 6960)レスポンダの指定
  -prior crl : 失効情報優先フラグ指定
      crl : CRL 優先、省略時は OCSP 優先
  -addvalid : 取得した検証情報(CRL/OCSP)を独自ストアに追加する
  -opt <constr|usecvs> : 検証オプション指定
      constr : 基本制約による CRL/ARL 判別をおこなう (CVS 等で指定を推奨)
      usecvs : CVS で検証済みの情報を使う場合に指定する
  -revoproxy url : LangEdge 独自検証プロキシサーバの指定
  -stime GeneralizedTime : 検証時刻の指定 省略時=現在 (試験用)
      ※ GeneralizedTime 型 例="20221204123456+0900"
  -not <ldap|http|ocsp|all> : 検証情報取得フラグ指定 (未取得エラーにしない)
      ※ -embed を使う時には -not オプションを指定してはいけない
      ldap : LDAP 経由の CRL/証明書取得を行わない
      http : HTTP 経由の CRL 取得を行わない
      ocsp : OCSP 取得を行わない
      all  : LDAP/HTTP 経由の CRL 取得と OCSP 取得を行わない
  -nochk <refs|mani|cert> : 非対象とする検証項目の指定
      refs : 全参照先(Reference)のハッシュ値チェックを行わない
      mani : 再帰参照先(Manifest)のハッシュ値チェックを行わない
```

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

```
      cert : 証明書の PKI 検証を行わない
  -use <stime|all> : 利用するオプション検証項目の指定
      stime : XAdES の SigningTime 要素を XAdES-BES の署名時刻として利用する
      all   : 署名中の全検証情報を再利用する
  -gpchk <sign|tsa> : 非対象とする検証項目の指定
      sign : 署名証明書認証パス用 CRL の猶予期間を考慮する
      tsa  : TSA 証明書認証パス用 CRL の猶予期間を考慮する
  -root certfile : 署名証明書の認証パス構築時のルート証明書を指定
  -basic [id] [passwd] : CRL/OCSP 取得時の Basic 認証対応
  -snum num : 複数署名時の対象署名番号指定(省略時かマイナス値は全ての署名)
  -prefix xs141 : TimeStampValidationData の prefix 指定 省略時=prefix 無し
  -def filepath : URI 未指定の参照デフォルト設定

options: 検証結果オプション
  -report : 検証結果レポートの画面出力
  -result [resultfile] : 検証結果 XML ファイルの出力
  -value  : 検証結果 XML に証明書/CRL/OCSP を HEX で追加
  -debug [dir] : デバッグ出力指定 (署名対象や各参照先をファイルに保存)

options: 検証情報の埋め込み(XAdES-XL/XAdES-A 検証情報追加)
  -embed : XAdES-XL/XAdES-AXL 用の検証情報の埋め込みを行う
  -valid resultfile : 検証結果 XML ファイルから検証情報を読み込み利用する
  -eopt <etst> : 検証情報埋め込みオプション指定
      etst : 失効情報(CRL のみ対応)をタイムスタンプトークンに埋め込む(非推奨)

options: 入出力オプション
  -in filepath  : 入力 XAdES ファイル(必須)
  -out filepath : 出力 XAdES ファイル(オプション、検証情報埋め込み時は必須)


> Lx3Cmd.exe -schema [-options] -in input.xml

options: XML スキーマチェックオプション

  -sfile filepath  : チェックするスキーマファイル(.xsd)を指定
      ※ -sfile の未指定時には "xsds/XAdES01903v141-201601.xsd" が利用される
  -result filepath  : チェック結果を保存するファイルを指定(未指定は標準出力)

options: 入出力オプション
  -in filepath : 入力 XAdES ファイル(必須)


> Lx3Cmd.exe -info [-options] -in input.xml [-out output.xml]

options: 情報取得オプション

  -list : 入力 XML ファイル中の Signature 要素の Id リスト(Id/XPath/Level)を表示
  -level : 入力 XML ファイルの XAdES レベルの出力 (LX3_XADES_LEVEL が返る)
  -refs : 指定された署名の全ての参照(Reference)情報の出力
  -snum num : 複数署名時の対象署名番号指定(省略時は最初の署名)

  note: XAdES Level
    NO-SIGN    (OO) : no Signature.
```

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

```
    XmlDsig    (01) : XML Signature (not XAdES).
    XAdES-MAKE (09) : XAdES make Signature (not singed).
    XAdES-BES  (10) : XAdES Signature only.
    XAdES-T    (11) : XAdES-BES + SignatureTimestamp.
    XAdES-XL   (12) : XAdES-T + Certificates and Revocation datas.
    XAdES-A    (13) : XAdES-XL + ArchiveTimestamp.
    XAdES-A-XL (14) : XAdES-A + Certificates and Revocation datas.

options: レベルダウン操作(必須：-out 指定)
  -cut <BES/T> : レベルダウン出力する XAdES 種別 (BES=XAdES-BES/T=XAdES-T)

options: 入出力オプション
  -in filepath   : 入力 XAdES ファイル(必須)
  -out filepath  : 出力 XAdES ファイル(オプション、レベルダウン時に指定)


> Lx3Cmd.exe -refs [-options] -out manifest.xml

options: 参照生成操作(Manifest) : 出力指定(-out)必須
 // 参照(ファイルを参照する形式): Detached options.
  -det filepath [mime] : 外部 Detached(ファイル指定)形式の署名対象追加
     mime:<MimeType>    : Mime-Type を指定する
       例) "mime:application/xml"
  -detdir dirpath      : ディレクトリ下全ファイルの Detached 署名対象追加
  -dlist dlistfile     : Detached 形式の署名対象をリストファイルから追加
 // 間接(対象群を含む Manifest を参照する形式): Manifest options.
  -dmani mfilepath  : Manifest ファイルの Detached(ファイル指定)追加
  -dmlist mlistfile : Manifest ファイルをリストファイルから Detached 追加
  // 共通オプション: Common options.
  -dir dirpath : 外部ファイル参照時のホームディレクトリ指定(-detdir 不可)

options: Manifest の追加オプション
  -id name : Manifest 要素の Id 属性名の指定
  -hash <s256/s384/s512/sha1> : 参照ハッシュ方式 省略時=s256
     s256 : SHA-2 (256 ビット)
     s384 : SHA-2 (384 ビット)
     s512 : SHA-2 (512 ビット)
     sha1 : SHA-1 (非推奨:160 ビット)
  -c14n <10/10wC/EX/EXwC/11/11wC/NOT> : XML の正規化方法指定 省略時=NOT
     10   : C14N 1.0
     10wC : C14N 1.0 with Comment
     EX   : C14N 1.0 Exclusive
     EXwC : C14N 1.0 Exclusive with Comment
     11   : C14N 1.1
     11wC : C14N 1.1 with Comment
     NOT  : (NOT USE)
     ※ NOT であっても拡張子が'.xml'の場合にはデフォルト C14N 正規化される
  -prefix ds : Manifest の prefix 指定 省略時=prefix 無し

options: 入出力オプション
  -out filepath  : 出力 Manifest ファイル(必須)
```

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

```
> Lx3Cmd.exe -server [-options]
  -server  : 仮署名付与/署名値埋込(サーバ機能)
  -options : オプション指定

note: 利用方法
  サーバ機能はクライアントからの要求に応じて以下の処理を行う
     LCS_CERT: 仮署名 > 仮署名 PDF 出力とハッシュ値 XML(LCS_HASH)出力
     LCS_SIGN: 署名値埋め込み > 署名済み XAdES 出力と結果 XML(LCS_RSLT)出力
  ※ 入力(-in)と出力(-out)が未指定の場合には以下の種別情報を出力する
     証明書 XML: <LCS_CERT sid="XXXXX" time="YYYYMMDDhhmmssZ" />
     署名値 XML: <LCS_SIGN sid="XXXXX" time="YYYYMMDDhhmmssZ" />
     エラー: <ERROR mesg="エラーメッセージ" />
  ※ 生成されるハッシュ値 XML(LCS_HASH)または結果 XML(LCS_RSLT)は標準出力される

options: オプション
  -req filepath : 証明書/署名値 XML ファイル (標準入力でも指定可能)
  -check : 証明書/署名値 XML ファイルの種別判定
  -hex <on/off> : HEX モード指定(通常指定不要) 省略時=入力と同じモード

options: LCS_HASH (仮署名)時オプション
 // 参照(ファイルまたは Id を参照する形式): Detached options.
  -det filepath [opts] : 外部 Detached(ファイル指定)形式の署名対象追加
  -det #Id [opts]      : 内部 Detached(Id 指定)形式の署名対象追加
  -detdir dirpath      : ディレクトリ下全ファイルの Detached 署名対象追加
  -dlist dlistfile     : Detached 形式の署名対象をリストファイルから追加
 // 内包(署名の中に対象を埋め込む形式): Enveloping options.
  -eping filepath [opts] : Enveloping 形式の署名対象追加
  -elist elistfile       : Enveloping 形式の署名対象をリストファイルから追加
     ※ 拡張子が.xml か MimeType が xml の場合以外ではバイナリとして追加
 // 外包(対象の中に署名を埋め込む形式): Enveloped options.
  -eped [xpath] : Enveloped 形式の署名対象追加、1 つのみ指定可
     ※ [xpath] に XPath オプションを指定可能
 // 間接(対象群を含む Manifest を参照する形式): Manifest options.
  -dmani mfilepath  : Manifest ファイルの Detached(ファイル指定)追加
  -dmlist mlistfile : Manifest ファイルをリストファイルから Detached 追加
  -emani mfilepath  : Manifest を読み込み Enveloping(埋め込み)追加
  -emlist mlistfile : Manifest ファイルをリストファイルから Enveloping 追加
  // 共通オプション: Common options.
  -dir dirpath : 外部ファイル参照時のホームディレクトリ指定(-detdir 不可)
 // 補足: Supplements.
  ※ dlist/elist/mlist のリストファイルは 1 行 1 対象ファイルパスで記述
  ※ det #Id/eped は -in オプションの同時指定が必須
  ※ [opts] には以下の引数が指定可能(-det #Id は Description のみ)
     mime:<MimeType>     : Mime-Type を指定する
       例) "mime:text/plain"
     encd:<Encoding>     : Encoding を指定する
       例) "encd:http://www.ietf.org/rfc/rfc2279.txt"
     desc:<Description> : Description(説明)を指定する
       例) "desc:This is test."
  -id name : Sinature 要素の Id 属性名の指定
  -xpath xpath : -in 指定時の署名挿入位置の XPath 指定
     ※ xpath は名前空間やプレフィックスを無視して指定
  -hash <s256/s384/s512/sha1> : 署名と参照のハッシュ方式 省略時=s256
```

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

```
      s256 : SHA-2 (256 ビット)
      s384 : SHA-2 (384 ビット)
      s512 : SHA-2 (512 ビット)
      sha1 : SHA-1 (非推奨:160 ビット)
  -c14n <10/10wC/EX/EXwC/11/11wC/NOT> : XML の正規化方法指定 省略時=10
      10   : C14N 1.0
      10wC : C14N 1.0 with Comment
      EX   : C14N 1.0 Exclusive
      EXwC : C14N 1.0 Exclusive with Comment
      11   : C14N 1.1
      11wC : C14N 1.1 with Comment
      NOT  : (NOT USE)
  -kopt <path|noval> : KeyInfo オプション指定
      path  : ds:KeyInfo に認証パス証明書全てを入れる
      noval : ds:KeyInfo に ds:KeyValue を含めない
  -xopt <nostime|nocert2|usedof> : XAdES 要素オプション指定
      nostime : SigningTime 要素を追加しない
      nocert2 : SigningCertificateV2 要素を使わず V1 を利用
      usedof  : SignedDataObjectProperties/DataObjectFormat 要素を追加する
  -prefix ds [xs] : prefix 利用と指定 省略時=prefix 無し
  -xmldsig : XML 署名(XAdES Object を追加しない) 省略時=XAdES 生成

options: LCS_RSLT (署名値埋め込み)時オプション
  -snum num : 複数署名時の対象署名番号指定(省略時は最初の署名)
  -redirect url : 処理完了時のリダイレクト URL 指定 省略時=未指定

options: 入出力オプション
  -in filepath  : 入力ファイル(LCS_HASH=署名フィールド PDF/LCS_RSLT=仮署名 PDF)
  -out filepath : 出力ファイル(LCS_HASH=仮署名 PDF/LCS_RSLT=署名済み PDF)

// 例) STEP1: クライアント側 (LCS_CERT) : 証明書選択 > 証明書 XML
 > Lx3Cmd -client -sid T01 -cert x509 sign.cer > CERT.xml

// 例) STEP2: サーバ側 (LCS_HASH) : 仮署名 < 証明書 XML > ハッシュ値 XML
 > Lx3Cmd -server -det ... -out make.xml < CERT.xml > HASH.xml

// 例) STEP3: クライアント側 (LCS_SIGN) : 署名値計算 < ハッシュ値 XML > 署名値 XML
 > Lx3Cmd -client -sid T01 -cert p12 sign.p12 PSWD < HASH.xml > SIGN.xml

// 例) STEP4: サーバ側 (LCS_RSLT) : 署名値埋め込み < 署名値 XML > 結果 XML
 > Lx3Cmd -server -in make.xml -out xades.xml < SIGN.xml > RSLT.xml

// 例) 種別確認
 > Lx3Cmd -server < CLIENT_REQEST.xml


> Lx3Cmd.exe -client [-options]
  -client  : 証明書選択と署名値生成(クライアント署名機能試験用)
  -options : オプション指定

note: 利用方法
  LCS_CERT: ハッシュ値 XML(LCS_HASH)が未指定の場合は証明書選択 (初期化操作)
  LCS_SIGN: -res 引数か標準入力でハッシュ値 XML(LCS_HASH)を指定して署名値生成
```

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

```
  ※ 生成される証明書 XML(LCS_CERT)または署名値 XML(LCS_SIGN)は標準出力される

options: LCS_CERT (証明書選択)時オプション
  -cert <p12/x509> : 署名証明書指定 (必須)
     p12 filepath passwd : PKCS#12 指定 ファイルとパスワードが必要
     x509 filepath       : X.509 証明書指定

options: LCS_SIGN (署名値生成)時オプション
  -res filepath : ハッシュ値 XML(LCS_HASH)ファイル (必須:標準入力指定可能)
  -cert <p12>   : 署名証明書指定 (必須)
     p12 filepath passwd : PKCS#12 指定 ファイルとパスワードが必要

options: 共通オプション
  -sid idstr : セッション ID の指定(必須:任意 ID 文字列)

// 例) STEP1: クライアント側 (LCS_CERT):証明書選択 > 証明書 XML
 > Lx3Cmd -client -sid T01 -cert x509 sign.cer > CERT.xml

// 例) STEP2: サーバ側 (LCS_HASH):仮署名 < 証明書 XML > ハッシュ値 XML
 > Lx3Cmd -server -det ... -out make.xml < CERT.xml > HASH.xml

// 例) STEP3: クライアント側 (LCS_SIGN):署名値計算 < ハッシュ値 XML > 署名値 XML
 > Lx3Cmd -client -sid T01 -cert p12 sign.p12 PSWD < HASH.xml > SIGN.xml

// 例) STEP4: サーバ側 (LCS_RSLT):署名値埋め込み < 署名値 XML > 結果 XML
 > Lx3Cmd -server -in make.xml -out xades.xml < SIGN.xml > RSLT.xml

// 例) 種別確認
 > Lx3Cmd -server < CLIENT_REQEST.xml
```

Lx3Cmd-Help.txt

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

２．２．Windows C++ 利用の場合

１）include フォルダをインクルードディレクトリに追加

C/C++設定の "追加のインクルード ディレクトリ" で include 直下を指定する。
インクルードする場所はリリースファイルを展開したディレクトリ（LeXAdES-3.XX.RX）下にある include ディレクトリ。
LeXAdES3.h をソースから以下のようにインクルードして API を利用する。

```
// LE:XAdES:Lib V3 インクルードファイル
#include <LeXAdES3/LeXAdES3.h>
```

２）lib_win フォルダをライブラリディレクトリに追加

リンカ設定の "追加のライブラリ ディレクトリ" で以下フォルダを指定する。

32bit リリース版は lib_win/Release を指定
32bit デバッグ版は lib_win/Debug を指定
64bit リリース版は lib_win/Release64 を指定
64bit デバッグ版は lib_win/Debug64 を指定

LeXAdES3.lib と LePKI.lib をリンカ設定の "追加の依存ファイル" で指定するか、以下をソースに追加する。

```
// LE:XAdES:Lib V3 インターフェイスライブラリファイル
#pragma comment(lib, "LeXAdES3.lib")
#pragma comment(lib, "LePKI.lib")
```

３）bin_win フォルダを環境変数の PATH に追加

32bit リリース版は bin_win/Release を PATH に追加
32bit デバッグ版は bin_win/Debug を PATH に追加
64bit リリース版は bin_win/Release64 を PATH に追加
64bit デバッグ版は bin_win/Debug64 を PATH に追加

※ 実例として sample/LeXAdES3/cpp の下にあるサンプルソースとプロジェクト cpp.sln を参照。

ファイル	概要
CppBuild.bat	バッチ式のビルド一括実行
CppAll.bat	一括テスト実行
CppXaesTest.bat CppSign.cpp	XAdES 署名/検証サンプル
CppClientTest.bat CppClient.cpp	クライアント署名サンプル

Windows C++用のサンプル

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

２．３．Linux C++ 利用の場合

０）前準備：「１．６．Linux 環境のインストールとソースビルド」に従いインストールを行う

C++のAPIを利用するので1-4A）または1-4B）の手順に従って環境のセットをする必要がある。

１）include フォルダをコンパイル時の -I オプションにてインクルードディレクトリに指定

インクルードする場所はリリースファイルを展開したディレクトリ（LeXAdES-3.XX.RX）下にある include ディレクトリ。

```
$ g++ -I../include sample.cpp
```

LeXAdES3.h をソースから以下のようにインクルードして API を利用する。

```
// LE:XAdES:Lib V3 インクルードファイル
#include <LeXAdES3/LeXAdES3.h>
```

２）lib_linux フォルダをリンク時の -L オプションにてリンクディレクトリに指定

リンクディレクトリ場所はリリースファイルを展開したディレクトリ(LeXAdES-3.XX.RX)下にある lib_linux ディレクトリ。

リンク時の引数に -lLeXAdES3 と -lLePKI を付け、libLeXAdES3.so と libLePKI.so を指定

```
$ g++ -o sample -L../lib_linux sample.o -lLeXAdES3 -lLePKI -lm -ldl -lstdc++
```

※ 実例として sample/LeXAdES3/cpp の下にあるサンプルソースと CppBuild.sh 他を参照。

ファイル	概要
CppBuild.sh	バッチ式のビルド一括実行
CppAll.sh	一括テスト実行
CppXaesTest.sh CppSign.cpp	XAdES 署名/検証サンプル
CppClientTest.sh CppClient.cpp	クライアント署名サンプル

Linux C++用のサンプル

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

２．４．Java 利用の場合

０）前準備：「１．６．Linux 環境のインストールとソースビルド」に従いインストールを行う

Java の API を利用するので 1-4A）または 1-4B）の手順に従って環境のセットをする必要がある。

Windows 環境では、リリースファイルを展開したディレクトリ（LeXAdES-3.XX.RX）下にある bin_win ディレクトリを PATH 環境変数に追加

１）パッケージを import する

Java ソースに LeXAdES V3 と LePKI を以下のようにインポートしておく。

```
import jp.langedge.LeXAdES3.*;
import jp.langedge.LePKI.*;
```

２）jar ファイルを classpath に指定する

コンパイルと実行時に classpath として lexades-3.0.XX.jar と lepki-1.0.XX.jar を指定。
XX はバージョン番号（例 V1.0.0 なら”0”）

```
// コンパイル
javac -classpath .;lexades-3.0.XX.jar;lepki-1.0.XX.jar Sample.java
// 実行
java -classpath .;lexades-3.0.XX.jar;lepki-1.0.XX.jar Sample
```

※ 実例として sample/LeXAdES3/java の下にあるサンプルソースと JavaBuild.bat 他を参照。

ファイル	概要
JavaBuild.bat	バッチ式のビルド一括実行
JavaAll.bat	一括テスト実行
JavaXadesTest.bat JavaXadesTest.sh LeXAdES_test.java	XAdES 署名/検証サンプル
JavaClientTest.bat JavaClientTest.sh LeXAdES_client.java	クライアント署名サンプル

Java 用のサンプル

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

注意：Java 環境におけるネイティブメモリの解放について

LE:XAdES:Lib V3 の Java クラスは全て JNI を利用しておりメモリもほとんど Java 管理外のネイティブ側で管理されている。この為に Java のガベージコレクターはあまりメモリを使っていないと判断してしまいすぐに解放されずメモリ不足になる場合がある。特にネイティブなメモリを消費するクラスは LeXAdES3 と LePKI である。

LE:XAdES:Lib V3 の各 Java クラスに用意されている finalize() を呼び出す事でネイティブ側で確保されたメモリが解放される。LeXAdES3 クラスと LePKI クラスは利用後に必ず finalize() を呼び出すこと。他のクラスに関しても出来れば利用後明示的に finalize() を呼び出すことを推奨する。詳しくはサンプルのソースを参照。

クラス名	ネイティブメモリ利用	補足
LeXAdES3	XML ファイルのサイズに依存 通常大量のメモリを消費する	最もネイティブメモリを消費するクラスなので必ず finalize() を呼び出す
LePKI	独自証明書ストアの利用に依存 比較的多くのメモリを消費する	比較的ネイティブメモリを消費するクラスなので必ず finalize() を呼び出す
LeReference LeManifest LpkCert	ある程度メモリを消費する	出来れば finalize() を呼び出す
LpkCrl	CRL のサイズに依存 ある程度メモリを消費する	大きな CRL を利用する場合は必ず、それ以外でも出来れば finalize() を呼び出す
XdaVerifyXml XdaClientXml	XML のサイズに依存 ある程度メモリを消費する	出来れば finalize() を呼び出す

ネイティブメモリを必要とする主なクラス

注意：Java 環境における 32bit と 64bit の問題について

LE:XAdES:Lib V3 の Java クラスは全て JNI を利用している為に、JNI からよびだされるネイティブ部と Java 本体の 32bit/64bit 環境が一致している必要がある。64bit の Java をご利用の場合には Linux は 64bit 版を、Windows は Release64 フォルダ下を、それぞれ使う必要がある。

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

２．５．.NET API 利用の場合

.NET は Windows 環境のみのサポートとなり、Linux 環境では現在.NET の API は非サポートです。.NET 用の LeXAdES3dnet.dll / LePKIdnet.dll はマネージド DLL ですのでアセンブリの関係で PATH 環境変数が通った場所に置いても利用できません。利用する実行ファイル（例: CsSign.exe）と同じディレクトリに入れることを推奨します。

モジュール	.NET 用ラッパマネージド DLL	本体アンマネージド DLL
LeXAdES V3	LeXAdES3dnet.dll	LeXAdES3.dll
LePKI	LePKIdnet.dll	LePKI.dll

LeXAdES3dnet.dll / LePKIdnet.dll をどうしても実行ファイルとは別のフォルダに置きたい場合には、<実行ファイル名>.config と DEVPATH 環境変数を使う方法がある。ただしこれは開発者向けの高度な利用方法なので、実行ファイルと同じ場所でのご利用を推奨する。詳しくは CsSign.exe.config.sample の中のコメントや以下サイトを参照。なお本体およびその他の DLL はアンマネージド DLL なので、PATH 環境変数が通った場所であればどこにあっても構わない。表示/取得されるバージョン番号は本体アンマネージド DLL のものとなる。

参考 https://msdn.microsoft.com/ja-jp/library/cskzh7h6%28v=vs.110%29.aspx
注：利用時には DEVPATH 環境変数と PATH 環境変数の両方を指定する必要がある。

CsSign.exe.config.sample	説明
<configuration> <runtime> <dependentAssembly> <assemblyIdentity name="LeXAdES3dnet" /> <assemblyIdentity name="LePKIdnet" /> </dependentAssembly> <developmentMode developerInstallation="true" /> </runtime> </configuration>	Configuration 開始 runtime 指定 dependentAssembly 設定 LeXAdES3 assemblyIdentity 名指定 LePKI assemblyIdentity 名指定 開発者モードセット ※1

※1 developmentMode developerInstallation が true の時に DEVPATH が有効になる。

利用方法：LeXAdES3dnet.dll / LePKIdnet.dll を参照に追加する

```
le::LeXAdES3 xades = new le::LeXAdES3();  // 生成
String version = xades.getVersion(); // 利用（バージョン番号は本体 dll から取得）
```

※ 実例として sample/LeXAdES3/cs の下にあるサンプルソース等を参照。

ファイル	概要
CsBuild2015.bat CsBuild2017.bat CsBuild2019.bat CsBuild2022.bat	バッチ式ビルド一括実行
CsAll.bat	一括テスト実行
CsXadesSignTest.bat CsXadesSign/Program.cs , Release64/CsXadesSign.exe.config.sample, Release/CsXadesSign.exe.config.sample	署名サンプル

.NET C#用のサンプル

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

注意：.NET 環境におけるネイティブメモリの解放について

LE:XSAdES:Lib V3 の.NET クラスはマネージドコードを利用していますが、内部は C++で記述されておりアンマネージドコードで構成されている。メモリもほとんどアンマネージドのネイティブ側で管理されている。この為に.NET のガベージコレクターはあまりメモリを使っていないと判断してしまいすぐに解放されずメモリ不足になる場合がある。特にネイティブなメモリを消費するクラスは LeXAdES3 と LePKI である。

LE:XAdES:Lib V3 の各 .NET クラスに用意されている finalize() を呼び出す事でネイティブ側で確保されたメモリが解放される。LeXAdES3 クラスと LePKI クラスは利用後に必ず finalize() を呼び出すこと。他のクラスに関しても出来れば利用後明示的に finalize() を呼び出すことを推奨する。詳しくはサンプルのソースを参照。

クラス名	ネイティブメモリ利用	補足
LeXAdES3	XML ファイルのサイズに依存 通常大量のメモリを消費する	最もネイティブメモリを消費するクラスなので必ず finalize() を呼び出す
LePKI	独自証明書ストアの利用に依存 比較的多くのメモリを消費する	比較的ネイティブメモリを消費するクラスなので必ず finalize() を呼び出す
LeReference LeManifest	ある程度メモリを消費する	出来れば finalize() を呼び出す
LpkCrl	CRL のサイズに依存 ある程度メモリを消費する	大きな CRL を利用する場合は必ず、それ以外でも出来れば finalize() を呼び出す
XdaVerifyXml XdaClientXml	XML のサイズに依存 ある程度メモリを消費する	出来れば finalize() を呼び出す

ネイティブメモリを必要とする主なクラス

注意：.NET 環境における 32bit と 64bit の問題について

LE:XAdES:Lib V3 の.NET クラスは内部的に C++を利用している為に、.NET からよびだされるネイティブ部と.NET 本体の 32bit/64bit 環境が一致している必要がある。64bit の.NET 環境をご利用の場合には Release64 フォルダ下を使う必要がある。

注意：必要となる.NET Framework のバージョンはビルド環境に依存する

Visual Studio 2015/2017/2019 版　　：.NET Framework v4.5.2 以上
Visual Studio 2022 版　　：.NET Framework v4.7.2 以上

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

２．６．Lx3Cmd コマンドの利用例

LE:XAdES:Lib V3 はライブラリ製品ではあるが、簡単に機能を利用するコマンドプログラム Lx3Cmd が提供されている。利用方法の詳細はヘルプコマンド（Lx3Cmd -help）を参照するとして、ここでは簡単に署名や検証の利用例を説明する。

１A）署名付与：署名対象となる参照の指定方法（XAdES-BES）

○ 複数外部ファイル Detached 参照と署名付与（PKCS#12 ファイルの指定）

```
＞ Lx3Cmd -sign -det test.txt -det image.png -cert p12 LeTest.p12 test -out signed1.xml
```

○ ファイル読み込み Enveloping 参照と署名付与（PKCS#12 ファイルの指定）

```
＞ Lx3Cmd -sign -eping test.txt -cert p12 LeTest.p12 test -out signed2.xml
```

○ Enveloped 参照と署名付与（埋め込み対象の XML を -in オプションで指定）

```
＞ Lx3Cmd -sign -eped -cert p12 LeTest.p12 test -in base.xml -out signed3.xml
```

○ Detached の内部 Id 参照と署名付与（対象の XML を -in オプションで指定）

```
＞ Lx3Cmd -sign -det #test -cert p12 LeTest.p12 test -in base.xml -out signed4.xml
```

○ Detached の内部 Id 参照と XPATH 指定の署名付与（対象の XML を -in オプションで指定）

```
＞ Lx3Cmd -sign -det #test -xpath /LeXMLData/SignRoot -cert p12 LeTest.p12 test ¥
    -in base.xml -out signed4.xml
```

○ Enveloped 参照と外部ファイル Detached 参照と署名付与（複数参照指定 1）

```
＞ Lx3Cmd -sign -eped -det test.txt -cert p12 LeTest.p12 test ¥
    -in base.xml -out signed5.xml
```

○ ベースディレクトリ指定の外部ファイル Detached 参照と署名付与（ベース位置指定）

```
＞ Lx3Cmd -sign -dir input_dir -det test.txt -det image.png -cert p12 LeTest.p12 test ¥
    -out signed6.xml
```

▼ ヒント

- -eped は 1 つのみ指定可能だが -det と -eping は複数指定が可能。
- -det と -eping は -detdir / -dlist / -elist を利用して複数ファイル一括指定も可能。
- -eped と組み合わせて -det と -eping の指定が可能。
- -eped でも xpath 指定による位置指定可能だが、標準ではルート要素直下となっている。
- -eped または -det の Id 指定の場合には署名済み XML ファイルを -in で指定することで複数署名となる。

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

１B）署名付与：署名オプションの指定方法（XAdES-BES / XmlDsig）

○ XML 署名の付与（XmlDsig の指定、タイムスタンプ等の長期署名は不可となる）
＞ Lx3Cmd -sign -det test.txt **-xmldsig** -cert p12 LeTest.p12 test -out xmldsig.xml

○ SHA-512 署名の付与（ハッシュ方式指定、Reference のハッシュアルゴリズムも同じとなる）
＞ Lx3Cmd -sign -det test.txt **-hash s512** -cert p12 LeTest.p12 test -out xades-bes1.xml

○ XML 正規化方法指定の署名の付与（以下例は 1.0 Exclusive、省略時は 1.0 Inclusive）
◎ Reference 毎に C14N 正規化を指定する場合にはコマンドではなく API の利用が必要です。
＞ Lx3Cmd -sign -det test.txt **-c14n EX** -cert p12 LeTest.p12 test -out xades-bes2.xml

○ KeyInfo へ認証パス証明書群の埋め込みと署名の付与（認証パス証明書群が必要）
＞ Lx3Cmd -sign -det test.txt **-kopt path** -cert p12 LeTest.p12 test -out xades-bes3.xml

○ XAdES の SigningTime を付けない署名の付与（欧州では必須だが日本では任意項目）
＞ Lx3Cmd -sign -det test.txt **-xopt nostime** -cert p12 LeTest.p12 test -out xades-bes4.xml

○ プレフィックス付き署名の付与（XmlDsig 名前空間=ds、XAdES132 名前空間=xs）
＞ Lx3Cmd -sign -det test.txt **-prefix ds xs** -cert p12 LeTest.p12 test -out xades-bes5.xml

○ Signature 要素の Id 指定付き署名の付与（省略時は同時生成 Id が使われる）
＞ Lx3Cmd -sign -det test.txt **-id MySignId** -cert p12 LeTest.p12 test -out xades-bes6.xml

○ 署名付与のオプションヘルプ
＞ Lx3Cmd -sign -help

２）タイムスタンプの追加（XAdES-T / XAdES-A）

○ 署名タイムスタンプ追加（入力ファイルが XAdES-BES でなければならない）
＞ Lx3Cmd -time **-ts 3161 %TS_URL%** -in xades-bes.pdf -out xades-t.pdf

○ アーカイブタイムスタンプ追加（入力ファイルが XAdES-XL でなければならない）
※ 入力ファイルの署名レベルが違うだけで署名タイムスタンプ追加と基本的に同じ
＞ Lx3Cmd -time **-ts 3161 %TS_URL%** -in xades-xl.pdf -out xades-a.pdf

○ アーカイブタイムスタンプ V1.3.2 追加（省略時は推奨の V1.4.1 を利用）
◎ 入力ファイルの署名レベルが違うだけで署名タイムスタンプ追加と基本的に同じ
＞ Lx3Cmd -time -ts 3161 %TS_URL% **-topt a132** -in xades-xl.pdf -out xades-a.pdf

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

○ タイムスタンプ追加（ハッシュ方式と Basic 認証付き）
◎ 以下例では、ハッシュ方式:SHA-512、ユーザーID:langedge、パスワード:test
＞ Lx3Cmd -time **-ts 3161 %TS_URL% s512 langedge test** -in xades-bes.pdf -out xades-t.pdf

○ タイムスタンプ追加（タイムスタンプ要素に Encoding 等のオプション属性を付ける）
＞ Lx3Cmd -time -ts 3161 %TS_URL% **-topt full** -in xades-bes.pdf -out xades-t.pdf

○ タイムスタンプ追加のオプションヘルプ
＞ Lx3Cmd -time -help

▼ ヒント
- 署名タイムスタンプは -sign による署名時に -ts オプションを指定することで署名と同時に追加することも可能です。
- 入力ファイルの署名レベルにより自動的に署名タイムスタンプ追加になるかアーカイブタイムスタンプ追加になるかが決まる。明示的に指定はできない。
- タイムスタンプ URL への接続可能な環境が必要です。特にプロキシ等を使っている場合にはプロキシ設定を確認してください。Windows の場合にはシステム設定となり、Linux の場合には独自設定が必要です。

3A）検証処理

○ 検証実行（検証結果レポート画面表示）
＞ Lx3Cmd -verify **-report** -in xades.xml

○ 検証実行（検証結果 XML 画面表示）
＞ Lx3Cmd -verify **-result** -in xades.xml

○ 検証実行（検証結果 XML のファイル保存）
＞ Lx3Cmd -verify **-result result.xml** -in xades.xml

○ 独自証明書ストア指定での検証実行（store の下に trusts や certs のディレクトリが必要）
＞ Lx3Cmd -verify -result **-store ../store** -in xades.xml

○ OCSP 優先での検証実行（CRL と OCSP の両方利用可能な場合に OCSP を優先）
＞ Lx3Cmd -verify -result **-prior ocsp** -in xades.xml

○ 外部接続しない検証実行（HTTP/LDAP 通信を行わず検証実行）
◎ 不足した失効情報は notUse と表示するがエラーにはしない、XAdES-XL/XAdES-A-XL なら影響なし
＞ Lx3Cmd -verify -result **-not all** -in xades.xml

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

○ 参照情報をチェックしない検証実行（Reference のチェックをしない）
＞ Lx3Cmd -verify -result **-nochk refs** -in xades.xml

○ SigningTime を署名時刻として検証実行（XAdES-BES の検証時）
＞ Lx3Cmd -verify -result **-use stime** -in xades.xml

○ 検証時刻を指定して検証実行（未指定時は現在時刻で検証される）
＞ Lx3Cmd -verify -result **-stime 20221204123456+0900** -in xades.xml

○ 参照ファイルベース場所指定での検証実行（Reference のベース位置指定）
＞ Lx3Cmd -verify -result **-dir ../input** -in xades.xml

▼ ヒント
- 検証結果レポートの表示は、検証結果 XML を解析して XdaVerifyXml クラスの getReport() メンバを使って取得した文字列を利用しています。
- API 利用する時の PKI 関連の検証設定は LePKI クラスに行い、LeXAdES3.setPki() メソッドでセットします。

3B）検証情報の追加（XAdES-XL / XAdES-A-XL）

○ 検証結果追加（XAdES-XL）
◎ -embed オプション指定時には -out オプションの指定が必須となります。
＞ Lx3Cmd -verify **-embed** -report -in xades-t.xml **-out xades-xl.xml**

○ 検証結果追加（XAdES-A-XL）
◎ 入力ファイルの署名レベルが XAdES-A の場合には XAdES-A-XL となります。
＞ Lx3Cmd -verify **-embed** -report -in xades-a.xml **-out xades-a-xl.xml**

○ 検証/検証情報追加のオプションヘルプ
＞ Lx3Cmd -verify -help

▼ ヒント
- 検証情報の追加は検証結果が VALID（正常）となっている必要があります。
- 入力ファイルの署名レベルにより自動的に XAdES-T への検証情報追加になるかアーカイブタイムスタンプ（XAdES-A）への検証情報追加になるかが決まる。明示的に指定はできない。
- 検証結果には、VALID（正常）・INDETERMINATE（不明）・INVALID（不正）・ERROR（異常）の 4 種類があります。INDETERMINATE は検証に必要な情報（トラストアンカー指定や失効情報等）が足りない状態です。検証結果レポートや XML から理由を確認してください。

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

４）Manifest 機能（参照の間接指定）

○ 複数外部ファイル Detached 参照の Manifest ファイル生成
＞ Lx3Cmd **-refs** -det test.txt -det image.png -out manifest1.xml

○ Manifest ファイル Detached 参照の Manifest ファイル生成（二重間接指定）
＞ Lx3Cmd -refs **-dmani manifest1.xml** -det test.txt -out manifest2.xml

○ ベースディレクトリ指定の Detached 参照の Manifest ファイル生成（ベース位置指定）
＞ Lx3Cmd -refs **-dir input_dir** -det test.txt -det image.png -out manifest3.xml

○ SHA-512 ハッシュ方式の指定（Reference のハッシュアルゴリズムの指定）
＞ Lx3Cmd -refs **-hash s512** -det test.txt -det image.png -out manifest4.xml

○ XML 正規化方法指定の Manifest 生成（以下例は 1.0 Exclusive、省略時は 1.0 Inclusive）
＞ Lx3Cmd -refs -det test.txt -det image.png **-c14n EX** -out manifest5.xml

○ プレフィックス付き署名の付与（XmlDsig 名前空間=ds）
＞ Lx3Cmd -refs -det test.txt -det image.png **-prefix ds** -out manifest6.xml

○ Manifest 要素の Id 指定（省略時は Id 無し）
＞ Lx3Cmd -refs -det test.txt -det image.png **-id ManiId** -out manifest7.xml

○ Manifest 生成のオプションヘルプ
＞ Lx3Cmd -refs -help

○ 外部 Manifest ファイル参照と署名付与（PKCS#12 ファイルの指定）
＞ Lx3Cmd **-sign -dmani manifest1.xml** -cert p12 LeTest.p12 test -out xades-mani1.xml

○ Manifest ファイル読み込み Enveloping 参照と署名付与（PKCS#12 ファイルの指定）
＞ Lx3Cmd **-sign -emani manifest1.xml** -cert p12 LeTest.p12 test -out xades-mani1.xml

▼ ヒント

- -det は -detdir / -dlist を利用して複数ファイルの一括指定も可能。
- -refs の-dmani は -dmlist を利用して複数 Manifest ファイルの一括指定も可能。
- -sign の-dmani / -emani は –dmlist / -emlist を利用して複数 Manifest ファイルの一括指定も可能。
- Manifest 要素を使った間接参照は参照段数を増やすことでトータルの参照数を増やすことが可能ですが、あまり数が多いと検証時に時間がかかるので注意が必要です。

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

５）署名情報取得（署名関連情報等の取得や署名レベルダウン等）

○ ファイル中の署名情報表示（複数の署名がある場合は全ての情報を出力）

```
＞ Lx3Cmd -info -list -in xades1.xml
```

○ ファイル中の署名レベル表示（複数の署名がある場合は全ての情報を出力）

```
＞ Lx3Cmd -info -level -in xades1.xml
```

○ 複数署名時の署名指定（-snum の番号で指定、省略時は全ての署名が対象）

◎ 複数署名時の指定は -verify や -time でも指定が可能。

```
＞ Lx3Cmd -info -level -snum 2 -in xades1.xml
```

○ 署名レベルダウン（BES 指定で XAdES-BES、T 指定で XAdES-T にレベルダウン）

◎ 高レベルの情報は削除されるので注意が必要です。

```
＞ Lx3Cmd -info -cut BES -in xades-a.xml -out xades-bes.xml
```

○ 署名情報取得のオプションヘルプ

```
＞ Lx3Cmd -info -help
```

６）その他機能（スキーマチェック）

○ XAdES/XmlDsig スキーマチェック（Lx3Cmd と同じ場所の xsds 下にスキーマファイルあり）

```
＞ Lx3Cmd -schema -in xades1.xml
```

○ スキーマチェックのオプションヘルプ

```
＞ Lx3Cmd -schema -help
```

※ 実例として sample/LeXAdES3/cmd の下にあるサンプルソースを参照。

ファイル	概要
CmdAll.bat CmdAll.sh	一括テスト実行
CmdXadesTest.bat CmdXadesTest.sh	XAdES 署名サンプル（XAdES-A-XL を生成）
CmdXadesPrefixTest.bat CmdXadesPrefixTest.sh	XAdES 署名サンプル（プレフィックス付き）
CmdDetachedIdTest.bat CmdDetachedIdTest.sh	Detached Id 指定サンプル
CmdEnvelopedTest.bat CmdEnvelopedTest.sh	Enveloped サンプル
CmdManifestTest.bat CmdManifestTest.sh	Manifest 指定サンプル
CmdClientTest.bat CmdClientTest.sh	クライアント署名サンプル

コマンド利用のサンプル

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

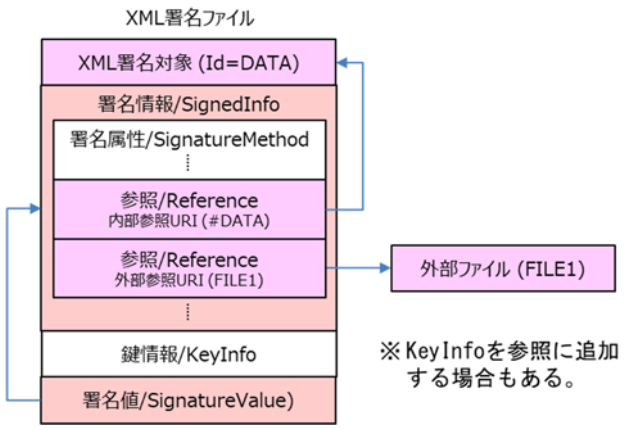
３．ＸＭＬ電子署名とＸＡｄＥＳ長期署名

３．１．XML 署名（XmlDsig）

XML 署名（XmlDsig）は W3C 勧告の仕様である。V1.0/V1.1/V2.0 が出ているが XAdES からも参照されている仕様は V1.1 となる。XML 署名の特長は実際の署名対象は Reference（参照）要素から URI 形式で複数の指定が可能である点である。署名値の対象となるのは Reference 群とそのハッシュ値を含む SignedInfo 要素である。署名鍵の情報は KeyInfo 要素に記載されるが証明書とは紐付いていない為に証明書置換攻撃の問題がある。これを防ぐには KeyInfo 要素に署名証明書を含み、KeyInfo を Reference する必要がある。

参照仕様：https://www.w3.org/TR/xmldsig-core1/
XML Signature Syntax and Processing Version 1.1
W3C Recommendation, 11 April 2013

０）XML 署名（XmlDsig）構造：署名のみ



- 左例は内部参照と外部参照が各 1 つのみの構造としている。
- SingedInfo を C14N 正規化したものが署名値の対象となる。
- Reference 先は Transform 指定がある場合は変換した上でハッシュ値を計算する。

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

３．２．XAdES 長期署名

XAdES はもともと XML Advanced Electronic Signatures の略称であったが、欧州の eIDAS にて Electronic Signatures は法的用語であり Digital Signatures であるとの判断から、XAdES は固有名詞となり正式には XAdES Digital Signatures となった。なお ISO 14533-2:2021 では過去の互換性から XML Advanced Electronic Signatures のままとなっている。

XAdES は XML 署名に XAdES 属性を追加した上位互換の仕様であり、XML 署名済みの署名ファイルを XAdES 長期署名に変更することはできない。XAdES 長期署名済みの署名ファイルは XML 署名としての検証が可能である。

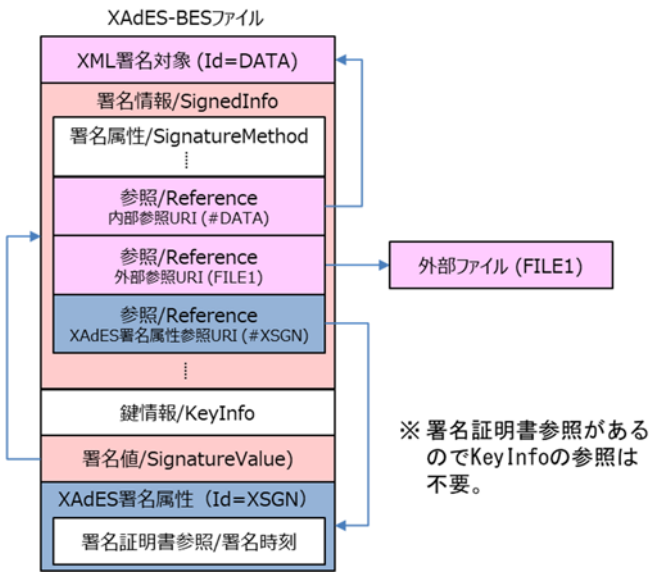
XAdES の署名属性（XAdES 属性のうち Reference され署名対象に含まれる属性部）として、署名証明書の参照情報（ハッシュ値等）を含んでいる為に証明書置換攻撃への耐性を持つ。また XAdES の非署名属性（XAdES 属性のうち Reference されず署名対象に含まれない属性部）として、署名タイムスタンプ・検証情報・アーカイブタイムスタンプを追加することが可能となる。特にタイムスタンプは XML 署名では付けることが仕様的にできない。このように XML 署名の問題を解決し新しい機能が使えるようになったことで、XAdES は長期署名では無く「先進署名」とも呼ばれている。

署名のみでも XML 署名では無く XAdES-BES 署名とするべき理由がこの先進性にある。

参照仕様：https://www.iso.org/standard/79129.html

ISO 14533-2:2021 Profiles for XML Advanced Electronic Signatures (XAdES)

１）XAdES-BES 構造：署名のみ

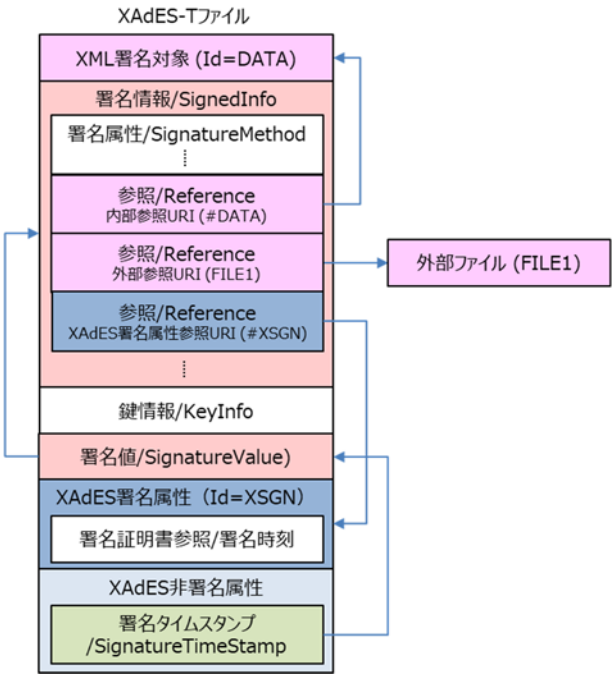


- 左例は内部参照と外部参照が各 1 つのみの構造としている。
- SingedInfo を C14N 正規化したものが署名値の対象となる。
- Reference 先は Transform 指定がある場合は変換した上でハッシュ値を計算する。
- **Reference として XAdES の署名属性を追加する必要がある。後から Reference 追加は出来ないので XML 署名済みのファイルを後から XAdES-BES にすることはできない。**

XAdES-BES は署名のみであり XML 署名とほぼ同じと言えるが、署名証明書が署名時に必要となる。署名時に署名証明書を取得できる必要があるので、特に IC カードの利用時等には署名手順を良く考える必要がある。

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

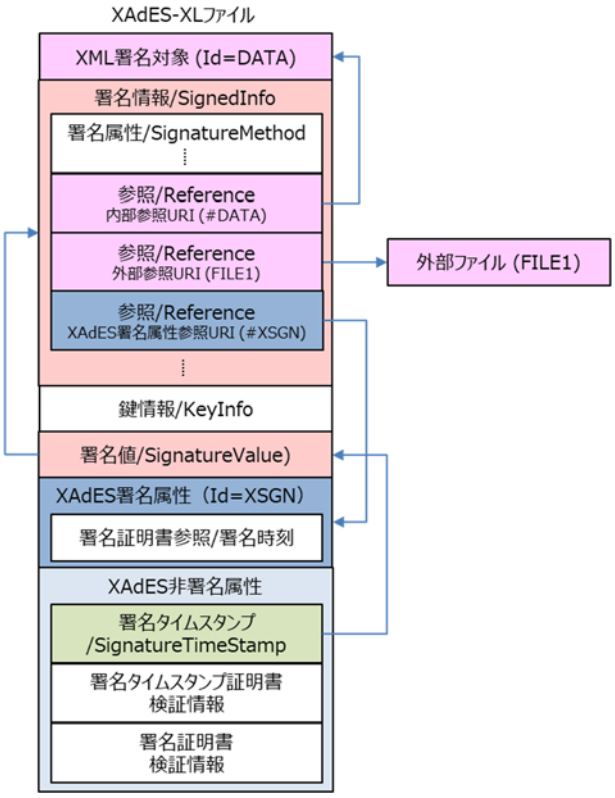
2）XAdES-T の構造：XAdES-BES ＋ 署名タイムスタンプ



- 左例は内部参照と外部参照が各 1 つのみの構造としている。
- SingedInfo を C14N 正規化したものが署名値の対象となる。
- Reference 先は Transform 指定がある場合は変換した上でハッシュ値を計算する。
- Reference として XAdES の署名属性を追加する必要がある。後から Reference 追加は出来ないので XML 署名済みのファイルを後から XAdES-BES にすることはできない。
- **署名タイムスタンプを非署名属性領域に追加。**
- **署名タイムスタンプは署名値 SignatureValue を参照している。**

署名タイムスタンプは署名値を参照しており追加時に Reference 要素等は不要。

3）XAdES-XL の構造：XAdES-T ＋ 検証情報

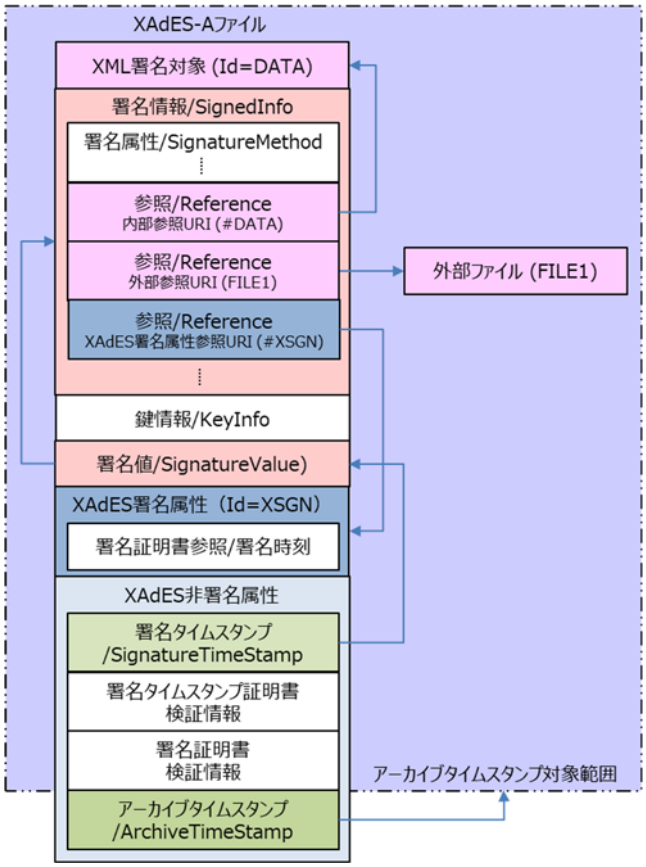


- 左例は内部参照と外部参照が各 1 つのみの構造としている。
- SingedInfo を C14N 正規化したものが署名値の対象となる。
- Reference 先は Transform 指定がある場合は変換した上でハッシュ値を計算する。
- Reference として XAdES の署名属性を追加する必要がある。後から Reference 追加は出来ないので XML 署名済みのファイルを後から XAdES-BES にすることはできない。
- 署名タイムスタンプを非署名属性領域に追加。
- 署名タイムスタンプは署名値 SignatureValue を参照している。
- **署名証明書と署名タイムスタンプ証明書の検証情報を非署名属性領域に追加。**

検証情報として、署名証明書と署名タイムスタンプの TSA 証明書の 2 つの認証パスの検証情報が必要となる。検証情報は非署名属性部にある為に保護されていない。

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

4）XAdES-A の構造：XAdES-XL ＋ アーカイブタイムスタンプ



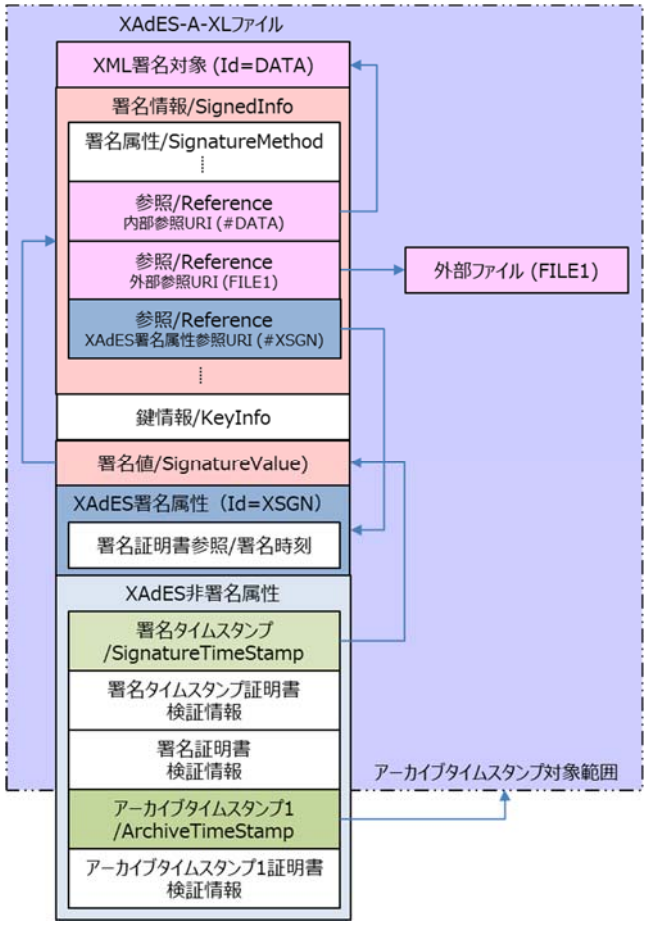
- 左例は内部参照と外部参照が各 1 つのみの構造としている。
- SingedInfo を C14N 正規化したものが署名値の対象となる。
- Reference 先は Transform 指定がある場合は変換した上でハッシュ値を計算する。
- Reference として XAdES の署名属性を追加する必要がある。後から Reference 追加は出来ないので XML 署名済みのファイルを後から XAdES-BES にすることはできない。
- 署名タイムスタンプを非署名属性領域に追加。
- 署名タイムスタンプは署名値 SignatureValue を参照している。
- 署名証明書と署名タイムスタンプ証明書の検証情報を非署名属性領域に追加。
- **アーカイブタイムスタンプを非署名属性領域に追加。**
- **アーカイブタイムスタンプは外部ファイルも含めてそれ以前の全てを参照している。**

XAdES-A ではアーカイブタイムスタンプのハッシュ値を計算する為に全 Reference 先を含むそこまでの全ての情報を必要とする。この為に外部参照をしている場合には追加時にその参照先ファイルも必要となるので注意が必要となる。なお Reference 先に Manifest（間接指定）要素を指定した場合には、Manifest 要素から Reference（参照）している情報はアーカイブタイムスタンプでは参照しない。

アーカイブタイムスタンプには V1.3.2 と V1.4.1 の 2 つの仕様がある。この 2 つはほぼ同じ内容ではあるが、未仕様の Object 要素の扱い等が異なり V1.4.1 の方がより安全な仕様となっている。この為に新規に追加する場合には V1.4.1 のアーカイブタイムスタンプを利用する。V1.3.2 のアーカイブタイムスタンプは過去互換性の為にある。欧州では V1.3.2 のアーカイブタイムスタンプの新規利用は認められていない。

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

５）XAdES-A-XL：XAdES-A ＋ 検証情報



- 左例は内部参照と外部参照が各１つのみの構造としている。
- SingedInfo を C14N 正規化したものが署名値の対象となる。
- Reference 先は Transform 指定がある場合は変換した上でハッシュ値を計算する。
- Reference として XAdES の署名属性を追加する必要がある。後から Reference 追加は出来ないので XML 署名済みのファイルを後から XAdES-BES にすることはできない。
- 署名タイムスタンプを非署名属性領域に追加。
- 署名タイムスタンプは署名値 SignatureValue を参照している。
- 署名証明書と署名タイムスタンプ証明書の検証情報を非署名属性領域に追加。
- アーカイブタイムスタンプを非署名属性領域に追加。
- アーカイブタイムスタンプは外部ファイルも含めてそれ以前の全てを参照している。
- **アーカイブタイムスタンプ証明書の検証情報を非署名属性領域に追加。**

追加の検証情報として、アーカイブタイムスタンプの TSA 証明書の認証パスの検証情報が必要となる。XAdES-A-XL 化した後に、次のアーカイブタイムスタンプを追加することができる。これにより電子署名の有効期間を延長することができる。

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

XAdES のレベルは LeXAdES3.getLevel() にて LX3_XADES_LEVEL として取得が可能となっている。Lx3Cmd では -info -level で取得ができる。

値	LX3_XADES_LEVEL 定義	説明
-1	XL_UNKNOWN	不定またはエラー
0	XL_NOSIGN	署名無し
1	XL_DSIG	XML 署名（W3C XmlDsig）
9	XL_MAKE	XAdES-MAKE / XAdES-M 署名基本未署名（仮署名状態）
10	XL_BES	XAdES-BES / XAdES-B 署名基本（EPES:ポリシー付含む）
11	XL_T	XAdES-T / XAdES-T 署名タイムスタンプ
12	XL_XL	XAdES-XL（XAdES-X-Long）/ XAdES-LT 検証情報付
13	XL_A	XAdES-A / XAdES-LTA アーカイブタイムスタンプ ※
14	XL_AXL	XAdES-A-XL / XAdES-LTA アーカイブタイムスタンプ ※（検証情報付）

※ 1署名にて複数のアーカイブスタンプがある時には、最後のアーカイブタイムスタンプに検証情報（TimeStampValidationData）が、無ければ XL_A に、あれば XL_AXL を返す。

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

３．３．XAdES オプション要素

LE:XAdES:Lib の V1/V2 では 1 つの署名しか対応していなかったが、LE:XAdES:Lib V3 からは複数署名に対応した。XML 署名/XAdES の場合に複数署名と言えば同一の XML ファイル中に複数の Signature 要素を持つ形式と言える。つまり内部 Detached（Id 指定）か Enveloped のようにベースとなる XML ファイルに署名を並列または平行に追加する形式となる。XAdES 仕様には 1 つの Signature 要素の中にカウンター署名の CounterSignature 要素を追加してシリアル署名を実現する形式もあるがあまりニーズが無い為に LE:XAdES:Lib V3 ではカウンター署名には未対応となる。

LE:XAdES:Lib V3 では、並列署名・平行署名時の署名（Signature 要素）追加場所は XPath にて指定が可能となっている。LE:XAdES:Lib V3 では、検証時や長期署名化する場合の署名は解析時に順番に付けられる署名番号でどの署名に対する操作かを指定可能となっている。

１）SignedSignatureProperties の要素（署名対象）

SignedSignatureProperties 下の要素は 1 署名つき 1 つ指定が可能となっている。SigningTime・SigningCertificateV2 以外に、SignaturePolicyIdentifier・SignatureProductionPlace・SignerRole 要素があるが、ほとんど使われていない為に LE:XAdES:Lib V3 では未サポートとなっている。（※必要な場合にはご相談ください。）

1-1）署名時刻：SigningTime

SigningTime は署名時刻となるが非 RFC 3161 タイムスタンプの署名システムのシステム時刻となる。XAdES-T 以上では検証時に署名タイムスタンプ時刻を利用するが、XAdES-BES では署名タイムスタンプが無い為に標準では検証時刻にて検証されるが、欧州の EN 仕様では SigningTime 時刻を検証にも使って良いことになっている。また欧州の EN 仕様では SigningTime は必須要素となっているが、ISO 14533-2 では任意要素となっている。欧州との互換性を考えると SigningTime 要素は付けておいた方が良くデフォルト設定では SigningTime 要素が追加される。追加したく無い時には LeXAdES3.sign() 時に署名フラグとして LX3_SIGN_FLAG::XS_NO_SIGN_TIME オプションを指定する。Lx3Cmd コマンドでは署名時に -xopt nostime を指定する。

XAdES-BES 検証時において SigningTime 時刻を検証時刻として利用するには、API では LeXAdES3.verify() 時に検証フラグとして LX3_VERIFY_FLAG:: XV_USE_SIGN_TIME オプションを指定する。Lx3Cmd コマンドでは検証時に -use stime を指定する。

1-2）署名証明書参照：SigningCertrificateV2 / SigningCertrificate（必須要素）

XAdES v1.3.2 では署名証明書を確認する署名対象情報として SiginingCertificate（V1）が使われていた。SiginingCertificate では署名証明書に関して、拇印（ハッシュ値）・証明書発行者名・シリアル番号が利用されるが、このうち証明書発行者名（Issuer Name）は RFC 2253 で指定されたテキスト形式にすることになっている。署名証明書においてこのテキスト情報との一致をする必要があるが、テキスト形式が RFC 2253 に準拠していない XAdES 実装が多く、比較が困難であると言う問題

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

があった。例えば「CN=LE Root,O=LangEdge,C=JP」が正しい場合に以下のようなケースが散見されている。つまりこれらのケースに対応した正規化が必要になると言うことになる。

C=JP,O=LangEdge,CN=LE Root	順番が逆順になっている
CN=LE Root, O=LangEdge, C=JP	一見正しく見えるがカンマの後に空白文字がある
cn=LE Root,o=LangEdge,c=JP	属性が小文字になっている

RFC 2253 違反の証明書発行者名の例

この為に署名者発行者名を X.509 電子証明書の Issuer Name からバイナリのままシリアル番号と共に Base64 化して格納する方式にした仕様が SigningCertificateV2 として標準化された。以下に同じ署名証明書に対する SigningCertificateV2 と SigningCertificate の例を示す。

```
<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#">
        nxlX1m9trpAd7WS23y8EvLnONZiTFh+kJBZSFGr5vx4=
      </DigestValue>
    </CertDigest>
    <IssuerSerialV2>
      MFkwUqRQME4xCzAJBgNVBAYTAkpQMREwDwYDVQQKDAhMYW5nRWRnZTENMAsGA1UECwwEZGVtbzEdMBsGA1UEAw
      wUTGFuZOVkZ2UgQOEgUm9vdCAwMDECAxAAFw==
    </IssuerSerialV2>
  </Cert>
</SigningCertificateV2>
```

SigningCertificateV2 の例

```
<SigningCertificate>
  <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#">
        nxlX1m9trpAd7WS23y8EvLnONZiTFh+kJBZSFGr5vx4=
      </DigestValue>
    </CertDigest>
    <IssuerSerial>
      <X509IssuerName>CN=LangEdge CA Root 001,OU=demo,O=LangEdge,C=JP</X509IssuerName>
      <X509SerialNumber>1048599</X509SerialNumber>
    </IssuerSerial>
  </Cert>
</SigningCertificate>
```

SigningCertificate の例

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

```
<SEQUENCE>
  <SEQUENCE>
    <Context-Specific order="4">
      <SEQUENCE>
        <SET>
          <SEQUENCE>
            <OBJECT_IDENTIFIER conv="3">2.5.4.6</OBJECT_IDENTIFIER>
            <PrintableString conv="2">JP</PrintableString>
          </SEQUENCE>
        </SET>
        <SET>
          <SEQUENCE>
            <OBJECT_IDENTIFIER conv="3">2.5.4.10</OBJECT_IDENTIFIER>
            <UTF8String conv="8">LangEdge</UTF8String>
          </SEQUENCE>
        </SET>
        <SET>
          <SEQUENCE>
            <OBJECT_IDENTIFIER conv="3">2.5.4.11</OBJECT_IDENTIFIER>
            <UTF8String conv="4">demo</UTF8String>
          </SEQUENCE>
        </SET>
        <SET>
          <SEQUENCE>
            <OBJECT_IDENTIFIER conv="3">2.5.4.3</OBJECT_IDENTIFIER>
            <UTF8String conv="20">LangEdge CA Root 001</UTF8String>
          </SEQUENCE>
        </SET>
      </SEQUENCE>
    </Context-Specific>
  </SEQUENCE>
  <INTEGER>100017</INTEGER>
</SEQUENCE>
```

IssuerSerialV2 に格納されている DER 情報例（LeBerXml テキスト出力形式）

要素名	SigningCerficate（急）	SigningCertificateV2（新）
拇印（ハッシュ値）	Cert/CertDigest	Cert/CertDigest（同左）
証明書発行者名	Cert/IssuerSerial/ X509IssuerName の下に Issuer の RFC 2253 形式情報	Cert/IssuerSerialV2 の下に Issuer+Serial のバイナリを Base64 化した情報
シリアル番号	Cert/IssuerSerial/ X509SerialNumber の下に Serial の BIG-INTEGER 情報	

SigningCertificate と SigningCertificateV2 の比較

XAdES 署名時において SigningCertificate を利用するには、API では署名時に署名フラグとして LX3_SIGN_FLAG::XS_NO_SIGN_CERT2 オプションを指定する。Lx3Cmd コマンドでは署名時に -xopt nocert2 を指定する。

なお LE:XAdES:Lib V3 では、XAdES 検証時に SigningCertificate が使われていた場合には証明書発行者名はチェックしておらず拇印のみチェックしている。

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

２）SignedDataObjectProperties/DataObjectFormat 要素（署名対象）

SignedDataObjectProperties 下の DataObjectFormat 要素は Reference 参照単位で指定が可能となっている。DataObjectFormat/ObjectIdentifier 要素と、CommitmentTypeIndication・AllDataObjectsTimeStamp・IndividualDataObjectsTimeStamp 要素は、ほとんど使われていない為に LE:XAdES:Lib V3 では未サポートとなっている。（※ 必要な場合にはご相談ください。）

LE:XAdES:Lib V3 では DataObjectFormat 下の要素の指定としては、MimeType・Encoding・Description の 3 種類の要素が LeReference クラスの参照先指時の引数として定可能となっている。このうち MimeType 要素は、欧州の EN 仕様では MimeType は必須要素となっているが ISO 14533-2 では任意要素となっている。DataObjectFormat 要素を追加するのであれば MimeType は利用すべき要素と言える。欧州との互換性を考えると MimeType 要素は付けておいた方が良いが、日本ではあまり利用されていない為にデフォルトではオフ（MimeType を使わない）設定となっている。

DataObjectFormat を追加するには MimeType・Encoding・Description の指定以外に、API では署名時の署名フラグ LX3_SIGN_FLAG::XS_USE_DOFORM オプションを指定し、Lx3Cmd では -opt usedof を指定する必要がある。

要素名	概要	説明
MimeType	メディア種別	ex) "text/plain" 欧州 EN 仕様では必須要素だが ISO 仕様ではオプション要素 Lx3Cmd では参照先指定時に mime:<MimeType> と指定可
Encoding	エンコード方法	ex) "http://www.ietf.org/rfc/rfc2279.txt" ← UTF-8 欧州 EN 仕様と ISO 仕様の共にオプション要素 Lx3Cmd では参照先指定時に encd:<Encoding> と指定可
Description	参照先の説明	任意の文字列（UTF-8）を指定可能。署名理由や場所等に利用。 欧州 EN 仕様と ISO 仕様の共にオプション要素 Lx3Cmd では参照先指定時に desc:<Description> と指定可

SignedDataObjectProperties/DataObjectFormat 下の要素

※ DataObjectFormat を追加する Lx3Cmd 例

```
> Lx3Cmd -sign -det test.txt "mime:text/plain" "encd:http://www.ietf.org/rfc/rfc2279.txt" ¥
    "desc:This is test." -xopt usedof -cert p12 LeTest.p12 test -out xades.xml
```

```
<SignedDataObjectProperties>
  <DataObjectFormat ObjectReference="#LxRF-5">
    <MimeType>text/plain</MimeType>
    <Encoding>http://www.ietf.org/rfc/rfc2279.txt</Encoding>
    <Description>This is test.</Description>
  </DataObjectFormat>
</SignedDataObjectProperties>
```

SignedDataObjectProperties/DataObjectFormat の例

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

３．４．Manifest 要素の指定

XML 署名(XmlDsig)には Manifest 要素により参照を間接的に行う仕様が含まれている。Manifest 要素の下には Reference 要素のみが許されている。SignedInfo の Reference から Manifest 要素またはファイルを参照し、その Manifest の Reference からまた別のファイルを参照できる。Manifest を Referenceする場合はType="http://www.w3.org/2000/09/xmldsig#Manifest" を指定する必要がある。

参照仕様：https://www.w3.org/TR/xmldsig-core1/#sec-Manifest
XML Signature Syntax and Processing Version 1.1
5.1 The Manifest Element

```
<Manifest xmlns="http://www.w3.org/2000/09/xmldsig#">
  <Reference URI="test.txt">
    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
    <DigestValue>PnRLnc0TibrwxaBmBYm4QC89u0m4mz518sk1WFKjxnc=</DigestValue>
  </Reference>
  <Reference URI="image.png">
    <DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
    <DigestValue>aYSX7keBZVQiknm0H0E7BHTvxg1EbYKRSR9x2ICrXnw=</DigestValue>
  </Reference>
  <Reference URI="data.xml">
    <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>S6QnH1pM1ouDk8y2FQbtBtch+VAlISNkTxDxsXAgm/o=</DigestValue>
  </Reference>
</Manifest>
```

SHA-256 の Reference を 3 つ持つ Manifest 要素の例

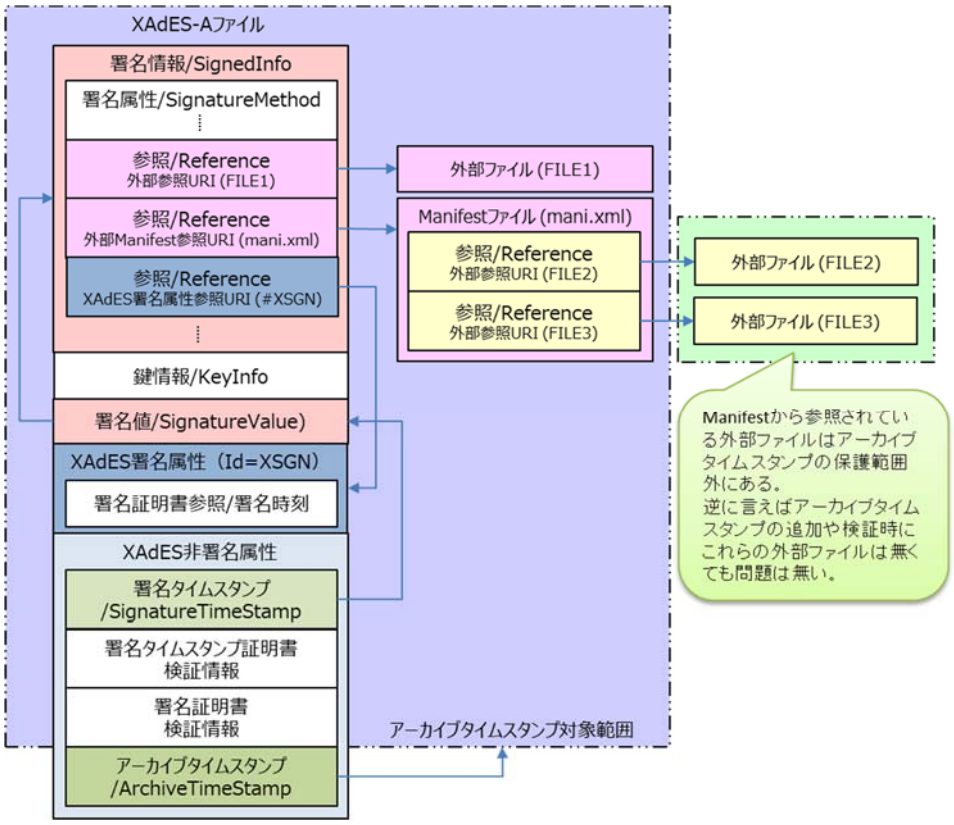
XAdES-A のアーカイブタイムスタンプのハッシュ値計算は、SignedInfo から Reference されている対象までは含まれるが、SignedInfo から Manifest 経由で Reference されるデータまでは含まれない（次ページの構造を参照）。これは Manifest から参照された対象は長期署名の保証外となることを示している。もし Manifest の中の Reference ハッシュ方式が危殆化した場合にも保護されない。
XAdES V1.4.1 には RenewedDigestsV2 要素にて再ハッシュ計算をする手段が提供されているが、LE:XAdES:Lib V3 では RenewedDigestsV2 要素は未サポートであり将来サポート予定となっている。

XAdES-A のアーカイブタイムスタンプのハッシュ値計算に Manifest からの Reference 参照が含まれないことは、大量の Reference 参照をしたい場合に長期署名検証と各参照要素のチェックを分けることができる利点にもなる。大量の Reference 参照のうち 1 つだけを検証したい場合に、まず長期署名検証を行い、長期署名検証が正常の場合にその 1 つだけのハッシュ値を確認することができる。

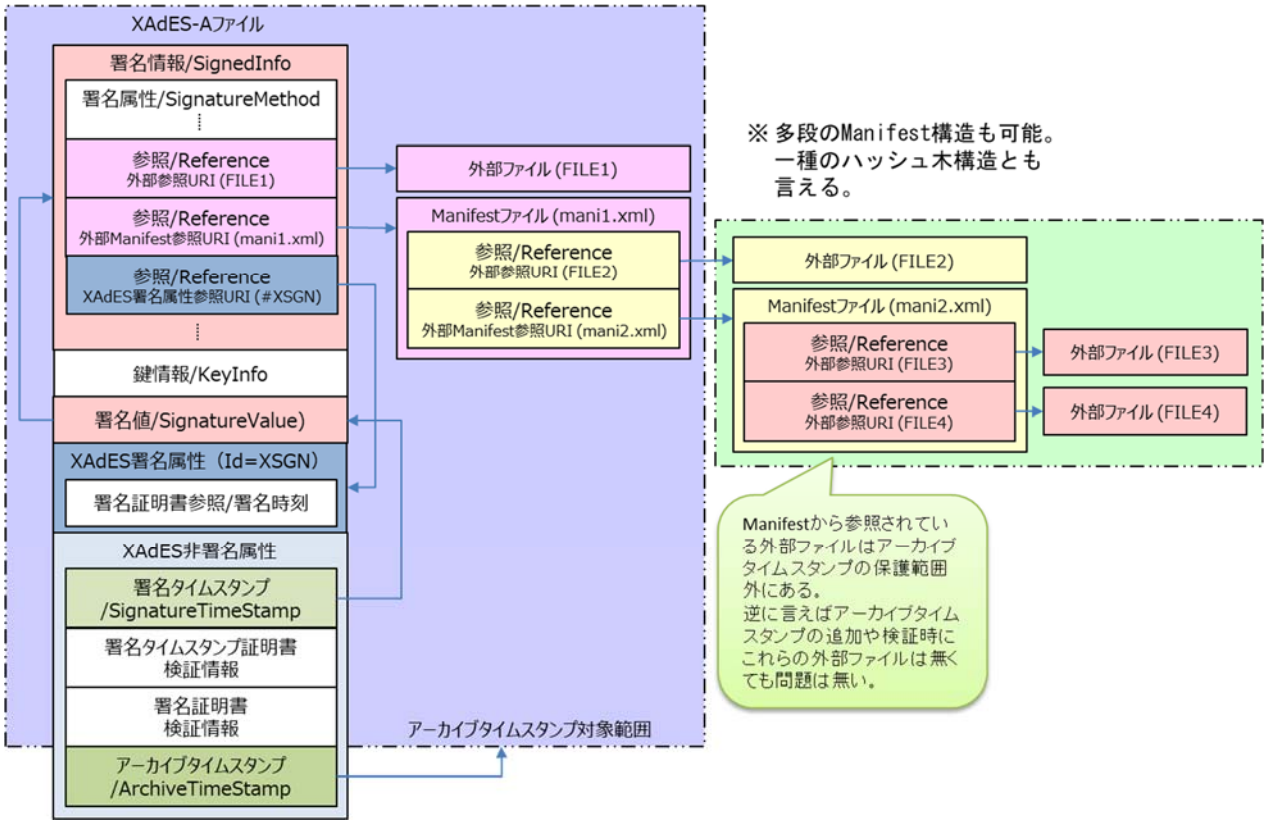
Manifest 要素の利用は注意する点もあるが、大量の署名対象を 1 つの署名とタイムスタンプで保護できる有用な仕様と言える。問題点と利点を理解した上で利用の検討をすべきである。

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

５）外部 Manifest 利用 XAdES-A 構造



６）多重外部 Manifest 利用 XAdES-A 構造

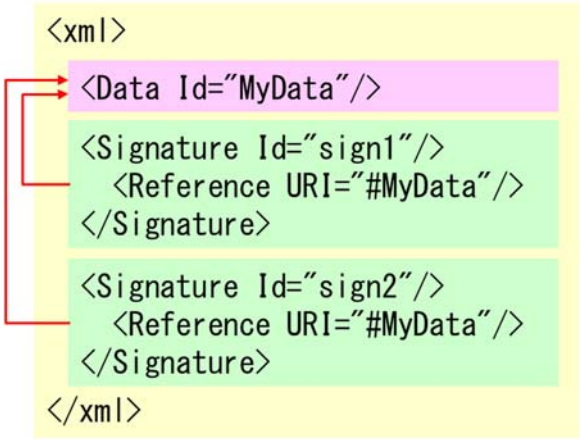
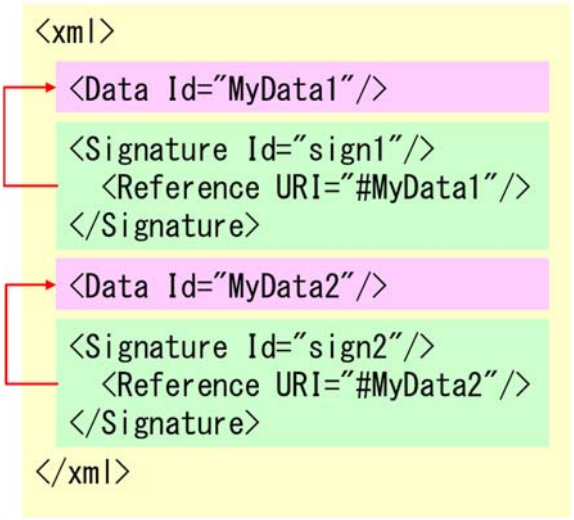


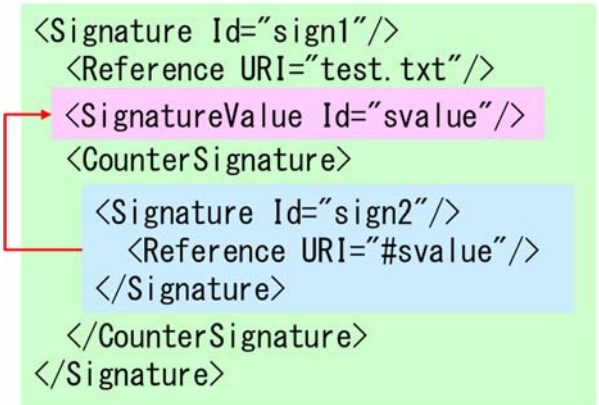
Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

３．５．複数署名

LE:XAdES:Lib の V1/V2 では 1 つの署名しか対応していなかったが、LE:XAdES:Lib V3 からは複数署名に対応した。XML 署名/XAdES の場合に複数署名と言えば同一の XML ファイル中に複数の Signature 要素を持つ形式と言える。つまり内部 Detached（Id 指定）か Enveloped のようにベースとなる XML ファイルに署名を並列または平行に追加する形式となる。XAdES 仕様には 1 つの Signature 要素の中にカウンター署名の CounterSignature 要素を追加してシリアル署名を実現する形式もあるがあまりニーズが無い為に LE:XAdES:Lib V3 ではカウンター署名には未対応となる。

LE:XAdES:Lib V3 では、並列署名・平行署名時の署名（Signature 要素）追加場所は XPath にて指定が可能となっている。LE:XAdES:Lib V3 では、検証時や長期署名化する場合の署名は解析時に順番に付けられる署名番号でどの署名に対する操作かを指定可能となっている。

並列署名：同一対象に複数の署名	平行署名：異なる対象に異なる署名
	
※ Enveloped か同じ Id 指定の内部 Detached	※ 異なる Id 指定の内部 Detached

直列署名：署名値に対するカウンター署名

※ 正確には XAdES の非署名属性部に追加 ※ LE:XAdES:Lib V3 では非サポート

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

複数署名の確認：Lx3Cmd の -info -list 利用

```
> Lx3Cmd -info -list -in eped-bes2.xml
[snum:0] Id='LxSig-1', XPath='/LeXMLData/Signature'
[snum:1] Id='LxSig-2', XPath='/LeXMLData/Signature[2]'
>
```

上例の意味（署名番号は 0 から始まる）

署名番号	Id	署名位置/XPath	説明
snum:0	LxSig-1	/LeXMLData/Signature	1 番目の署名、snum=0 で操作が可能
snum:1	LxSig-2	/LeXMLData/Signature[2]	2 番目の署名、snum=1 で操作が可能

署名番号を指定した検証：2 番目の署名だけを検証

```
> Lx3Cmd -verify -snum 1 -report -in eped-bes2.xml
LE:XAdES3:Lib Signature Verify Result Report V3.00.R1

署名有効性検証: 有効:VALID - 署名や参照先は改ざんされておらず証明書も有効
 署名[1]: 検証(有効:VALID) パス(/LeXMLData/Signature[2]) Id(LxSig-2) 形式(10:XAdES-BES)
  署名参照検証: 検証(有効:VALID) Id(LxSI-3)
   署名値: 検証(有効:VALID) 暗号(RSA/SHA256) 正規化(INC) 鍵(cert)
   参照[0]: 検証(有効:VALID) URI() 種別(EPED) ハッシュ(SHA256) 変換(EPED,INC)
   参照[1]: 検証(有効:VALID) URI(#LxXS-8) 種別(XOBJ) ハッシュ(SHA256) 変換(INC)
  署名証明書検証: 検証(有効:VALID) 時刻(20230817220626+0900)
   証明書: 検証(有効:VALID) 取得(keyInfo) 検証(CRL)
           名前(LE Test2 100017)
           番号(100017) 期限(20300101090000+0900)
    CRL: 取得(network) 更新(20230817042101+0900)
   証明書: 検証(有効:VALID) 取得(orgStore)
           名前(LangEdge CA Root 001)
           番号(038D7EA4C68005) 期限(20430128191838+0900)
  長期署名検証: 検証(有効:VALID) Id(LxXS-8) 署名時(20230809220144+0900)

>
```

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

３．６．署名検証

署名がある場合には、検証結果に有効・不明・無効の 3 つの状態がある。複数の署名がある場合には、全て VALID なら VALID に、1 つでも INDETERMINATE があり INVALID が無ければ INDETERMINATE に、1 つでも INVALID があれば INVALID になる。

状態	説明	補足
LPK_VS_NONE	署名無し	---
LPK_VS_VALID	有効	検証項目が全て有効であった
LPK_VS_INDETERMINATE	不明（検証情報の不足）	改ざんは検出されないが、CRL/OCSP や認証パスに必要な証明書が足りないか、ルート証明書の信頼性が確認できない等
LPK_VS_INVALID	無効（失効や改ざん等）	1 つでも無効または異常があった

検証結果の状態（LPK_VALID_STATUS）

1 つの署名に付き必ず必要な検証項目は「署名検証 / Signature verification」と「証明書検証 / Certificate validation」になる。XAdES-T 以上の署名レベルであれば更に「長期署名検証 / Long-term verifcation」が必要となり、タイムスタンプ毎にタイムスタンプトークン検証と TSA 証明書の証明書検証を行う。全てを合わせて「署名有効性検証 / Signature validation」と呼ぶ。

検証項目		説明
署名参照検証		署名値と参照している署名対象を確認
	署名値の確認	署名対象に対して署名証明書の公開鍵で検証をして改ざん有無を確認
	参照先の確認	Reference されている対象のハッシュ値を計算して改ざん有無を確認
	必須項目の確認	仕様に定められている必須の属性が含まれている事を確認 仕様で禁止されている属性が含まれていない事を確認
署名証明書検証		検証情報（証明書/CRL/OCSP）を集めて指定時刻で検証を行う
	認証パスの構築	ルート証明書（自己署名証明書）まで認証パスを構築できる事を確認
	認証パスの失効確認	認証パスに含まれる全ての証明書の失効を確認 ※ 最後のルート証明書は通常失効確認はされない
	トラストアンカー確認	認証パス最後のルート証明書が信頼済みのトラストアンカーか確認 ※ 独自証明書ストアの trusts にあるか、Windows 証明書ストアで確認
長期署名検証		タイムスタンプと検証情報を確認
	署名タイムスタンプ検証	タイムスタンプトークンのハッシュ値を確認する タイムスタンプの TSA 証明書の証明書検証を行う
	アーカイブタイムスタンプ検証（複数ある場合あり）	タイムスタンプトークンのハッシュ値を確認する タイムスタンプの TSA 証明書の証明書検証を行う

検証項目（検証結果レポートされる項目）

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

証明書検証では失効の有無を、署名時刻（署名タイムスタンプ時刻か署名後最初のタイムスタンプ時刻）か、署名時刻が確認できない場合には現在時刻において確認する。現時点で失効していても署名時刻において有効であれば問題なく有効である。

CRL/OCSP には署名が付いているがルート証明書の移行があった場合には、署名証明書のルート証明書と、CRL/OCSP 署名の署名証明書のルート証明書が異なる場合がある（通常は一致する）。LE:XAdES:Lib V3 では、両方のルート証明書がトラストアンカーと信頼されておりかつ発行者名が同じ場合にはエラーにならない仕様となっている。発行者名が異なるかトラストアンカーとして確認できない場合には不明（INDETERMINATE）となる。

XAdES V3 の長期保管の一般的な形式は XAdES-A となる。検証結果の例を以下に示す。

```
LE:XAdES3:Lib Signature Verify Result Report V3.00.R1

署名有効性検証: 有効:VALID - 署名や参照先は改ざんされておらず証明書も有効
 署名[0]: 検証(有効:VALID) パス(/Signature) Id(LxSig-1) 形式(14:XAdES-A-XL)
  署名参照検証: 検証(有効:VALID) Id(LxSI-2)
   署名値: 検証(有効:VALID) 暗号(RSA/SHA256) 正規化(INC) 鍵(cert)
   参照[0]: 検証(有効:VALID) URI(#LxOE-6) 種別(EPING) ハッシュ(SHA256) 変換(B64)
   参照[1]: 検証(有効:VALID) URI(image.png) 種別(DOUT) ハッシュ(SHA256)
   参照[2]: 検証(有効:VALID) URI(#LxXS-9) 種別(XOBJ) ハッシュ(SHA256) 変換(INC)
  署名証明書検証: 検証(有効:VALID) 時刻(20230816220803+0900)
   証明書: 検証(有効:VALID) 取得(keyInfo) 検証(CRL)
           名前(LE Test2 100017)
           番号(100017) 期限(20300101090000+0900)
    CRL: 取得(sigValid) 更新(20230816034801+0900)
   証明書: 検証(有効:VALID) 取得(sigValid)
           名前(LangEdge CA Root 001)
           番号(038D7EA4C68005) 期限(20430128191838+0900)
  長期署名検証: 検証(有効:VALID) Id(LxXS-9) 署名時(20230816220803+0900)
   署名 TS: 検証(有効:VALID) Id(LxST-1)
    TS トークン: 検証(有効:VALID) TS 時刻(20230816220803+0900) ハッシュ(SHA512) 正規化(INC)
    TSA 証明書検証: 検証(有効:VALID) 時刻(20230816220803+0900)
     証明書: 検証(有効:VALID) 取得(data) 検証(CRL)
             名前(LE TSA 200030)
             番号(200030) 期限(20300101090000+0900)
      CRL: 取得(stsValid) 更新(20230816034801+0900)
     証明書: 検証(有効:VALID) 取得(tstData)
             名前(LangEdge CA Root 002)
             番号(038D7EA4C68006) 期限(20430128192003+0900)
   保管 TS[0]: 検証(有効:VALID) Id(LxAT-1)
    TS トークン: 検証(有効:VALID) TS 時刻(20230816220803+0900) ハッシュ(SHA512)
    TSA 証明書検証: 検証(有効:VALID) 時刻(20230816220804+0900)
     証明書: 検証(有効:VALID) 取得(data) 検証(CRL)
             名前(LE TSA 200030)
             番号(200030) 期限(20300101090000+0900)
      CRL: 取得(atsValid) 更新(20230816034801+0900)
     証明書: 検証(有効:VALID) 取得(tstData)
             名前(LangEdge CA Root 002)
             番号(038D7EA4C68006) 期限(20430128192003+0900)
```

XAdES-A の検証結果例

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

検証情報の取得先の一覧を以下に示す。検証結果では PKI 要素はどこから取得したかを以下の表示文字列で示している。

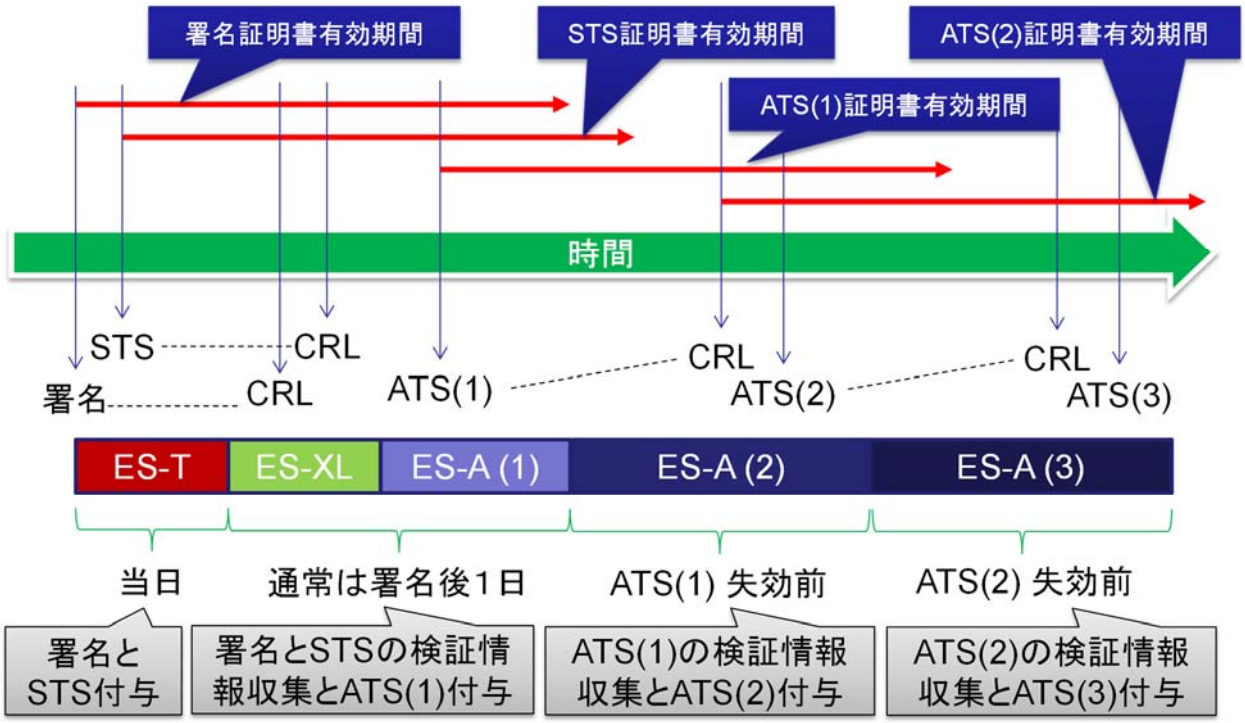
LPK_INFO_FROM 定義	表示文字列	説明
LPK_FM_UNKNOWN	unknown	取得先不明（エラー）
LPK_FM_DATA	data	署名データから取得
LPK_FM_ORGSTORE	orgStore	独自証明書ストアから取得
LPK_FM_ADDSTORE	addStore	追加した検証情報から取得、CVS 取得情報等
LPK_FM_WINSTORE	winStore	Windows 証明書ストアから取得
LPK_FM_NETWORK	network	ネットワークから取得
LPK_FM_VALID	valid	検証データ（OCSP/CRL 等）から取得
LPK_FM_PDF_DSS	pdfDss	PAdES：PDF の DSS 辞書から取得
LPK_FM_NOTUSE	notUse	未利用（ネットワーク接続しない時等）
LPK_FM_LCACHE	cache	キャッシュから取得
LPK_FM_TS_DATA	tstData	XAdES：TimeStampToken から取得
LPK_FM_KEYINFO	keyInfo	XAdES：KeyInfo から取得
LPK_FM_SIGN_VALID	sigValid	XAdES：CertificateValues/RevocationValues から取得
LPK_FM_SIGTS_VALID	stsValid	XAdES：署名タイムスタンプの TimeStampValidationData から取得
LPK_FM_ARCTS_VALID	atsValid	XAdES：アーカイブタイムスタンプの TimeStampValidationData から取得

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

３．７．長期署名の運用

XAdES では、XAdES-BES → XAdES-T → XAdES-X Long（XAdES-XL） → XAdES-A → XAdES-A-XL と情報を追加して行く。XAdES-A-XL の後は XAdES-A → XAdES-A-XL を繰り返して保証期間を延長して行く。XAdES-XL と XAdES-A-XL では検証情報を追加する。検証情報のうち失効確認の CRL と OCSP に関しては運用について注意が必要となる。CRL の場合には最新情報を取得するには猶予期間を考慮する必要がある。OCSP の場合には通常であれば最新の結果がリアルタイムに取得できる為に猶予期間を考慮する必要はない。CRL と OCSP の優先について詳しくは「LE:PKI:Lib マニュアル」（LePKI-manual.pdf）の「３．１１．失効情報取得の高度な指定」を参照。

なお CRL を利用したとしても署名後すぐに取得する運用をしても通常であれば問題は無い。これは通常１日を争うような状況は無いからである。もし厳密に１日を争うような必要があれば猶予期間を設けた方が良いと言うことになるので、どちらにするかの判断は必要となる。



猶予期間を考慮した長期署名生成のタイミング例（CRL が毎日更新される場合）

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

３．８．XML スキーマチェック

XML スキーマ（Schema）とは、XML データの構造を定義する為の言語の１つである。XML 署名（XmlDsig）も XAdES も構造をチェックする為のスキーマファイルが提供されている。
LE:XAdES:Lib V3 では XmlDsig/XAdES のスキーマファイルを bin_linux や bin_win（の下の Release/Release64）の下の xsds ディレクトリ下に置いている。XAdES01903v141-201601.xsd はローカルの XAdES01903v132-201601.xsd を参照するように元のファイルを修正してある（オンライン参照がうまく動作しない為）。

No	xsds 下のファイル名	説明
1	XAdES01903v141-201601.xsd	ETSI XAdES V1.4.1 用のスキーマファイル ※ XAdES01903v132-201601.xsd を参照するように修正 ※ xmldsig-core-schema.xsd をオンライン参照
2	XAdES01903v132-201601.xsd	ETSI XAdES V1.3.2 用のスキーマファイル ※ xmldsig-core-schema.xsd をオンライン参照
3	xmldsig-core-schema.xsd	W3C XmlDsig Core 用のスキーマファイル

XML スキーマチェックは「検証」ではない。あくまで XML 構造が仕様に合致しているかをチェックするだけとなる。LE:XAdES:Lib V3 の生成する XAdES ファイルは XAdES V1.4.1 のスキーマチェックに成功するように実装されている（仕様通りになっている）為に毎回チェックする必要は無いと言える。外部で生成された XAdES ファイルの場合には XML スキーマチェックの機能を利用することに意味がある。標準（API の場合に引数を NULL 指定した場合）では XAdES の XML スキーマチェックとなるが、別途引数に任意の XML スキーマファイルを指定してチェックすることも可能となっている。なおスキーマチェックは LeXAdES3.checkSchema() で、エラーがあった場合に詳細は LeXAdES3.getErrors() で取得できる。

```
/** XMLスキーマのチェック [ユニコード]¥n¥n
*
* @param schemaFile [IN] チェックに利用するスキーマファイルを指定する
* @retval マイナス値 XDA_ERR_SCHEMA_PARSE等エラーコードが返る
* @note 事前にXMLをsetXml/loadXmlでセットしておく必要がある
* @note スキーマの外部参照の為にスキーマはファイル指定となる
* @note XAdESチェックは schemaFile に xsds/XAdES01903v141-201601.xsd を指定
* @note 署名済みの場合には各Signature要素をチェック、未署名なら全体をチェックする
*/
INT checkSchema(const WCHAR* schemaFile);

/** XAdESの解析/署名のエラー群取得¥n¥n
*
* @param snum [IN] 対象署名番号を指定（省略時は全ての署名）
* @retval LEX_ERRORの配列(std::vector) でエラー情報が返る、配列サイズが0ならエラー無し
* @note 解析（parse）と検証（verify）のエラー群が返る
*/
LEX_ERRORs        getErrors(INT snum = -1);
```

C++の XML スキーマチェック API：LeXAdES3 クラス

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

スキーマチェックをするだけであれば Lx3Cmd コマンドを利用した方が手軽である。エラーメッセージは複数エラーがある場合も全て画面に表示されるが、-result オプションにてエラー出力ファイルの指定も可能となっている。

```
// XAdES/XmlDsig スキーマチェックの実行（違反があればエラーメッセージを画面出力）
Lx3Cmd -schema -in xades.xml
```

コマンド（Lx3Cmd）によるスキーマチェック例

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

４．ＰＫＩ要素と電子署名付与

署名時の証明書（LpkCert）の利用方法やタイムスタンプ（LpkTimestamp）の使い方や独自証明書ストア等の、PKI 要素に関しては「ラング・エッジ PKI 基本ライブラリ LE:PKI:Lib マニュアル」（LePKI-manual.pdf）を参照。本書では重複するが独自証明書ストアは重要であるので記載する。

４．１．独自証明書ストア

一般に Windows 環境では「Windows 証明書ストア」が、Java 環境では「Java 証明書ストア」が利用される。LE:XAdES:Lib V3 は C++ にて開発されている関係で「Java 証明書ストア」が利用できない。その為に LE:XAdES:Lib V3 では「独自証明書ストア」を提供する。なお Windows 版では「Windows 証明書ストア」も利用が可能である。

「独自証明書ストア」は「信頼済みルート証明書（trusts）」「中間証明書（certs）」「検証情報（valids）」のディレクトリで構成される。LePKI::setStore() でディレクトリ指定と読み込みを行う。ディレクトリ指定を NULL にすると実行ファイルのあるディレクトリ下の store フォルダが指定される。例えば Lx3Cmd コマンドであれば、Lx3Cmd の下にあるディレクトリとなる。サーバ組み込み時には明示的なディレクトリの指定を推奨する。その他 addTrust() / addCert() / addCrl() / addOcsp() の各 API を使って独自証明書ストアに情報追加も可能である。なお署名証明書（と秘密鍵）は PKCS#12 形式のファイルで指定するので独自証明書ストアには含まれない。

フォルダ・ファイル	概要
Lx3Cmd	実行ファイル（Windows は Lx3Cmd.exe）
store	証明書ストアディレクトリ（setStore()で指定可能）
trusts	信頼済みルート証明書（トラストアンカー証明書）用ディレクトリ
*.cer / *.der	証明書群（拡張子 .cer と .der を読み込む）
certs	中間証明書用ディレクトリ
*.cer / *.der	証明書群（拡張子 .cer と .der を読み込む）
valids	失効情報（CRL/OCSP）用ディレクトリ
*.crl / *.ocsp	失効情報群（拡張子 .crl と .ocsp を読み込む）

独自証明書ストアのディレクトリ構成

LE:XAdES:Lib V3 の Windows 版では Windows 証明書ストアから、信頼済みルート証明書（"ROOT"）と中間証明書（"CA"）を取得する。setStoreFlag(FLAG flag) の引数として LPK_WIN_STORE をオフにして指定すると Windows 証明書ストアは利用されない。

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

４．２．署名鍵の置き場所による分類

電子署名システムを設計する上で最も重要な点は署名に使う（署名値の計算を行っている）署名鍵（秘密鍵）をどこに置くかを決めることである。LePKI ライブラリでは PKCS#12 ファイル形式の署名鍵/証明書をサポートしているが、署名鍵はコピーや漏洩をしない HSM（ハードウェア セキュリティ モジュール）や IC カード/USB トークン等に格納して運用されることが望ましい。なお電子証明書（公開鍵）は通常秘匿されるものではない（JPKI のマイナンバーカード等が例外）ので、X.509 ファイル形式で保存して利用して構わない。

署名鍵の保持には「ユーザー所有」「サーバー保管」「外部サービス」の 3 パターンがある。署名方式毎に LE:XAdES:Lib V3 の使い方を以下にまとめる。「ユーザー所有」の場合には「ローカル署名」と「クライアント署名」に分かれる。また「サーバー保管」の場合には PKCS#12 ファイルか HSM/クラウド HSM かによって実装方法が分かれる。「外部サービス」の場合にはリモート署名のデファクト標準である CSC API が良く使われている。（CSC：Cloud Signature Consortium）

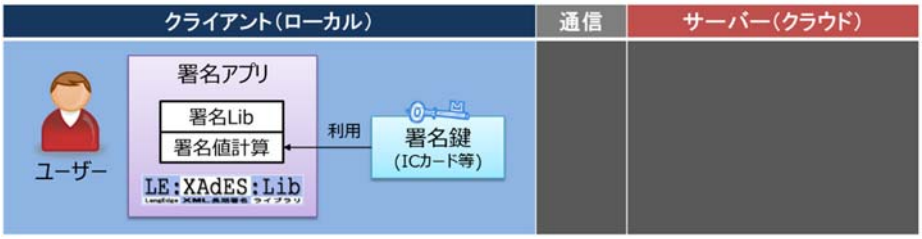
LE:XAdES:Lib V3 には、署名値を直接自分で計算する「直接署名」の組み込み方と、署名値を外部で計算する為の「仮署名」の組み込み方の 2 種類がある。署名方式を決めてどちらを利用するか決める必要がある。

署名鍵	署名方式		概要
ユーザー所有	ローカル署名		LE:XAdES:Lib をネイティブアプリに組み込んで利用。 ※ 直接署名 API：LeXAdES3.sign() 利用
	クライアント署名 (LE:Client:Sign)		LE:XAdES:Lib は署名サーバーに組み込む。 署名値の計算のみローカル PC で行う。 クライアント側に LE:Client:Sign を組込み連携する。 ローカル PC に LE:Client:Sign インストールが必要。 LE:XAdES:Lib のクライアント署名連携 API を利用可。 ・ **クライアント署名連携部の追加開発が必要。** ※ 仮署名 API：LeXAdES3.make() 利用
サーバー保管	サーバー署名	PKCS#12 安全性低	LE:XAdES:Lib は署名サーバーに組み込んで利用。 ※ 直接署名 API：LeXAdES3.sign() 利用
		HSM 安全性高	LE:XAdES:Lib は署名サーバーに組み込む。 署名値の計算のみ HSM/クラウド HSM で行う。 ・ **HSM/クラウド HSM 連携部の追加開発が必要。** ※ 仮署名 API：LeXAdES3.make() 利用
外部サービス (RSSP)	リモート署名		LE:XAdES:Lib は署名サーバーに組み込む。 署名値の計算のみリモート署名サービスで行う。 ・ **リモート署名サービス連携部の追加開発が必要。** ※ 仮署名 API：LeXAdES3.make() 利用

署名鍵の保持方法による電子署名システムの分類

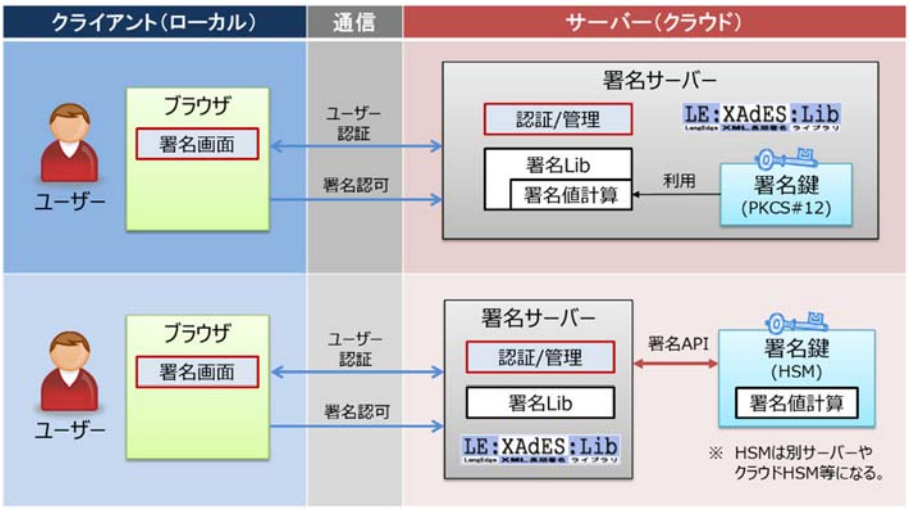
Copyright © 2012-2026 LangEdge,Inc. All rights reserved.

A）ローカル署名方式

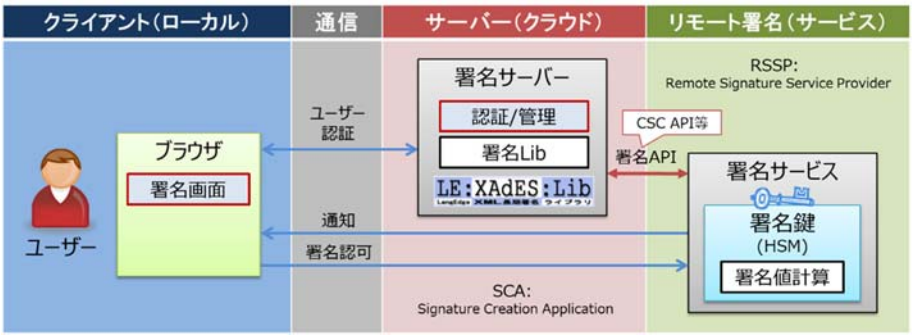


B）クライアント署名方式

C）サーバー署名方式（上：PKCS#12 ファイル利用／下：HSM かクラウド HSM 利用）



D）リモート署名方式



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

４．３．仮署名（HSM の利用等）

サーバー上で署名を行うサーバー署名の仕組みでは別ハードウェア上で署名値計算を行う HSM の利用が推奨される。署名値のみを外部で計算すると言う意味ではクライアント署名（７．クライアント署名 参照）と同じではあるが、サーバー側に置かれる HSM またはクラウド HSM にて署名値の計算を行う点で異なる。外部で計算された署名値を使う為には**仮署名**（LE:XAdES:Lib の造語：外部で署名値を計算する為のハッシュ値計算と途中結果の XML 取得を行う）を使う必要がある。HSM を使った場合の署名の処理手順を以下に示す。

処理手順		API 実行
LE:XAdES:Lib V3 の処理（仮署名）		
0	Detached 等の Reference 追加（準備）	LeXAdES3.addReference()
1	XML 仮署名の実行（署名鍵不要） ※ X.509 証明書指定が必要	LeXAdES3.make()
2	署名対象ハッシュ値計算と取得 ※ SignedInfo のハッシュ値	LeXAdES3.getSignedInfoHash()
3	仮署名済み XML の保存	LeXAdES3.saveXml()
HSM の処理（署名値の計算）		
4	ハッシュ値から署名値の計算	※ PKCS#11 や Java SPI や RESTful API 等で計算
LE:XAdES:Lib V3 の処理（署名値埋め込み）		
5	仮署名済み XML の読み込み	LeXAdES3.loadXml()
6	署名値の埋め込み ※ SignatureValue へセット	LeXAdES3.setSignatureValue()
7	署名済み XML の保存	LeXAdES3.saveXml()

HSM/クラウド HSM 用に認証局から電子証明書を発行してもらう為には、HSM 上に鍵ペアを生成して認証局から指定された形式の CSR（Certificate Signing Request：証明書発行要求）ファイルに署名鍵で署名して提出する必要がある。鍵ペアの生成方法や CSR ファイルの取得方法等は HSM 毎に異なるケースが多く、HSM を提供しているベンダーやサービスにやり方等を確認する必要がある。

現在 HSM の利用は可能ではあるが電子証明書の発行も含め比較的煩雑な手順が必要であるので、可能であれば別途サポートを受けることを推奨する。また HSM の利用方法はベンダーやサービス毎に異なるのでその点でも別途のサポートが必要となるケースは多い。

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

Java 用の LeXAdES.make() を使った実装サンプルが sample/LeXAdES3/java/LeXAdES_hsm.java にある。

Lx3Cmd コマンドを使った仮署名の実装サンプルが sample/LeXAdES3/cmd の下に、CmdHsmTest.bat / CmdHsmTest.sh としてある。

※ 仮署名コマンド実行イメージ

```
// 仮署名の実行（test.cer と test.p12 は同じ証明書）
Lx3Cmd -sign -make -det test.txt -cert x509 test.cer -out make.xml > HASH_VALUE
// 後処理：変数 $HASH_VALUE に実行結果をセット
// ハッシュ値から署名値の計算（HSM の代わり）
Lx3Cmd -sign -value $HASH_VALUE -cert p12 test.p12 test > SIGN_VALUE
// 後処理：変数 $SIGN_VALUE に実行結果をセット
// 署名値の埋め込み
Lx3Cmd -sign -setval $SIGN_VALUE -in make.xml -out make-bes.xml
```

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

４．４．証明書検証サーバー（CVS）の利用：GPKI/LGPKI 等

LE:PKI:Lib Ver1.05.R1 より GPKI（日本政府 PKI）や LGPKI（地方自治体 PKI）で利用されている、証明書検証サーバ（CVS）の利用が可能となった。CVS を利用した場合には署名証明書の認証パスの確認等の検証は全てサーバー側で行える。この為に CVS の利用は別途実装が必要となる。取得 API は LpkUtil::getCvs() である。取得時にはトラストアンカーとなるルート証明書（通常は BCA 自己署名証明書）の指定が必要となる。なお CVS のリクエストとレスポンスの仕様は、OCSP 仕様の拡張となっている。

```
/** CVS の取得(V1.08.R1 以降)¥n¥n
 *
 * @param cvs [OUT] 取得した cvs バイナリ
 * @param cert [IN] 失効確認の対象となる証明書を指定
 * @param parent [IN] 失効確認の対象となる証明書の親証明書を指定
 * @param opt [IN] GPKI 拡張の応答フォーマットをセット（value = 1 でセット）
 * @param url [IN] CVS 取得 URL(http のみ指定可能)
 * @param basic [IN] 基本認証用指定(空白文字で区切り"ID PASSWORD"で指定)
 * @param htype [IN] HTTP 通信方式を指定(Windows のみ指定可能、Linux は LPK_HTTP_SOCKET 固定)
 * @retval マイナス値 エラーコードが返る
 */
int getCvs(
        BINARY& cvs,                    // 結果
        const LpkCert& cert,            // 検証対象の証明書
        const LpkCert& parent,          // 通常 GPKI または LGPKI のルート証明書
        FLAG opt,                       // 1 をセットすると証明書群/CRL 群/OCSP 群を返す
        const CHAR* url,                // CVS サーバーの URL
        const CHAR* basic=NULL,         // Basic 認証情報（通常指定不要）
        LPK_HTTP_TYPE htype=LPK_HTTP_DEFALT     // HTTP 通信方式指定（通常指定不要）
);
```

C++の証明書検証サーバー利用 API：LpkUtil クラス

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

取得した CVS のバイナリイメージは LpkOcsp クラスの setBin() によりセットして結果の取得が可能となっている。

```
/** GPKI 拡張の認証パスステータスを取得（無い場合はマイナス値が返る）
 *
 * @param num [IN] 複数ある場合に情報番号を指定（通常 0 で良い）
 * @retval int GPKI 拡張の認証パスステータスが返る
 */
int getCertPathStatus (int num = 0) const;

/** 認証パスの証明書を取得(GPKI)
 *
 * @param num [IN] 複数ある場合に情報番号を指定（通常 0 で良い）
 * @retval LpkCerts 認証パスの証明書群が返る
 */
LpkCerts getCertPath (int num = 0) const;

/** CRL/ARL を取得(GPKI)
 *
 * @param num [IN] 複数ある場合に情報番号を指定（通常 0 で良い）
 * @retval LpkCrls 認証パスに必要な CRL/ARL 群が返る
 */
LpkCrls getRevocationList (int num = 0) const;

/** OCSP レスポンスを取得(GPKI)
 *
 * @param num [IN] 複数ある場合に情報番号を指定（通常 0 で良い）
 * @retval LpkOcsps 認証パスに必要な OCSP 群が返る
 */
LpkOcsps getOCSPResponse (int num = 0) const;
```

C++の LpkOcsp クラスの GPKI 関連 API

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

```
/** XAdES-X Long：検証情報を追加する¥n¥n
 * 1) XAdES-T から XAdES-X-Long に更新（署名+署名タイムスタンプの検証情報追加）
 * 2) XAdES-A から XAdES-A-XL に更新（アーカイブタイムスタンプの検証情報追加）
 *
 * @param xml [OUT] 追加に使われた検証情報の署名検証結果 XML が返される
 * @param vflag [IN] 検証フラグ(XD3_VERIFY_FLAG)を指定
 * @param verifyDate [IN] 検証時刻を GeneralizedTime 型で指定(通常は null で良い)
 * @param rootCert [IN] トラストアンカー証明書を指定、NULL なら任意の自己署名証明書まで
 * @param snum [IN] 対象署名番号を指定（省略時は最初の署名）
 * @retval 0 値 正常 / マイナス値 エラーコードが返る
 * @note 内部的には verify(xml, true, false, vflag, verifyDate, snum) と同じ処理
 * @note 検証エラーとなった場合には署名検証結果 XML にて理由等が確認できる
 */
INT addEsXL(
        BINARY& xml,                    // 検証結果 XML
        FLAG vflag = XV_DEFAULT,        // 検証フラグ
        const CHAR* verifyDate = NULL,  // 検証時刻指定(通常 NULL の現在時刻で良い)
        LpkCert* rootCert = NULL,       // トラストアンカー指定(CVS と同じルート証明書)
        INT snum = 0                    // 署名番号
);
```

```
/** 署名検証を行い結果を XML 形式で返す¥n¥n
 *
 * @param xml [OUT] 署名検証結果 XML が返される
 * @param embed [IN] true なら XAdES に検証情報を追加する(XAdES-T から XAdES-XL か XAdES-A から XAdES-A-XL にする)
 * @param value [IN] true なら証明書や検証情報を XML に埋め込む
 * @param vflag [IN] 検証フラグ(XD3_VERIFY_FLAG)を指定
 * @param verifyDate [IN] 検証時刻を GeneralizedTime 型で指定(通常は null で良い)
 * @param rootCert [IN] トラストアンカー証明書を指定、NULL なら任意の自己署名証明書まで
 * @param snum [IN] 対象署名番号を指定（省略時は全ての署名）
 * @retval 0 値 正常 / マイナス値 エラーコードが返る
 * @note embed = true として検証後に saveXml()/getXml() することで XAdES-XL/XAdES-A-XL に更新可能
 * @note 出力された XML は XdaVerifyXml クラスで読み込んで情報取得できる
 */
INT verify(
        BINARY& xml,                    // 検証結果 XML
        bool embed = false,             // ES-XL にする場合に true 指定
        bool value = false,             // 証明書や CRL/OCSP を検証結果 XML に追加する
        FLAG vflag = XV_DEFAULT,        // 検証フラグ
        const CHAR* verifyDate = NULL,  // 検証時刻指定(通常 NULL の現在時刻で良い)
        LpkCert* rootCert = NULL,       // トラストアンカー指定(CVS と同じルート証明書)
        INT snum = -1                   // 署名番号
);
```

C++の証明書検証サーバー結果を利用した検証 API：LeXAdES3 クラス

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

証明書検証サーバー（CVS）を使った LTV 化の手順説明：

手順	利用 API	処理内容
Step.1	LpkUtil::getCvs()	CVS の URL とトラストアンカー（BCA ルート）証明書と value=1 を指定して getCvs を実行する。
Step.2	LpkOcsp::setBin()	取得した CVS バイナリを LpkOcsp クラスにセットする。
Step.3	LpkOcsp:: getCertPathStatus()	取得した CVS 応答のステータスを取得してチェックする。 ステータスが 0 以外の場合はエラーなので以下手順は行えない。 ※　詳細は GPKI や LGPKI の技術仕様書を参照。
Step.4	LpkOcsp:: getCertPath()	取得した CVS 応答から証明書群 LpkCerts クラスを取得する。
Step.5	LePKI::addCert()	全取得証明書をループして順番に独自証明書ストアに追加する。
Step.6	LpkOcsp:: getRevocationList()	取得した CVS 応答から CRL 群 LpkCrls クラスを取得する。
Step.7	LePKI::addCrl()	全取得 CRL をループして順番に独自証明書ストアに追加する。
Step.8	LpkOcsp:: getOCSPResponse()	取得した CVS 応答から OCSP 群 LpkOcsps クラスを取得する。
Step.9	LePKI::addOcsp()	全取得 OCSP をループして順番に独自証明書ストアに追加する。
Step.10	LeXAdES3::setPki()	独自証明書ストアをセットした LePKI インタンスを LeXAdES3 クラスにセットする。
Step.11	LeXAdES3::addEsXL() LeXAdES3::verify()	検証を実行する。この時に getCvs 取得時に指定したトラストアンカー証明書を rootCert 引数に指定する。

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

```
////////////////////////////////////////////////////////////////////////
// ※ 本サンプルの実行には V3.00.R1 以降が必要です。
// ※ 本サンプルでは見やすくする為にエラーチェックを省いています。

////////////////////////////////////////////////////////////////////////
// CVS 処理
le::LpkUtil util;
le::LpkCert root;                           // CVS のルート証明書（必須）
le::BINARY bin;

// トラストアンカーの指定（GPKI のルート証明書）
rc = root.setFile(cvsRoot.c_str());
// CVS での検証実行
le::FLAG opt = 1;                           // CVS への CERTs/CRLs/OCSPs の取得指定
rc = util.getCvs(bin, cert, root, opt, cvsUrl.c_str());

////////////////////////////////////////////////////////////////////////
// CVS の解析
int stat;
le::LpkOcsp cvs;          // CVS は OCSP クラスにセットして解析する
le::LpkCerts certs;
le::LpkCrls crls;
le::LpkOcsps ocsps;

// CVS 結果のセット
rc = cvs.setBin(bin);
if (rc < 0)
// CVS のステータス確認
stat = cvs.getCertPathStatus();
if (stat != 0)
{
        std::cout << "get CVS status error(" << rc << ")." << std::endl;
        return Exit(1);
}
// CVS 情報から証明書の取得とセット
certs = cvs.getCertPath();
for (size_t i = 0; i < certs.size(); i++) {
        std::cout << "SET CVS CERT[" << i << "]" << std::endl;
        pki.addCert(certs[i]);
}
// CVS 情報から CRL の取得とセット
crls = cvs.getRevocationList();
for (size_t i = 0; i < crls.size(); i++) {
        std::cout << "SET CVS CRL[" << i << "]" << std::endl;
        pki.addCrl(crls[i]);
}
// CVS 情報から OCSP の取得とセット
ocsps = cvs.getOCSPResponse();
for (size_t i = 0; i < ocsps.size(); i++) {
        std::cout << "SET CVS OCSP[" << i << "]" << std::endl;
        pki.addOcsp(ocsps[i]);
}
```

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

```
////////////////////////////////////////////////////////////////////////
// 署名付与
le::LeXAdES3 xades;

// Detached 追加
le::LeReference ref;
rc = ref.setDetachedFile(detFile);
rc = xades.addReference(ref);

// 署名証明書/署名鍵の準備
le::LpkCert cert;
rc = cert.setPkcs12(certFile.c_str(), passwd.c_str());

// 署名付与(XAdES-BES)
le::FLAG sflag = le::XS_NONE;
const char* id = NULL;
const char* xpath = NULL;
rc = xades.sign(cert, sflag, id, xpath);
int snum = rc;

// タイムスタンプの追加(XAdES-BES)
le::LpkTs3161 ts;
rc = ts.set(tsUrl.c_str(), le::LPK_SHA_512);le::FLAG tflag = le::XT_NONE;
rc = xades.addEsT(ts, tflag, snum);

////////////////////////////////////////////////////////////////////////
// 検証情報の埋め込み

// PKI のセット(CVS の結果をセット済み)
rc = pades.setPki(pki);

// 検証と情報の埋め込み(XAdES-XL)
le::FLAG vflag = le::XV_NONE;              // 検証フラグ
const char* verifyDate = NULL;             // 通常 null で良い（試験用）
le::BINARY verifyXml;
rc = xades.addEsXL(verifyXml, vflag, verifyDate, &root, snum);

// xml ファイル出力
rc = xades.saveXml(outputFile.c_str());
```

C++による CVS 利用例

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

４．５．検証結果 XML 仕様（XdaVerifyXml）

[DUMMY]						ルート	検証結果によりルート要素名は異なる None：署名無し Verified：有効［VALID］ Indeterminate：不明［INDETERMINATE］（検証情報の不足） Failed：無効［INVALID］（失効や改ざん等） ERROR：その他エラー
	Sign					要素	署名情報（Signature 要素毎に出力される）
		num				属性	署名番号（検証順に独自に附番される）
		Id				属性	Signature 要素の Id 属性
		xpath				属性	Signature 要素のファイル内の位置を XPath 形式で示す
		level				属性	※1 LX3_XADES_LEVEL とレベル名を返す ex) level="13:XAdES-A"
		status				属性	※2 署名全体の検証結果ステータス
		SigInfo				要素	署名検証結果
			Id			属性	SignedInfo 要素の Id 属性
			status			属性	※2 SigInfo 全体の検証結果ステータス
			SigVal			要素	署名値（SignatureValue）の検証結果
				Id		属性	SignatureValue 要素の Id 属性
				sigAlg		属性	署名アルゴリズム（ECDSA は現在未サポート） "RSA/SHA1", "RSA/SHA256", "RSA/SHA384", "RSA/SHA512", "ECDSA/SHA1", "ECDSA/SHA256", "ECDSA/SHA384", "ECDSA/SHA512", "unknown"
				c14nAlg		属性	※3 C14N 正規化方式（XML 正規化）
				from		属性	※5 公開鍵情報取得場所
				key		属性	公開鍵の種類（"cert": 証明書、"keyVal": KeyValue 要素）
				status		属性	※2 署名値の検証結果ステータス
				Value		要素	署名値のバイナリ値（オプション）
			Ref			要素	参照先（Reference 要素）の検証結果 ▼1
				num		属性	参照番号（検証順に独自に附番される）
				Id		属性	Reference 要素の Id 属性
				type		属性	参照種別（Detached/Enveloped/Enveloping 等） "EPED"：Enveloped 指定（署名対象内に署名データ） "DOUT"： 外部 File 指定（署名データと外部の署名対象が別） "DIN"： 内部 Id 指定（署名データと内部の署名対象が別） "EPING"：Enveloping 指定（署名データ内に署名対象） "DMANI"：Manifest 指定（Detached） "EMANI"：Manifest 指定（Enveloping） "XOBJ"：XAdES の SignedProps 指定（長期署名用） "OPC"：Open Packaging Conventions 指定（未サポート） "unknown"： 不明
				URI		属性	Reference 参照先の URI
				hashAlg		属性	※4 ハッシュアルゴリズム

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

				trans	属性	参照変換方式（オプション） "EPED"：XmlDsig Eveloped Transform "B64"：Base64 Transform "XPATH"：XmlDsig XPath Transform "FILTER2"：XmlDsig Filter2 Transform "RELSHIP"：OOXML Relationship Transform "unknown"： 不明
				status	属性	※2 参照先の検証結果ステータス
				Ref	要素	参照先が Manifest の場合の Manifest 参照先の検証結果 更に参照先が Manifest の場合には再帰され Ref 子要素を持つ ▼1 と同じ属性と子要素となるので属性と子要素を省略
		Certs			要素	署名証明書の認証パス ▼2
			status		属性	※2 認証パス全体の検証結果ステータス
			time		属性	※6 検証時刻
			Cert		要素	1 証明書の検証結果（認証パス分存在）
				name	属性	証明書一般名（Subject の CommonName）
				status	属性	※2 証明書の検証結果ステータス
				valid	属性	証明書検証種類（CRL か OCSP か） CRL：検証は CRL による OCSP：検証は OCSP による
				from	属性	※5 情報取得場所
				limit	属性	※6 証明書有効期限（失効予定日時）
				id	属性	証明書のシリアル番号（HEX 文字列）
				Value	要素	証明書のバイナリ値（オプション）
				Crl	要素	CRL（証明書失効リスト）
					name	属性	CRL 発行者一般名（Issuer の CommonName）
					from	属性	※5 情報取得場所
					update	属性	※6 CRL 発行日時
					Value	要素	CRL のバイナリ値（オプション）
				Ocsp	要素	OCSP（オンライン証明書状態プロトコル）
					name	属性	OCSP 署名証明書一般名（Subject の CommonName）
					from	属性	※5 情報取得場所
					update	属性	※6 OCSP 発行日時
					Value	要素	OCSP のバイナリ値（オプション）
		XAdES			要素	長期署名要素の検証結果
			Id		属性	SignedSignProps 要素の Id 属性
			time		属性	SigningTime 要素の時刻（"YYYY-MM-DDThh:mm:ssZ"）
			status		属性	※2 長期署名要素全体の検証結果ステータス
			SigTs		要素	署名タイムスタンプ検証結果（1 署名中に 1 つのみ）
				Id	属性	SignatureTimeStamp 要素の Id 属性
				status	属性	※2 署名タイムスタンプ全体の検証結果ステータス

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

				Token		要素	署名タイムスタンプのタイムスタンプトークン情報
					date	属性	※6 署名タイムスタンプ時刻（ミリ秒省略）
					hashAlg	属性	※4 ハッシュアルゴリズム
					c14nAlg	属性	※3 C14N 正規化方式（XML 正規化）
					status	属性	※2 署名タイムスタンプの検証結果ステータス
				Certs		要素	署名タイムスタンプ TSA 証明書の認証パス ▼2 と同じ属性と子要素となるので属性と子要素を省略
			ArcTs			要素	アーカイブタイムスタンプ検証結果（1 署名中に複数可）
				num		属性	アーカイブタイムスタンプ番号（古い方から順番に附番）
				Id		属性	ArchiveTimeStamp 要素の Id 属性
				status		属性	※2 このアーカイブタイムスタンプ全体の検証結果ステータス
				Token		要素	アーカイブタイムスタンプのタイムスタンプトークン情報
					date	属性	※6 アーカイブタイムスタンプ時刻（ミリ秒省略）
					hashAlg	属性	※4 ハッシュアルゴリズム
					status	属性	※2 アーカイブタイムスタンプの検証結果ステータス
				Certs		要素	アーカイブタイムスタンプ TSA 証明書の認証パス ▼2 と同じ属性と子要素となるので属性と子要素を省略

※1 署名のレベル	LX3_XADES_LEVEL のレベルと文字列は以下 XL_DSIG = "01:XmlDsig" XL_MAKE = "09:XAdES-MAKE" XL_BES = "10:XAdES-BES" XL_T = "11:XAdES-T" XL_XL = "12:XAdES-XL" XL_A = "13:XAdES-A" XL_AXL = "14:XAdES-A-XL"
※2 検証結果ステータス	Verified：有効 Indeterminate：不明（検証情報の不足） Failed：異常（失効や改ざん等） ERROR：その他エラー
※3 C14N 正規化方式（XML 正規化）	"INC"：C14N V1.0 Inclusive "INC-WC"：C14N V1.0 Inclusive with comment "EXC"：C14N V1.0 Exclusive "EXC-WC"：C14N V1.0 Exclusive with comment "V11"：C14N V1.1 Inclusive "V11-WC"：C14N V1.1 Inclusive with comment "unknown"： 不明な方式（未サポート）
※4 ハッシュアルゴリズム	"MD5", "SHA1", "SHA256", "SHA384", "SHA512", "unknown"
※5 情報取得場所	"data"： データ(署名データ等)自身から取得 "orgStore"： 独自証明書ストアから取得 "addStore"： 独自証明書ストアに後から追加 "winStore"：Windows 証明書ストアから取得 "network"： ネット接続から取得(キャッシュも含む) "valid"： 検証データ(OCSP か CRL 署名書)から取得 "notUse"： 取得しない(ネットワーク利用しない) "cache"： キャッシュ(ネット接続取得済み情報)

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

	"tstData"：XAdES の TimeStampToken から取得 "keyInfo"：XmlDsig の KeyInfo から取得 "sigValid"： 署名証明書用の CertificateValue 等 "stsValid"：STS の TimeStampValidationData から "atsValid"：ATS の TimeStampValidationData から "unknown"： 不明
※6 時刻情報	"YYYYMMDDhhmmssZ" 形式

```
<?xml version="1.0" encoding="UTF-8"?>
<Verified>
  <Sign num="0" Id="Id" xpath="/Signature" level="13:XAdES-A" status="Verified">
    <SigInfo Id="Id-Si-3" status="Verified">
      <SigVal Id="Id-Sv-4" sigAlg="RSA/SHA256" c14nAlg="INC" from="keyInfo" key="cert" status="Verified"/>
      <Ref num="0" Id="Id-Ref-2" type="EPING" URI="#Id-Obj-1" hashAlg="SHA256" trans="INC"
        status="Verified"/>
      <Ref num="1" Id="Id-Ref-7" type="XOBJ" URI="#Id-xadesSp-6" hashAlg="SHA256" trans="INC"
        status="Verified"/>
    </SigInfo>
    <Certs status="Verified" time="20220731084120Z">
      <Cert name="LE Test2 100012" id="100012" limit="20230401000000Z" from="keyInfo" valid="CRL"
        status="Verified">
        <Crl from="sigValid" name="LangEdge CA Root 01" update="20220730192001Z"/>
      </Cert>
      <Cert name="LangEdge CA Root 01" id="038D7EA4C68003" limit="20380429005226Z" from="sigValid"
        status="Verified"/>
    </Certs>
    <XAdES Id="Id-xadesSp-6" time="2022-07-31T08:41:20Z" status="Verified">
      <SigTS Id="Id-STS-1" status="Verified">
        <Token date="20220731084120Z" hashAlg="SHA512" c14nAlg="INC" status="Verified"/>
        <Certs status="Verified" time="20220731084134Z">
          <Cert name="LE TSA 200020" id="200020" limit="20260401000000Z" from="data" valid="CRL"
            status="Verified">
            <Crl from="stsValid" name="LangEdge CA Root 02" update="20220730192001Z"/>
          </Cert>
          <Cert name="LangEdge CA Root 02" id="038D7EA4C68004" limit="20380429005401Z" from="tstData"
            status="Verified"/>
        </Certs>
      </SigTS>
      <ArcTS num="0" Id="Id-ATS-1" status="Verified">
        <Token date="20220731084134Z" hashAlg="SHA512" status="Verified"/>
        <Certs status="Verified" time="20220731084228Z">
          <Cert name="LE TSA 200020" id="200020" limit="20260401000000Z" from="data" valid="CRL"
            status="Verified">
            <Crl from="atsValid" name="LangEdge CA Root 02" update="20220730192001Z"/>
          </Cert>
          <Cert name="LangEdge CA Root 02" id="038D7EA4C68004" limit="20380429005401Z" from="tstData"
            status="Verified"/>
        </Certs>
      </ArcTS>
    </XAdES>
  </Sign>
</Verified>
```

検証結果 XML の例

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

※ 検証結果 XML "error" 要素のメッセージ一覧

PKI エラー（通常検証エラーとして出力されるメッセージ）	
"not trusted."	ルート証明書が信頼されていない 対処：ルート証明書を確認して証明書ストアにセットする
"cannot build cert path."	認証パス構築ができなかった/LPK_ERR_PATH_BUILD エラー 対処：認証パスに必要な中間 CA 証明書等を確認してセットする
"cert path build error."	認証パス構築時のエラー/LPK_ERR_PATH_BUILD 以外のエラー 対処：認証パスの全ての証明書の確認
"cannot get valid data."	CRL/OCSP の取得に失敗した 対処：ネットワーク環境や取得先 URL へのアクセス権の確認
"revoked by OCSP."	OCSP により失効している
"revoked by CRL."	CRL により失効している
"crl strict check error."	CRL 発行日時の厳密チェックエラー（署名日時より前に発行された CRL を利用している） ※ XV_DATE_SIG_REVO か XV_DATE_TSA_REVO の指定時のみチェック
"certificate expired."	証明書の期限切れ（認証パスに期限切れの証明書がある）

XAdES エラー（XAdES 仕様違反）		
Ref	"not found"	参照先情報 Reference が見つからない
	"invalid hash"	参照先のハッシュ値が一致しない 対処：参照先が正しいか確認する
Certs	"Sign[N] Signing cert not found"	署名証明書が見つからない
	"Sign[N] TSA cert not found"	TSA 証明書が見つからない
	"Sign[N] Signing cert verify error"	署名証明書の検証に失敗した
	"Sign[N] TSA cert cert verify error"	TSA 証明書の検証に失敗した
	"Sign[N] Signing cert verify xml is empty"	署名証明書の検証結果が空だった
	"Sign[N] TSA cert cert verify xml is empty"	TSA 証明書の検証結果が空だった
	"Sign[N] Signing cert xml add error"	署名証明書のノード追加に失敗
	"Sign[N] TSA cert cert xml add error"	TSA 証明書のノード追加に失敗
SigTS	"Sign[N] no SignatureTimestamp"	署名タイムスタンプがない
	"Sign[N] TSA cert not found"	署名タイムスタンプから TSA 証明書が取得できない
ArcTS	"Sign[N] no ArchiveTimestamp"	アーカイブタイムスタンプがない
	"Sign[N] TSA cert not found"	署名タイムスタンプから TSA 証明書が取得できない
Token	"invalid timestamp token sign."	タイムスタンプトークンの検証に失敗

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

４．６．タイムスタンプ取得エラーのデバッグ

Ver3.01.R3 よりド署名タイムスタンプやアーカイブタイムスタンプを付与する場合に、タイムスタンプ取得時のエラー情報が LeXAdES3 クラスの getTsError() により取得可能となった。タイムスタンプ取得時のエラーには大きく分けて、「HTTP 通信時の API エラー」と「HTTP 通信時の Web サーバーエラー」と「タイムスタンプ要求への応答解析時のエラー」の 3 種類がある。getTsError() の結果は Lx3Cmd コマンドを使った場合には画面に表示される。

１）HTTP 通信時の API エラー

「HTTP 通信時の API エラー」は、HTTP 通信をするための API がエラーを返した場合となる。例えば存在しないサーバー名を利用したような場合に表示されるが、各 HTTP 通信方式において getTsError() は以下のような応答を返す。

○ 1-1：Socket 通信時の API エラー例

HTTP OPEN ERROR (Socket)：-1004 LeSocket error: (-1004)	オープン時のサブエラーコード：-1004 ソケット API における内部エラー
-1004：ID_SOCK_GETHOST_ERROR	

○ 1-2：WinHTTP 通信時の API エラー例

HTTP REQUEST ERROR (WinHTTP)：-1105 WinHTTP error(12007)	リクエスト時のサブエラーコード：-1105 WinHTTP の API エラーコード：12007
-1105：ID_WHTTP_REQUEST3_ERROR 12007：The server name cannot be resolved (サーバー名が解決できません)	

○ 1-3：WinInet 通信時の API エラー例

HTTP REQUEST ERROR (WinInet)：-1005 WinInet error(12007)	リクエスト時のサブエラーコード：-1005 WinHTTP の API エラーコード：12007
-1005：ID_WINET_REQUEST3_ERROR 12007：The server name cannot be resolved (サーバー名が解決できません)	

※ サブエラーコードは本ライブラリ独自の定義であり本章最後に示す。
※ WinHTTP と WinInet では 2 行目に Windows システムのエラーコードが返される。

２）HTTP 通信時の Web サーバーエラー

「HTTP 通信時の Web サーバーエラー」は、HTTP 通信自体は送信したが Web サーバーがエラーを返した場合となる。例えばサーバー名は存在するが、存在しないパスの URL を指定したような場合に表示されるが、各 HTTP 通信方式において getTsError() は以下のような応答を返す。

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

○ 2-1：Socket 通信時の Web サーバーエラー例

HTTP REQUEST ERROR (Socket)：-1001 HTTP STATUS (Socket) = 404	リクエスト時のサブエラーコード：-1001 応答 HTTP ステータスコード
-1001：ID_SOCK_HTTPSTAT_ERROR 404：Not Found	

○ 2-2：WinHTTP 通信時の Web サーバーエラー例

HTTP REQUEST ERROR (WinHTTP)：-2105 HTTP STATUS (WinHTTP) = 404	リクエスト時のサブエラーコード：-2105 応答 HTTP ステータスコード
-2105：ID_SYS_STATUS_ERROR 404：Not Found	

○ 2-3：WinInet 通信時の Web サーバーエラー例

HTTP REQUEST ERROR (WinInet)：-2005 HTTP STATUS (WinInet) = 404	リクエスト時のサブエラーコード：-2005 応答 HTTP ステータスコード
-2005：ID_SYS_STATUS_ERROR 404：Not Found	

HTTP 通信では Web サーバーからのレスポンスとして 3 桁の HTTP ステータスコード（RFC 9110 定義）が返される。正常に通信に成功した場合には 200（OK）が返される。通常 HTTP 通信に失敗すると 400 番台か 500 番台のステータスコードが返される。詳しくは RFC 9110 を参照。

ステータスコード	意味
100 番台	1xx Informational 情報 通常 100 番台のステータスコードが返されることはない。
200 番台	2xx Success 成功 本ライブラリでは 200 以外はエラーとする。
300 番台	3xx Redirection リダイレクション リダイレクトには未対応なので本ステータスは通常返されない。
400 番台	4xx Client Error クライアントエラー 不正な URL 等へのリクエストようにクライアント側のエラーがあった。 以下によるあるエラーステータスを示す。 400 Bad Request (不正なリクエスト、POST に対応していない等) 404 Not Found (指定された URL が見つからない) 408 Request Timeout (リクエストがタイムアウトした)
500 番台	5xx Server Error サーバーエラー サーバー側で何らかの問題を生じた。サーバー管理者へ確認する。

HTTP ステータスコードの種類

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

3）タイムスタンプ要求への応答解析時のエラー（タイムスタンプレスポンスのエラー）

「タイムスタンプ要求への応答解析時のエラー」は、HTTP 通信自体は成功（HTTP ステータスが 200）したが Web サーバーが返した応答が正しいタイムスタンプレスポンス形式（タイムスタンプトークンを含んだレスポンス形式）になっていない場合となる。例えば URL 自体は存在するがサーバーがエラーのタイムスタンプレスポンスを返すか、HTML 文字列を返すような場合に表示される。この種類のエラーは HTTP 通信方式には依存せず getTsError() は以下のような応答を返す。

○ 3：レスポンス解析時のエラー例

LPK_ERR_TS_PARSE (Socket) : Timestamp Response parse error (-11)
-11：解析時にエラーの場所を示す内部コード、LpkTimestamp.cpp で確認する。

レスポンスの内容を確認すると HTML 形式になっていたりエラーレスポンスになっていたりするので、内容を確認すると原因が分かる場合が多い。レスポンスは LpkTimestamp クラスの setHttp() の 2 番目の引数としてデバッグファイルのパスを指定して取得することができる。Lx3Cmd コマンドを使う場合には -tserr 引数で指定が可能となる。-tserr 引数を指定した場合には出力されたエラーレスポンスの出力ファイルが以下のように画面表示される。

```
$ Lx3Cmd -tserr -ts ...
LPK_ERR_TS_PARSE (WinHTTP) : Timestamp Response parse error (-11)
Response file: .¥20250521170250-tserr.dat
sign command: xades3 error (-5440/XDA_ERR_GET_NET_SIGTS).
署名タイムスタンプのネットからの取得エラー
```

Lx3Cmd コマンドでは出力されるファイル名は YYYYMMDDhhmmss-tserr.dat に固定となる。最初の部分は年月日時分秒を示す数字となっている。-tserr 引数には出力ディレクトリを指定することも可能となっている。例えば -tserr C:¥¥Temp や -tserr /tmp 等である。出力ディレクトリを省略した場合には現在のディレクトリ（カレントディレクトリ）となる。

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

※A：ソケットを利用した HTTP 通信時にサブエラーコード（Linux デフォルト通信方式）

```
// エラー定義（LeHttpConnectSocket.cpp）
enum LE_HTTP_WIN32_ERROR {
        ID_NO_ERROR                 = 0,
        ID_SOCK_WINSOCK_ERROR       = -1000,
        ID_SOCK_HTTPSTAT_ERROR      = -1001,
        ID_SOCK_NOCLEN_ERROR        = -1002,
        ID_SOCK_SSL_ERROR           = -1003,
        ID_SOCK_GETHOST_ERROR       = -1004,
        ID_SOCK_SOCKET_ERROR        = -1005,
        ID_SOCK_CONNECT_ERROR       = -1006,
        ID_SOCK_PROXY_ERROR         = -1007,
        ID_SOCK_PROXYSTAT_ERROR     = -1008,
        ID_SYS_URL_ERROR            = -2001,
        ID_SYS_NOTINIT_ERROR        = -2002,
        ID_SYS_CLENSIZE_ERROR       = -2003,
        ID_SYS_READ_ERROR           = -2004,
        ID_SYS_RESP_ERROR           = -2005,
        ID_SYS_SSL_WRITE_ERROR      = -2006,
        ID_SYS_SSL_READ_ERROR       = -2007,
        ID_EXCEPTION_FATAL          = -9998,
        ID_UNK_EXCEPT_FATAL         = -9999,
};
```

※B：WinHTTP を利用した HTTP 通信時にサブエラーコード（Windows デフォルト通信方式）

```
// エラー定義（LeHttpConnectWinHTTP.cpp）
enum LE_HTTP_WINHTTP_ERROR {
        ID_NO_ERROR                 = 0,
        ID_WHTTP_OPEN_ERROR         = -1101,
        ID_WHTTP_CONNECT_ERROR      = -1102,
        ID_WHTTP_REQUEST1_ERROR     = -1103,
        ID_WHTTP_REQUEST2_ERROR     = -1104,
        ID_WHTTP_REQUEST3_ERROR     = -1105,
        ID_WHTTP_REQUEST4_ERROR     = -1106,
        ID_WHTTP_DATAGET1_ERROR     = -1107,
        ID_WHTTP_DATAGET2_ERROR     = -1108,
        ID_WHTTP_DATAGET3_ERROR     = -1109,
        ID_WHTTP_PROXY_ERROR        = -1110,
        ID_SYS_URL_ERROR            = -2101,
        ID_SYS_NOTINIT_ERROR        = -2102,
        ID_SYS_CLENSIZE_ERROR       = -2103,
        ID_SYS_READ_ERROR           = -2104,
        ID_SYS_STATUS_ERROR         = -2105,
        ID_EXCEPTION_FATAL          = -9998,
        ID_UNK_EXCEPT_FATAL         = -9999,
};
```

※C：WinInet を利用した HTTP 通信時にサブエラーコード

```
// エラー定義（LeHttpConnectWin32.cpp）
enum LE_HTTP_WIN32_ERROR {
        ID_NO_ERROR                 = 0,
        ID_WINET_OPEN_ERROR         = -1001,
```

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

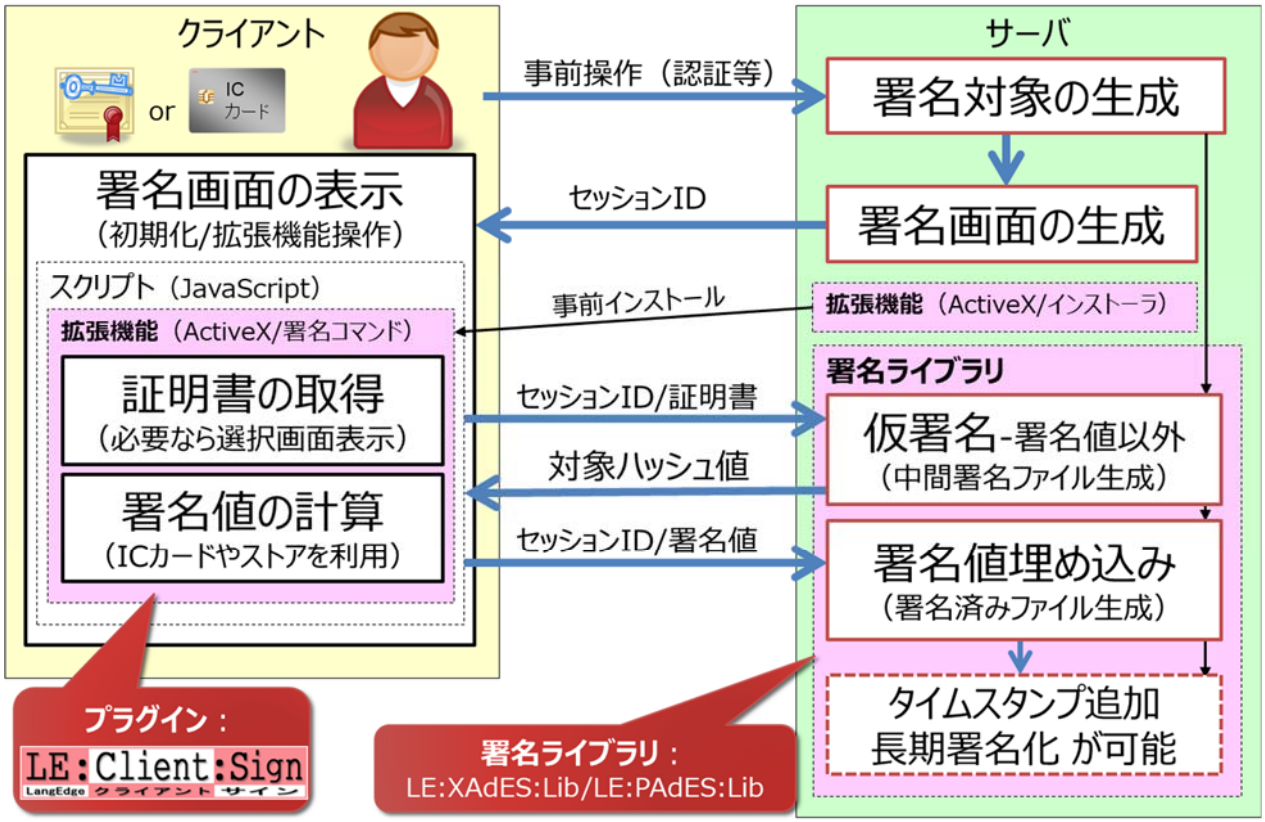
```
        ID_WINET_CONNECT_ERROR  = -1002,
        ID_WINET_REQUEST1_ERROR = -1003,
        ID_WINET_REQUEST2_ERROR = -1004,
        ID_WINET_REQUEST3_ERROR = -1005,
        ID_WINET_GETCLEN_ERROR  = -1006,
        ID_WINET_PROXY_ERROR    = -1007,
        ID_SYS_URL_ERROR        = -2001,
        ID_SYS_NOTINIT_ERROR    = -2002,
        ID_SYS_CLENSIZE_ERROR   = -2003,
        ID_SYS_READ_ERROR       = -2004,
        ID_SYS_STATUS_ERROR     = -2005,
        ID_EXCEPTION_FATAL      = -9998,
        ID_UNK_EXCEPT_FATAL     = -9999,
};
```

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

５．クライアント署名（Ver2.1）

現在サポートしているクライアント署名は Ver2.1 となり、別プロダクト LE:Client:Sign（LeClientSign）となる。この為マニュアルも「クライアント署名 V2.1 マニュアル」として別途提供される為にここでは概要のみ開設する。クライアント署名 V2.1 のマニュアルや実行環境は clientV2 ディレクトリ下に格納されている。クライアント署名 Ver2.0 以降では実行環境として、ActiveX を使った IE11（V1.0 サポート済み）と、署名コマンドを使った Edge/Chrome が利用可能となっている。

クライアント署名 V2 マニュアル　clientV2/doc/LeClientSign2-manual.pdf



クライアント署名概要

クライアント署名 V2 のライセンス：

ご契約ライブラリ	ご契約ライセンス	クライアント署名 V2 ライセンス
XML 長期署名 LE:XAdES:Lib	サーバ/開発ライセンス	ライセンス範囲（1 サーバ）内で利用可能。
	OEM ライセンス	LE:Client:Sign ライセンスが**別途必要**。※1
	ASP ライセンス	利用可能。

※1：ご利用される場合にはお問合せください。

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

５．１．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.

５．２．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）	

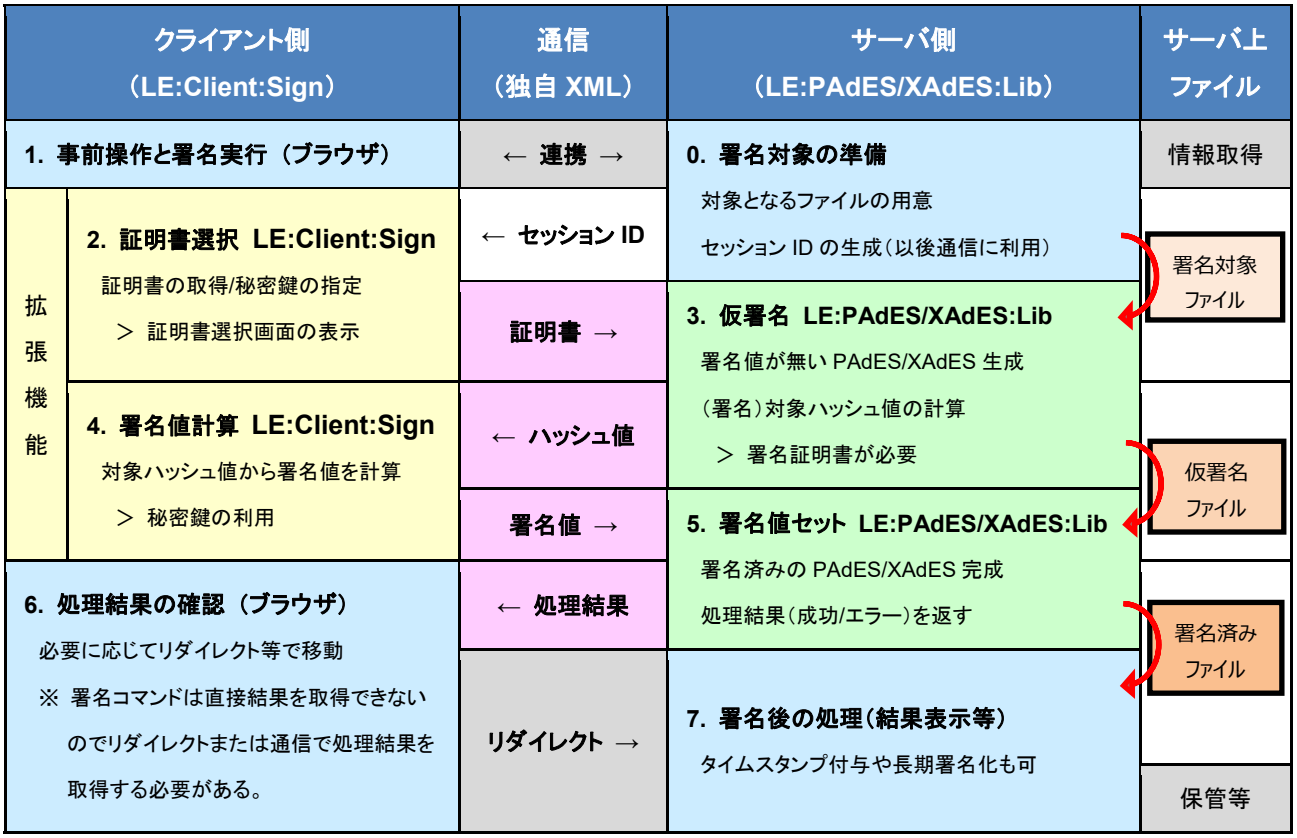
ActiveX プラグインと署名コマンドの提供モジュール

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

５．３．署名サーバー連携

クライアント署名は署名サーバーと連携して署名処理を行う。サーバー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。サーバー側にて独自 XML 通信プロトコルに対応した実装を行う必要がある。またサーバー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。

またサーバー側のライブラリを 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、LE:XAdES:Lib では XdaClientXml を提供している。



クライアントとサーバー間の連携とサーバー側の処理

※ LE:Client:Sign Ver2.1 よりサーバー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.

５．４．署名サーバー実装例

１）実装サンプル：Tomcat 用の Java 実装例

clientV2/bin_cmd/server_xades/XLeCSignCert.java　：証明書取得時
clientV2/bin_cmd/server_xades/XLeCSignSign.java　：署名値取得時
clientV2/bin_cmd/server_xades/XLeCSignShow.java　：結果表示時

２）Lx3Cmd のダミークライアント -client 引数による実装例

// Windows
sample/LaXAdES3/cmd/CmdClientTest.bat　：シンプルな利用方法
sample/LaXAdES3/java/JavaClientTest.bat　：Java 試験
// Linux
sample/LaXAdES3/cmd/CmdClientTest.sh　：シンプルな利用方法
sample/LaXAdES3/java/JavaClientTest.sh　：Java 試験
// Java
sample/LaXAdES3/java/LeXAdES_client.java　：Java からの実装例

```
// クライアント側（LCS_CERT）：証明書選択 > 証明書 XML
$ Lx3Cmd.sh -client -cert p12 LeTest.p12 test -sid T01 > CERT.xml
// サーバー側（LCS_HASH）：仮署名 < 証明書 XML > ハッシュ値 XML
$ Lx3Cmd.sh -server -eped -in base.xml -out make.xml < 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
$ Lx3Cmd.sh -client -cert p12 LeTest.p12 test -sid T01 < HASH.xml > SIGN.xml
// サーバー側（LCS_RSLT）：署名値埋め込み < 署名値 XML > 結果 XML
$ Lx3Cmd.sh -server -in make.xml -out sign.xml < SIGN.xml > RSLT.xml
// 結果確認（本来はクライアント側で確認）
$ cat RSLT.xml
```

Lx3Cmd 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.

５．５．LE:Client:Sign フォルダ構成

LE:XAdES:Lib V3 のリリースには LE:Client:Sign のバイナリ版が clientV2 として同梱される。バイナリ版であるのでソースコードは提供されない。また LE:Client:Sign のライセンスは別途となるので注意が必要となる。ソースコードは LE:Client:Sign 製品版にて取得可能。

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_xades]	LE:XAdES:Lib V3 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
[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

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

付録Ａ．エラーコード

Ａ．１．LeUtil エラーコード（src/LeCommon/LeUtil.h）-1000 ～ -1099

LeUtil のエラーコードは全体を通して利用される汎用ユーティリティが返すエラーコードです。ただし表面に表示されることは無いはずです。

```
LPU_ERR_FILEWOPEN       = -1000,        // 書き込みファイルオープンエラー
LPU_ERR_FILEROPEN       = -1001,        // 読み込みファイルオープンエラー
```

Ａ．２．Lx3Cmd エラーコード（src/LeXAdES3/Lx3Cmd/Lx3Cmd.h）-1200 ～ -1399

Lx3Cmd のエラーコードはコマンドライン Lx3Cmd ツールが表示するエラーコードです。

```
LXC_NO_ERROR            =     0, // 正常終了（ゼロ保証）
LXC_ERR_SIGN_TARGET_ARG = -1200, // 署名対象の引数エラー
LXC_ERR_SIGN_ARG        = -1201, // 署名の引数エラー
LXC_ERR_TS_ARG          = -1202, // タイムスタンプの引数エラー
LXC_ERR_EMBED_ARG       = -1203, // 検証情報埋め込みの引数エラー
LXC_ERR_VERIFY_ARG      = -1204, // 検証の引数エラー
LXC_ERR_SCHEMA_ARG      = -1205, // スキーマチェックの引数エラー
LXC_ERR_MANIFEST_ARG    = -1206, // Manifest の引数エラー
LXC_ERR_CLT_SIGN        = -1210, // クライアント署名処理エラー
LXC_ERR_CLT_END         = -1211, // クライアント終了処理エラー
LXC_ERR_CLT_CANCEL      = -1212, // 証明書選択キャンセル
LXC_ERR_CLT_ARG         = -1213, // クライアント引数エラー
LXC_ERR_SRV_MAKE        = -1220, // サーバー仮署名処理エラー
LXC_ERR_SRV_ARG         = -1221, // サーバー引数エラー
LXC_ERR_SRV_CERT        = -1222, // サーバー署名の証明書エラー
LXC_ERR_SRV_XML         = -1223, // サーバー応答 XML エラー
LXC_ERR_SRV_EMBED       = -1224, // サーバー埋め込みエラー
LXC_ERR_SIGN_NO_CERT    = -1233, // 署名付与証明書未指定エラー
LXC_ERR_SIGN_NO_TS      = -1234, // 署名付与タイムスタンプ未指定エラー
LXC_ERR_INVALID_LEVEL   = -1235, // XAdES のレベルが正しくない
LXC_ERR_VERIFY_XML      = -1241, // 検証 XML エラー
LXC_ERR_VERIFY_NO_REPORT = -1242, // 検証結果レポートエラー
LXC_ERR_VERIFY_INDETERM = -1250, // 検証結果が不明
LXC_ERR_VERIFY_INVALID  = -1251, // 検証結果が不正
LXC_ERR_RESULT_XML_OPEN = -1270, // 検証結果 XML ファイルが開けない
LXC_ERR_READ_FILE       = -1300, // Lx3Cmd ファイル読み込みエラー
LXC_ERR_WRITE_FILE      = -1301, // Lx3Cmd ファイル書き込みエラー
LXC_ERR_NOT_SUPPORT     = -1390, // Lx3Cmd 未対応機能
LXC_ERR_SYSTEM          = -1397, // システムエラー（インスタンス生成失敗等）
LXC_ERR_EXCEPTION       = -1398, // 不明例外発生
LXC_ERR_UNKNOWN         = -1399, // 不明エラー
```

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

Ａ．３．LePKI エラーコード（include/LePKI/LePKI.h）-3000 ～ -3999

LePKI のエラーコードは PKI（公開鍵基盤）の操作に関するエラーで生じます。

```
LPK_NO_ERROR            =     0, // 正常終了（ゼロ保証）
LPK_ERR_ARGUMENT        = -3000, // 引数異常
LPK_ERR_NOT_INIT        = -3001, // 未初期化
// LePKI
LPK_ERR_PKI_INIT        = -3100, // LePKI 未初期化
LPK_ERR_WINSTORE        = -3101, // Windows 証明書ストアエラー
LPK_ERR_GETMODULE       = -3102, // 実行ファイルパス取得エラー
LPK_ERR_SET_STORE       = -3103, // 証明書ストアセットエラー
LPK_ERR_SET_CACHE       = -3104, // CRL キャッシュセットエラー
LPK_ERR_ADD_REPOSIT     = -3105, // リポジトリサーバセットエラー
LPK_ERR_ADD_CVS         = -3106, // 証明書検証サーバセットエラー
LPK_ERR_ADD_BASICAUTH   = -3107, // 基本認証セットエラー
LPK_ERR_PATH_BUILD      = -3108, // 認証パス構築エラー
LPK_ERR_PATH_VALID      = -3109, // 認証パスの検証情報収集エラー
LPK_ERR_PATH_VERIFY     = -3110, // 認証パスの検証エラー
LPK_ERR_GET_CRL         = -3111, // CRL 取得エラー
LPK_ERR_GET_PARENT      = -3112, // 親証明書取得エラー
// LpkCert
LPK_ERR_WCERT_SETFILE   = -3200, // Windows 証明書ファイルセットエラー
LPK_ERR_WCERT_SETP12    = -3201, // Windows 証明書(PKCS#12)セットエラー
LPK_ERR_WCERT_NOTFOUND  = -3202, // Windows 証明書未定義エラー
LPK_ERR_WCERT_CANCEL    = -3203, // Windows 証明書選択キャンセル
LPK_ERR_WCERT_CAPI      = -3204, // Windows 証明書 CAPI エラー
LPK_ERR_OCERT_SETFILE   = -3210, // OpenSSL 証明書ファイルセットエラー
LPK_ERR_OCERT_SETBIN    = -3211, // OpenSSL 証明書バイナリセットエラー
LPK_ERR_OCERT_SETP12    = -3212, // OpenSSL 証明書(PKCS#12)セットエラー
LPK_ERR_OCERT_INITP11   = -3215, // OpenSSL 証明書(PKCS#11)初期化エラー
LPK_ERR_OCERT_SIGNP11   = -3216, // OpenSSL 証明書(PKCS#11)署名エラー
LPK_ERR_CERT_GET_INFO   = -3220, // 証明書からの情報取得エラー(情報が無い場合を含む)
// LpkCrl
LPK_ERR_CRL_INIT        = -3230, // LpkCrl 未初期化
LPK_ERR_OCRL_CHECK      = -3231, // LpkCrl(OpenSSL)確認エラー
// LpkOcsp
LPK_ERR_OCSP_INIT       = -3240, // LpkOcsp 未初期化
LPK_ERR_OOCSP_STATUS    = -3241, // LpkOcsp(OpenSSL)確認エラー
LPK_ERR_OOCSP2_STATUS   = -3242, // LpkOcsp(LeBerXml)確認エラー
// LpkCades
LPK_ERR_CADES_INIT      = -3250, // LpkCades 未初期化
LPK_ERR_CADES_ADDTS     = -3251, // LpkCades タイムスタンプ追加エラー
LPK_ERR_CADES_SIGN      = -3252, // LpkCades 署名エラー
LPK_ERR_CADES_VERIFY    = -3253, // LpkCades 検証エラー
LPK_ERR_CADES_HASH      = -3254, // LpkCades ハッシュ方式エラー
LPK_ERR_CADES_SETTS     = -3255, // LpkCades タイムスタンプセットエラー
LPK_ERR_CADES_SETBIN    = -3256, // LpkCades バイナリセットエラー
LPK_ERR_CADES_SETREVO   = -3257, // LpkCades 失効情報セットエラー
// LpkPkcs7
LPK_ERR_PKCS7_INIT      = -3270, // LpkPkcs7 未初期化
LPK_ERR_PKCS7_ADDTS     = -3271, // LpkPkcs7 タイムスタンプ追加エラー
```

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

```
LPK_ERR_OPKCS7_SIGN      = -3272, // LpkPkcs7(OpenSSL)署名エラー
LPK_ERR_OPKCS7_VERIFY    = -3273, // LpkPkcs7(OpenSSL)検証エラー
LPK_ERR_OPKCS7_HASH      = -3274, // LpkPkcs7(OpenSSL)ハッシュ方式エラー
LPK_ERR_OPKCS7_SETTS     = -3275, // LpkPkcs7(OpenSSL)タイムスタンプセットエラー
LPK_ERR_OPKCS7_SETBIN    = -3276, // LpkPkcs7(OpenSSL)PKCS#7 バイナリセットエラー
// LpkTimestampToken
LPK_ERR_TST_INIT         = -3290, // LpkTimestampToken 未初期化
LPK_ERR_OTST_INIT        = -3291, // LpkTimestampToken(OpenSSL)未初期化
LPK_ERR_OTST_MSGIMPL     = -3292, // LpkTimestampToken(OpenSSL)ハッシュ値取得エラー
// LpkTimestamp
LPK_ERR_TS_PARSE         = -3300, // LpkTimestamp 結果解析エラー
LPK_ERR_TS_SET           = -3301, // LpkTimestamp 設定エラー
// LpkTsAmano
LPK_ERR_ATS_SET          = -3320, // LpkTsAmano 設定エラー
// LpkUtil
LPK_ERR_UTIL_GETCRL      = -3350, // LpkUtil の CRL 取得エラー
LPK_ERR_UTIL_GETOCSP     = -3351, // LpkUtil の OCSP 取得エラー
LPK_ERR_UTIL_GETCERT     = -3352, // LpkUtil の証明書取得エラー
LPK_ERR_UTIL_GETCVS      = -3353, // LpkUtil の CVS 取得エラー
LPK_ERR_UTIL_GETOCSP2    = -3354, // LpkUtil の独自 OCSP 取得エラー
LPK_ERR_UTIL_HTTP        = -3355, // HTTP 通信エラー
// LpkCrypto
LPK_ERR_HASH_INIT        = -3400, // ハッシュ初期化エラー
LPK_ERR_HASH_ERROR       = -3401, // ハッシュエラー
LPK_ERR_ENC_INIT         = -3410, // 共通鍵暗号初期化エラー
LPK_ERR_ENC_ERROR        = -3411, // 共通鍵暗号エラー
LPK_ERR_PKEY_INIT        = -3420, // 公開鍵暗号初期化エラー
LPK_ERR_PKEY_ERROR       = -3421, // 公開鍵暗号エラー
LPK_ERR_PKEY_GET         = -3422, // 公開鍵暗号利用エラー
LPK_ERR_SIGN_ALG         = -3430, // 不明な公開鍵暗号アルゴリズム
// LeBerXml 用確保
// -3600 ～ -3699 の定義は LeBerXml/LeBerXml.h の中で定義
// XML
LPK_ERR_PKI_XML_LOAD     = -3700, // LePKI の XML 読み込みエラー
LPK_ERR_PKI_XML_SAVE     = -3701, // LePKI の XML 書き込みエラー
LPK_ERR_CERT_XML_LOAD    = -3710, // LpkCert の XML 読み込みエラー
LPK_ERR_CERT_XML_SAVE    = -3711, // LpkCert の XML 書き込みエラー
LPK_ERR_TS_XML_LOAD      = -3720, // LpkTimestamp の XML 読み込みエラー
LPK_ERR_TS_XML_SAVE      = -3721, // LpkTimestamp の XML 書き込みエラー
// JNI
LPK_ERR_JNI_ARGUMENT     = -3800, // LePKI の JNI 引数エラー
LPK_ERR_JNI_INIT         = -3801, // LePKI の JNI 初期化エラー
LPK_ERR_JNI_EXEC         = -3802, // LePKI の JNI の API エラー
// .NET
LPK_ERR_DOTNET_ARGUMENT  = -3810, // LePKI の.NET 引数エラー
LPK_ERR_DOTNET_INIT      = -3811, // LePKI の.NET 初期化エラー
LPK_ERR_DOTNET_EXEC      = -3812, // LePKI の.NET の API エラー
// その他
LPK_ERR_NO_SUPPORTED     = -3990, // 現在未サポートの機能が使われた
LPK_ERR_SYSTEM           = -3997, // システムエラー（インスタンス生成失敗等）
LPK_ERR_EXCEPTION        = -3998, // 不明例外発生
LPK_ERR_UNKNOWN          = -3999, // 不明エラー
```

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

Ａ．４．LeXAdES3 エラーコード（include/LeXAdES3/LexError.h）-5000 ～ -5999

LeXAdES3 のエラーコードは XML 電子署名に関するエラーです。

```
XDA_NO_ERROR                =     0, // 正常終了（ゼロ保証）
// 基本エラー
XDA_ERR_ARGUMENT            = -5000, // 引数異常
XDA_ERR_NOT_INIT            = -5001, // 未初期化
XDA_ERR_NO_SIGNED           = -5002, // 未署名だった
XDA_ERR_NO_XADES            = -5003, // XAdES では無く XML 署名だった
XDA_ERR_NO_VERIFIED         = -5004, // 未検証だった
XDA_ERR_PTR_NULL            = -5005, // 想定外のポインタエラー
XDA_ERR_FILE_OPEN           = -5010, // ファイルが開けない
XDA_ERR_FILE_EMPTY          = -5011, // ファイルが空だった
XDA_ERR_FILE_PATH           = -5012, // ファイルパス生成エラー
XDA_ERR_HASH_ALG            = -5020, // 未サポートのハッシュアルゴリズム
XDA_ERR_HASH_CALC           = -5021, // ハッシュ計算エラー
XDA_ERR_SIGN_ALG            = -5022, // 未サポートの署名アルゴリズム
XDA_ERR_SIGN_CALC           = -5023, // 署名計算エラー
// libxml2
XDA_ERR_NO_XMLDOC           = -5100, // XML ドキュメントが未設定
XDA_ERR_PARSE_XML           = -5101, // XML 読み込みエラー
XDA_ERR_NEW_XMLDOC          = -5102, // XML ドキュメントの生成エラー
XDA_ERR_NEW_NODE            = -5103, // XML ノードの生成エラー
XDA_ERR_NEW_CONTENT         = -5104, // XML コンテンツの生成エラー
XDA_ERR_NEW_OBJECT          = -5105, // XML オブジェクトの生成エラー
XDA_ERR_NEW_ROOT            = -5106, // XML ドキュメントのルートエラー
XDA_ERR_SAVE_XMLDOC         = -5107, // XML ドキュメントの保存エラー
XDA_ERR_NULL_NODE           = -5108, // XML ノード異常
XDA_ERR_SCHEMA_PARSE        = -5120, // XML スキーマファイルエラー
XDA_ERR_SCHEMA_VALID        = -5121, // XML スキーマチェックエラー
XDA_ERR_SCHEMA              = -5122, // XML スキーマエラー(XML 異常)
XDA_ERR_INVALID_ROOT        = -5150, // XML ルート要素名エラー
XDA_ERR_INVALID_NAMESP      = -5151, // XML 名前空間エラー
XDA_ERR_NO_CHILD            = -5152, // XML 子要素が無いエラー
XDA_ERR_NO_VALUE            = -5153, // XML 値要素が無いエラー
XDA_ERR_INVALID_ELEMENT     = -5154, // XML 不正要素エラー
XDA_ERR_XPATH_ELEMENT       = -5155, // XML に XPath 指定要素が見つからない
XDA_ERR_XML_EXCEPTION       = -5199, // XML 操作時の例外
// LxdSignature
XDA_ERR_SIGNEDINFO          = -5200, // SignedInfo の取得エラー
XDA_ERR_SIGNVALUE           = -5201, // SignatureValue の取得エラー
XDA_ERR_KEYINFO             = -5202, // KeyInfo の取得エラー
XDA_ERR_NO_REF              = -5203, // Reference(参照)の取得エラー
XDA_ERR_NO_REF_URI          = -5204, // Reference の URI 取得エラー
XDA_ERR_NO_REF_DOC          = -5205, // Reference の参照先取得エラー
XDA_ERR_REF_DATA            = -5206, // Reference データの取得エラー
XDA_ERR_REF_FILE            = -5207, // Reference ファイルの取得エラー
XDA_ERR_REF_MANIFEST_ID     = -5208, // Reference する Manifest の Id 取得エラー
XDA_ERR_TRANSFORM           = -5220, // Transform(変換)指定エラー
XDA_ERR_C14N_TRANS          = -5221, // XML 正規化(C14N)変換読み込みエラー
XDA_ERR_XPATH_TRANS         = -5222, // XPath 変換読み込みエラー
```

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

```
XDA_ERR_BASE64_TRANS              = -5223, // Base64 変換読み込みエラー
XDA_ERR_UNKNOWN_TRANS             = -5224, // 不明変換読み込みエラー
XDA_ERR_INVALID_ALG               = -5250, // 未サポートの暗号アルゴリズム
XDA_ERR_NO_HASH_VALUE             = -5251, // ハッシュ値が未設定
XDA_ERR_NO_SIGN_KEY               = -5260, // 署名鍵が見つからない
XDA_ERR_NO_SIGN_CERT              = -5261, // 署名証明書が見つからない
XDA_ERR_SIGNER_EXPIRED            = -5262, // 署名証明書が有効期間内では無い
XDA_ERR_SIGNER_PATH               = -5263, // 署名証明書が認証パス取得エラー
XDA_ERR_KEYINFO_CERT              = -5264, // KeyInfo の証明書エラー
XDA_ERR_INVALID_SIGN              = -5265, // 署名値検証エラー
XDA_ERR_INVALID_HASH              = -5266, // ハッシュ値チェックエラー
XDA_ERR_SINGROOT_EXIST            = -5267, // Signature 要素が既に存在する(署名済み)
XDA_ERR_NO_TARGET                 = -5268, // Reference(参照)対象未セットエラー
XDA_ERR_INVALID_TARGET            = -5269, // Reference(参照)対象取得エラー
XDA_ERR_NO_SET_REFS               = -5270, // Reference 要素が見つからない
XDA_ERR_NO_SIGNVALUE              = -5271, // SignatureValue が見つからない(MAKE)
// 検証
XDA_ERR_VERIFY_NOT_VALID          = -5300, // 検証結果が VALID では無い
XDA_ERR_VERIFY_NO_SIGN            = -5301, // 検証結果が未署名だった
XDA_ERR_VERIFY_EMBED_LEVEL        = -5302, // 検証情報の追加エラー(追加不要等)
// XAdES3
XDA_ERR_INVALID_SIGN_NUM          = -5400, // 署名番号(snum)エラー
XDA_ERR_INVALID_ADD_LEVEL         = -5401, // 情報追加時の XAdES レベルエラー
XDA_ERR_INVALID_LEVEL_ELEMENT     = -5402, // 長期署名要素が正しい順序では無かった
XDA_ERR_XADES_OBJ_ERROR           = -5403, // XAdES オブジェクトエラー
XDA_ERR_INVALID_CERT              = -5404, // 証明書からの情報取得エラー
XDA_ERR_INVALID_TIMESTAMP         = -5405, // タイムスタンプトークン取得エラー
XDA_ERR_NO_SIGTS                  = -5410, // SignatureTimeStamp 取得エラー
XDA_ERR_NO_SIGTS_TOKEN            = -5411, // SignatureTimeStamp のトークン取得エラー
XDA_ERR_NO_CERT_SIGTS             = -5412, // SignatureTimeStamp 証明書取得エラー
XDA_ERR_INVALID_SIGTS             = -5413, // SignatureTimeStamp ハッシュ不一致
XDA_ERR_MULT_SIGTS                = -5414, // SignatureTimeStamp 重複エラー
XDA_ERR_SIGTS_TOKEN               = -5415, // SignatureTimeStamp 更新エラー
XDA_ERR_TOKEN_OCSP_NO_SUPPORTED   = -5416, // V1.3.2 の TimeStampToken に OCSP 追加は未サポート
XDA_ERR_NO_TOKEN                  = -5417, // TimeStampToken 更新エラー
XDA_ERR_NO_ARCTS                  = -5420, // ArchiveTimeStamp 取得エラー
XDA_ERR_NO_ARCTS_TOKEN            = -5421, // ArchiveTimeStamp のトークン取得エラー
XDA_ERR_NO_CERT_ARCTS             = -5422, // ArchiveTimeStamp 証明書取得エラー
XDA_ERR_INVALID_ARCTS             = -5423, // ArchiveTimeStamp ハッシュ不一致
XDA_ERR_ARCTS_TOKEN               = -5424, // ArchiveTimeStamp 更新エラー
XDA_ERR_INVALID_TS_VER            = -5425, // ArchiveTimeStamp バージョンエラー(V1.4.1 の後に
V1.3.2 の追加)
XDA_ERR_SIGNED_PROP_ID            = -5430, // SignedProperties 取得エラー
XDA_ERR_GET_NET_SIGTS             = -5440, // 署名タイムスタンプのネットからの取得エラー
XDA_ERR_GET_SIGTS_ID              = -5441, // 署名タイムスタンプ Id 取得エラー
XDA_ERR_GET_NET_ARCTS             = -5442, // アーカイブタイムスタンプのネットからの取得エラー
XDA_ERR_GET_ARCTS_ID              = -5443, // アーカイブタイムスタンプ Id 取得エラー
XDA_ERR_GET_BASE64                = -5444, // バイナリ要素の Base64 変換エラー
XDA_ERR_GET_XADES_OBJ             = -5445, // XAdES Object 取得エラー
XDA_ERR_GET_XADES_XML             = -5446, // XAdES XML 取得エラー
// Reference
XDA_ERR_REF_INVALID_TYPE          = -5500, // 不正な Rerefence 種別
// XML
```

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

```
XDA_ERR_VERIFYXML_XML_LOAD      = -5750, // XdaVerifyXml の XML 読み込みエラー
XDA_ERR_VERIFYXML_XML_INIT      = -5751, // XdaVerifyXml の XML 未初期化エラー
XDA_ERR_VERIFYXML_XML_STAUS     = -5752, // XdaVerifyXml の XML 結果種別エラー
// JNI
XDA_ERR_JNI_ARGUMENT            = -5800, // LeXAdES3 の JNI 引数エラー
XDA_ERR_JNI_INIT                = -5801, // LeXAdES3 の JNI 初期化エラー
XDA_ERR_JNI_EXEC                = -5802, // LeXAdES3 の JNI の API 実行エラー
// .NET
XDA_ERR_DOTNET_ARGUMENT         = -5810, // LeXAdES3 の.NET 引数エラー
XDA_ERR_DOTNET_INIT             = -5811, // LeXAdES3 の.NET 初期化エラー
XDA_ERR_DOTNET_EXEC             = -5812, // LeXAdES3 の.NET の API 実行エラー
// その他
XDA_ERR_NO_SUPPORTED            = -5990, // 現在未サポートの機能が使われた
XDA_ERR_EVALUATION              = -5992, // 評価版の制限エラー
XDA_ERR_SYSTEM                  = -5997, // システムエラー（インスタンス生成失敗等）
XDA_ERR_EXCEPTION               = -5998, // 不明例外発生
XDA_ERR_UNKNOWN                 = -5999, // 不明エラー
```

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

付録Ｂ．適合宣言（Declaration of Conformity）

Required Level:　M=Mandatory（必須）、O=Optional（任意）、C=Conditional（条件付き）

Ｂ．１．Version number of ETSI EN 319 132-2 to be referenced

LE:XAdES:Lib V3

Ｂ．２．Scope of profile implementation

B.2.1 profile implementation

Profile identifier	Creator	Validator
XAdES-T	✔	✔
XAdES-A	✔	✔

Ｂ．３．Conformity to the XAdES-T profile

B.3.1 Signature Element

Element	Required level	Creator	Validator
Id attribute	M (NOTE 1)	✔	✔
ds:SignedInfo	M	✔	✔
ds:CanonicalizationMethod	M	✔	✔
ds:SignatureMethod	M	✔	✔
ds:Reference	M	✔	✔
ds:Transforms	O	✔ ※1	✔ ※1
ds:DigestMethod	M	✔	✔
ds:DigestValue	M	✔	✔
ds:SignatureValue	M	✔	✔
ds:KeyInfo	O (NOTE 2)	✔	✔
ds:Object	M	✔	✔
NOTE 1: XmlDsig ではオプションだが XAdES では必須 NOTE 2: KeyInfo を Reference に含めるか、SigningCertificateV2 で署名証明書参照のどちらかが必要			
※1: 対応 Transform は Base64・Enveloped・XPath・XML 正規化(C14N 1.0/1.1)			

B.3.2 Object element

Element	Required level	Creator	Validator
xs:QualifyingProperties	M	✔	✔
xs:SignedProperties	M	✔	✔
xs:UnsignedProperties	O	✔	✔
xs:QualifyingPropertiesReference	C	---	---

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

B.3.3 Singed Properties element

Element	Required level	Creator	Validator
xs:SignedSignatureProperties	M	✔	✔
xs:SigningTime	O (NOTE 1)	✔	✔
xs:SigningCertificateV2	O (NOTE 1, 2)	✔	✔
xs:SigningCertificate	C (NOTE 3)	✔	✔
xs:SignaturePolicyIdentifier	C	---	---
xs:SignatureProductionPlaceV2	C	---	---
xs:SignatureProductionPlace	C (NOTE 3)	---	---
xs:SignerRoleV2	C	---	---
xs:SignerRole	C (NOTE 3)	---	---
xs:SignedDataObjectProperties	C	✔	✔
xs:DataObjectFormat	C (NOTE 1)	✔	✔
xs:Description	C	✔	✔
xs:ObjectIdentifier	C	---	---
xs:MimeType	C (NOTE 1)	✔	✔
xs:Encoding	C	✔	✔
xs:CommitmentTypeIndication	C	---	---
xs:AllDataObjectsTimeStamp	C	---	---
xs:IndividualDataObjectsTimeStamp	C	---	---
NOTE 1: ETSI EN 319 132–1 v1.1.1 Baseline Signature の必須（M）項目 NOTE 2: KeyInfo を Reference に含めるか、SigningCertificateV2 で署名証明書参照のどちらかが必要 NOTE 3: これらの要素は ETSI TS 101 903 v1.1.1 との過去互換の為にのみ利用			

B.3.4 Unsigned properties

Element	Required level	Creator	Validator
xs:UnsignedSignatureProperties	M	✔	✔
xs:CounterSignature	O	---	---
Trusted time	M	✔	✔
xs:SignatureTimeStamp	O (NOTE 1)	✔	✔
xs:Time mark or other method	C	---	---
xs:UnsignedDataObjectProperties	C	---	---
NOTE 1: ETSI EN 319 132–1 v1.1.1 Baseline Signature の必須（M）項目			

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

Ｂ．４．Conformity to the XAdES-X-Long form

B.4.1 Additional Unsigned properties

Element	Required level	Creator	Validator
xs141:TimeStampValidationData	C	✔	✔
xs:CertificateValues	C	✔	✔
xs:RevocationValues	C	✔	✔
xs141:CompleteCertificateRefsV2	O	---	---
xs:CompleteCertificateRefs	C (NOTE 1)	---	---
xs:CompleteRevocationRefs	C (NOTE 1)	---	---
xs:CRLRef	O	---	---
xs:OCSPRef	O	---	---
xs:OtherRef	C	---	---
xs141:AttributeCetrificateRefsV2	O	---	---
xs:AttributeCetrificateRefs	C (NOTE 3)	---	---
xs:AttributeRevocationRefs	C (NOTE 3)	---	---
xs141:SigAndRefsTimeStampV2	C	---	---
not distributed case	M	---	---
distributed case	C	---	---
xs:SigAndRefsTimeStamp	C (NOTE 1)	---	---
xs141:RefsOnlyTimeStampV2	C	---	---
not distributed case	M	---	---
distributed case	C	---	---
xs:RefsOnlyTimeStamp	C	---	---
xs:CertificateValues	M	✔	✔
xs:EncapsulatedX509Certificate	O	✔	✔
xs:OtherCertificate	C	---	---
Certificates maintained by trusted service	C (NOTE 2)	---	---
xs:RevocationValues	M	✔	✔
xs:CRLValues	O	✔	✔
xs:OCSPValues	O	✔	✔
xs:OtherValues	C	---	---
Certificates maintained by trusted service	C (NOTE 2)	---	---
xs:AttrAuthoritiesCertValues	C (NOTE 3)	---	---
xs:AttributeRevocationValues	C (NOTE 3)	---	---
Any unsigned signature property defined in any other version of XAdES	C	---	---

NOTE 1: これらの要素は ETSI TS 101 903 v1.1.1 との過去互換の為にのみ利用

NOTE 2: 認証局または他の信頼できるサービスで保管されるなら証明書を置く必要は無い

NOTE 3: これらの要素は ETSI TS 101 903 v1.1.1 and 1.2.1 では未定義です

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

Ｂ．５．Conformity to the XAdES-A profile

B.5.1 Additional Unsigned properties

Element	Required level	Creator	Validator
Archiving	M (NOTE 1)	✔	✔
xs141:ArchiveTimeStamp	O	✔	✔
not distributed case	M	✔	✔
distributed case	C	---	---
Evidence Record	C (NOTE 2)	---	---
xs:ArchiveTimeStamp	C (NOTE 3)	✔	✔
Other method	C (NOTE 1)	---	---
xs141:TimeStampValidationData	C (NOTE 4)	✔	✔
xs:CertificateValues	C	✔	✔
xs:RevocationValues	C	✔	✔
xs141:RenewedDigests	C (NOTE 5)	---	---
Any unsigned signature property defined in any other version of XAdES	C	---	---

NOTE 1: ArchiveTimeStamp に代わる他の信頼できるサービスがあれば「他の方法」を適用可能
NOTE 2: IETF RFC 6283 で定義
NOTE 3: これらの要素は ETSI TS 101 903 v1.1.1 との過去互換の為にのみ利用
NOTE 4: タイムスタンプ用の検証データ保管でありタイムスタンプ毎に必要
NOTE 5: Manifest 要素にて利用

Ｂ．６．Specifications to be referenced by elements "Conditional"

B.6.1 Unsigned properties

No.	Element name	Referenced specification
1	無し	無し

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

付録Ｃ．制限事項・履歴

- V3.00 では現在以下の制限があります。

  1）数十メガバイトを超えるような大きな XML ファイルの利用は未対応です。
  → DOM の XML パーサー（libxml2）の制限です。
  2）Manifest のハッシュアルゴリズム危殆化対応の RenewedDigests には未対応です。
  3）複数署名のカウンター署名（CounterSignature）には未対応です。
  4）XAdES-C/XAdES-X の仕様には未対応です。
  5）その他未対応の仕様は「付録 B. 適合宣言」にて確認してください。

- その他バージョンアップの履歴に関してはルートディレクトリ下にある readme-LeXAdES3.txt に記載。

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

付録Ｄ．コピーライト表示

```
[OpenSSL License]
Copyright 2002-2020 The OpenSSL Project

Licensed under the Apache License, Version 2.0 (the “License”);
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is
distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and limitations under the License.

 [OpenLDAP License]
This product includes softwares developed by:
 Copyright 1999-2003 The OpenLDAP Foundation, Redwood City, California, USA.  All Rights Reserved.
 http://www.openldap.org/

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

 [iconv License]
This product includes softwares developed by:
 Copyright (C) 1999-2003 Free Software Foundation, Inc.
 http://www.gnu.org/licenses/lgpl.html

[zlib License]
This product includes softwares developed by:
 Copyright (C) 1995-2012 Jean-loup Gailly and Mark Adler.
 http://www.zlib.net/
```

※ license フォルダ下に詳細なコピーライトファイルあり。

◆ 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 https://www.langedge.jp/

以上

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