PDF Structure

Version 1.10.2

暫定版

株式会社トラスト・ソフトウェア・システム
2024 年 6 月

目次

1.0 はじめに

PDF Structure はPDF（Portable Document Format）文書に以下のような処理を施す機能をもっており、その機能をアプリケーションに追加するためのライブラリです。

- PDF文書を画像データに変換
- PDF文書にスタンプ（文字、画像、図形、など）を追加
- PDF文書のメタデータの読み出しや追加・変更
- PDF文書を構成するオブジェクトの抽出と追加
- PDFの文書情報（タイトル、レイヤ、など）の抽出
- PDF文書の署名検証と電子署名（タイムスタンプを含む）

本説明書はPDF文書の署名検証と電子署名（タイムスタンプ）機能について説明しています。他の機能につきましては、それぞれの機能説明書を参照してください。

PDF Structure では署名の検証において電子証明書の有効性や署名値の正当性（ハッシュ値などによって判定します）を判断しますが、その電子署名の有効性（署名検証の結果）はアプリケーションのポリシーに従って判断できるように多くのデータをPDF文書から取り出すようにしています。

PDF Structure によって生成されたPDF文書は増加更新された（Incremental Updated PDF）文書として生成されます。

注意：この文書に記載されたメソッドおよびプロパティの仕様は暫定的な仕様です。正式版ではその内容が変わる場合があります。

1.1 試用版について

PDF Structure ライブラリは試用のためのライセンスキーを指定すると、すべての機能を試すことができます。ただし、出力されたPDFなどに透かしが挿入されます。また、ライセンスキー文字を空白にすると一部の機能が制限されます。

これらの制限は正規のライセンスキーを使用することで解除されます。

2.0 利用環境および PDF のバージョン

PDF Structure は以下の環境で利用できます。

利用環境	Windows 8、8.1、10、11
	Windows Server 2016、2019、2022
開発環境	C/C++、C#

PDFのバージョン PDF 1.4 から PDF 1.7 およびPDF 2.0（その一部）を処理します。

PDF Structure はパスワードで暗号化されたPDF文書を処理の入力に指定できますが、出力されるPDF文書を暗号化できません。暗号化されたPDF文書はすべての処理前に復号されますので解像度や印刷の可能・不可能などの制限項目は作成されたPDF文書に反映されません。

PDF文書に追加できるテキスト文字列のフォントは「True Type 形式」または「Compact Font 形式」のOpenTypeフォントファイルです。抽出または追加できる画像の形式は、PNG形式、JPEG形式、TIFF形式、BMP形式です。

3.0 インストール

PDF Structure には以下のフォルダおよびファイルが含まれます。

doc	「PDF Structure 説明書」および「使用許諾契約書」
include	C/C++で使用するためのヘッダファイル、他
lib	ライブラリ群
sample	C/C++、C#（Visual Studio 2019プロジェクト）サンプル コードなど

開発環境に応じて適切なフォルダにコピーして使用してください。

3.1 ファイルの概要

PDF Structure に含まれるファイルの概要です。

doc/PdfStructure 説明書.pdf	「PDF Structure 説明」（本書）
doc/使用許諾契約書 第1版.pdf	「PDF Structure 使用許諾契約書」
lib/x64/PdfStructure.dll	PDF文書を解析・変更するためのライブラリ（64ビット環境用）
lib/win32/PdfStructure.dll	PDF文書を解析・変更するためのライブラリ（32ビット環境用）
lib/x64/PdfStructure.lib	C/C++開発環境用のリンクライブラリ（64ビット環境用）
lib/win32/PdfStructure.lib	C/C++開発環境用のリンクライブラリ（32ビット環境用）
lib/PdfStructureNET.dll	C#用ラッパーDLL
include/PdfStructure.h	C/C++用のヘッダファイル
sample/.NET	C#言語のサンプルコード サンプルプロジェクトは適切なネイティブライブラリを使用するように最適化されています。
sample/C	C 言語のサンプルコード
sample/init.xml	PDF Structure の初期設定を変更する初期化ファイルの例

注意：

ネイティブ DLL（PdfStructure.dll）は32ビット環境用および64ビット環境用にそれぞれ最適化されています。必ず開発・利用環境に合ったDLLを使用してください。

また、開発時のプラットフォームターゲットは利用環境に必ず合致させてください。

3.2 C/C++ インターフェース

ネイティブ C/C++ 開発環境ではヘッダファイル（PDFStructure.h）を使い、ライブラリ（PdfStructure.lib）をリンクしてください。実行時には PdfStructure.dll を参照できるようにしてください。

3.3 .NET インターフェース

PDFライブラリ(PdfStructure.dll)は.NETアセンブリではありません。C#からスタンプ付加機能を使うための.NETアセンブリDLL(PdfStructureNET.dll)を参照しなければなりません。これらのDLLは、コンパイル・実行時において適切に参照できるようにしてください。

名前空間:

```
PDFTools.PDFStructure
PDFTools.PDFStructure.Security
PDFTools.PDFStructure.Security.Misc
```

以下の手順で Structure クラスのインスタンスを生成してください。

```
Structure structure = new Structure();
```

Security クラスのインスタンスは以下の手順で生成してください。

```
SecurityInterface security = structure.GetSecurityInterface();
```

4.0 Security 機能概要

Security機能は既存の電子署名（またはタイムスタンプ）された PDF 文書署名を検証したり、既存の PDF 文書に電子署名またはタイムスタンプを施したりするライブラリです。

Security機能は電子証明書の検証や署名値（ハッシュ値）の正しさを返値とします。それらの値はPDF Structureライブラリを使ったアプリケーションのポリシーに従って判断できる判定情報となります。

4.1 電子署名、タイムスタンプおよび署名検証の手順

必ず以下の手順に従って実施してください。

1. ライブラリの初期化

2. PDF文書のオープン

3. Securityインターフェースの取得
   既にオープンされたPDF文書に署名があれば自動的に検証されます。

4. 検証に必要なデータの取得
   署名データや電子証明書なのデータを取得できます。
   署名の正当性や証明書の有効性はライブラリで確認した結果を取得できます。
   これらの情報を使いアプリケーションのポリシーに従った署名検証を実施します。

5. 電子証明書や時刻認証局（Timestamp Authority）の指定

6. 出力ファイル名の指定
   電子署名が施されたPDFファイルはこの名称で出力されます。

7. Securityインターフェースを終了
   電子署名はこの時に実施されます。

8. PDFファイルをクローズ

9. ライブラリの終了

5.0 Structure メソッド・プロパティ

C#で利用するメソッドおよびプロパティです。
C/C++開発環境で利用する場合は include フォルダ内のヘッダファイルを参照してください。

5.1 初期化

PDF Structure ライブラリの使用に先立って初期化します。
なお、ライブラリは使用後に `Uninitialize` メソッドを使って開放しなければなりません。

```
メソッド: int Initialize(
              string    licenseKey
          );
```

引数

`licenseKey`　　PDF Structure を使用するためのライセンス文字列

戻り値

ライブラリ初期化に成功すると0（ゼロ）が戻ります。それ以外の場合はエラーでその値はエラーコードです。

5.16 ライブラリの終了

ライブラリの使用を終了します。

```
メソッド: void Uninitialize();
```

引数

ありません。

戻り値

ありません。

5.2 ライセンス情報の表示および取得

PDF Structure の初期化の後に、そのライセンスキーの内容を文字列で表示または取得します。
`ShowLicenseInfo` は結果をコンソールに出力し、`GetLicenseInfo` は結果の文字列を戻します。

```
メソッド: void ShowLicenseInfo();

メソッド: string GetLicenseInfoStr();
```

引数

ありません。

戻り値

`ShowLicenseInfo` は戻り値がありません。
`GetLicenseInfoStr` は成功するとライセンス情報文字列が戻ります。失敗すると `null` 文字が戻ります。

5.3 初期化ファイル（または初期化データ）を指定する

PDF Structure はPDF文書で指定されたフォントの代替などを初期化ファイルまたは初期化データで指定できます。初期化ファイル（初期化データ）はXML形式です。

初期化ファイル（初期化データ）でのフォントの代替やスタンプデータを指定します。

```
メソッド: int LoadInitFile(
              string    initFileName
          );

メソッド: int LoadInitData(
              string    data
          );

メソッド: int LoadInitData(
              string    data
              int       length
          );
```

引数

initFileName	初期化ファイルのパス名
data	初期化データ（XML 形式データ）
length	初期化データのバイト数

戻り値

初期化ファイルを読み取ると0（ゼロ）が戻ります。それ以外の場合はエラーコードが戻ります。

5.4 既存のPDFファイルをオープン

PDFファイル（またはデータ）をオープンします。

PDF Structure はパスワードで暗号化されたPDFファイル（またはデータ）をオープンしスタンプを付加できます。しかし、PDF Structure は出力されるPDF文書を暗号化しませんので元のPDF文書の暗号化や制限項目はすべて削除されます。

引数に指定されたパスワードはPDFファイルがパスワードで暗号化されている場合のみ使用し、暗号化されていない場合は無視されます。

```
メソッド: int OpenDoc(
              string    fileName,
              string    password,
          );

メソッド: int OpenMemDoc(
              byte[]    pdfData,
              uint      length
              string    password,
          );
```

引数

fileName	PDF のファイルパス
pdfData	バイト配列の PDF データ
length	PDF データのサイズ（バイト長）
password	暗号化に使用したパスワード

戻り値

成功すると0（ゼロ）が戻り、失敗した場合はエラーコードが戻ります。

5.5 現在のPDFデータをファイルに出力

現在のPDFデータをPDF文書としてファイルに出力します。

```
メソッド: int SavePDF(
              string    filePath,
          );
```

引数

filePath	出力するPDFファイルのパス名

戻り値

成功すると、0（ゼロ）が戻り、失敗するとエラーコードが戻ります。

5.6 文書のクローズ（PDF文書処理の終了）

オープンしたPDF文書の処理を終了します。

```
メソッド: void CloseDoc();
```

引数

ありません。

戻り値

ありません。

5.7 Security インターフェース取得

Security インターフェースを取得します。
インターフェースの取得に先立って、既存PDFファイルのオープンが必須です。
取得した Security インターフェースクラスは Security.SignCloseInterface()また Security.CloseInterface()はで終了させます。

```
メソッド： Security GetSecurityInterface();
```

引数

ありません。

戻り値

`Security` インターフェースのクラスが戻ります。
失敗すると、`null` ポインターが戻ります。

6.0 Securityインターフェース

Security インターフェースはオープンしたPDFファイルの署名を検証したり、電子署名を施したりするためのクラスです。

6.1 PDF文書増加更新(Incremantal Update)数

PDF文書が増加更新された数を返します。
更新数には論理的な更新(リニアライズなどでの更新)を含みません。

```
プロパティ(get)：  int DocumentHierarchyCount
```

値：増加更新数
　　値が 1 の場合はPDF文書が更新されていません。

6.2 PDF文書増加更新(Incremantal Update)情報

PDF文書が増加更新された各バージョンの情報を返します。
この情報には電子署名によって追加された様々なデータが含まれますので、この情報を使ってアプリケーションのポリシーに従い電子署名の正当性を判定できます。
更新数には論理的な更新(リニアライズなどでの更新)を含みません。

```
プロパティ(get)：  DocumentHierarchyInfo[] DocumentHierarachyInformation
```

値：DocumentHierarchyInfoクラスの配列
　　戻り値の要素数は、PDF文書の増加更新数です。

6.3 タイムスタンプ機関

タイムスタンプ機関のURLを指定します。
空白文字を指定すると、タイムスタンプは追加されません。

```
プロパティ(set)：  String TimestampAuthority
   既定値：  ""
```

値：タイムスタンプ機関のURI

6.4 電子証明書指定

電子証明書のサブジェクト名(RDN名)を指定します。
指定されたサブジェクト名を個人の証明書ストアで検索し、最初に検索された証明書で文書に署名します。

```
プロパティ(set)：  String SigneriCertificateSubject
   既定値：  ""
```

値：Windows 証明書ストア(個人)に格納された証明書を識別できるRDN名

6.5 出力ファイル名指定

生成されるPDFファイル名を指定します。
空白文字を指定すると、署名されたPDFファイルは作成されません（署名操作はキャンセルされます）。

```
プロパティ(set)：  String SigneriCertificateSubject
   既定値：  ""
```

値：署名されたPDFファイルとして格納されるファイルパス
空白またはnull文字を指定すると、PDFファイルは生成されません

6.6 電子署名してからSecureインターフェースを終了

オープンしたPDF文書に電子署名を施したのちに Security インターフェースを終了します。
証明書およびタイムスタンプ機関が指定されていない場合は、電子署名されずにインターフェースを終了します。

```
メソッド：int SignCloseInterface(
              String    filePath
          )
```

引数

filePath	生成される署名済みのPDFファイルのパス名 省略した場合は、電子署名がキャンセルされます。

戻り値

電子署名が成功しPDFファイルが作成されると0が戻ります。それ以外の場合はエラーコードが戻ります。

6.7 Secureを終了

Secureインターフェースを終了します。電子署名書またはタイムスタンプ機関が指定され生成されるPDFファイル名が指定されていても電子署名は実施されずPDFファイルも生成されません。

```
メソッド：void CloseInterface()
```

引数

ありません

7.0 DocumentHierarchy インターフェース

DocumentHierarchyインターフェースはPDF文書の増加更新されたバージョンごとの電子署名情報がSecurityインターフェース取得時点で格納されたクラスです。このクラスが配列の場合は、先頭の要素（インデックスが0の要素）がPDF文書のオリジナルバージョンです。最終の要素は最新バージョンで追加された情報です。

これらの情報はアプリケーションが自身のポリシーに従った署名を検証するために使用されます。

7.1 Document Security Store(DSS)の有無

対象バージョンにDocument Security Storeディクショナリが存在するかを示します。

このディクショナリには電子署名（タイムスタンプを含む）に使用された電子証明書のほかに、電子証明書の失効リスト（Certificate Revocation List[CRL]）および証明書ステータス・プロトコル（Certificate Status Protocol[OCSP]）レスポンスなどが含まれています。詳細は「ISO32000-2 12.8.4.3 Document Security Store」を参照してください。

```
プロパティ(get)：  bool has_DSS
```

値：

True	DSSディクショナリがあります。
False	DSSディクショナリがありません。

7.2 Validation-related Information(VRI)の有無

対象バージョンにValidation-related information ディクショナリが存在するかを示します。

このディクショナリには一つの電子署名（やタイムスタンプ）に関連する電子証明書や電子証明書の失効リスト（Certificate Revocation List[CRL]）および証明書ステータス・プロトコル（Certificate Status Protocol[OCSP]）レスポンスなどが含まれています。詳細は「ISO32000-2 12.8.4.4 Validation-related information」を参照してください。

```
プロパティ(get)：  bool has_VRI
```

値：

True	VRIディクショナリがあります。
False	VRIディクショナリがありません。

7.2 電子署名の有無とタイプ

対象バージョンの署名の有無とタイプを取得します。

署名のタイプは以下の3種類です。長期署名とは無関係です。

- ・電子署名
- ・副署名タイムスタンプが追加された電子署名
- ・ドキュメント タイムスタンプ

これらの値PDF文書の署名ディクショナリのType要素に従います。

署名ディクショナリの詳細は「ISO32000-2 12.8Digital Signature」を参照してください。

```
プロパティ(get)：  int has_SIG
```

値：

0	電子署名されていません。(type 要素または署名ディクショナリが未定義)
1	タイムスタンプ(type は DocTimeStamp)
2	電子署名(type は Sig)
3	副署名にタイムスタンプがある電子署名(type は Sig)

7.3 電子署名値の検証結果

対象バージョンの署名値を検証した結果を取得します。この結果は、電子証明書の有効性とは無関係ですので、署名に使用した証明書の有効性を別途確認しなければなりません。

```
プロパティ(get)：  bool sig_verified
```

値：

true	署名値は正当です。
fasle	署名値は不正です。

7.4 Page、Resource、Annotation 情報の有無

対象バージョンが署名情報と共に Page、Resource、Anotation 情報が含まれる場合は署名時にページにたいして何らかの追加または変更があることを示します。

```
プロパティ(get)：  bool has_page_elm
プロパティ(get)：  bool has_resource_elm
プロパティ(get)：  bool has_annotation_elm
```

値：

true	追加または変更されたオブジェクトが含まれています。
fasle	追加および変更はありません。

7.5 電子署名値の有効性

PDF文書の署名は、署名ディクショナリに記載された領域（ByteRange）が対象です。PDF Structure はこの領域の署名値を計算し、CMS（Crypt Message Syntax）署名の署名値（SignatureValue）と比較します。

hash_is_matchet_sign はCMS署名の署名値を比較し、hash_is_matchet_ts はタイムスタンプの対象であるハッシュ値の署名値を比較した結果です。

CMSおよびタイムスタンプの詳細は、RFC5652 および RFC3161 を参照してください。

```
プロパティ(get)：  bool hash_is_matchet_sign
プロパティ(get)：  bool hash_is_matchet_ts
```

値：

true	ふたつの値が同じ
fasle	値が違う

7.6 署名者証明書の失効状況

電子署名に使われている署名者証明書の失効状況を取得します。

```
プロパティ(get)：  int SignerCertIsNotRevoked
```

値：

0	未確認
2	失効している
3	失効していない

7.7 署名者証明書の現在時刻での有効性

電子署名に使われている署名者証明書の現在時刻（PCのシステム時刻）における有効性を取得します。

```
プロパティ(get)：  bool SignerCertIsCurrValid
```

値：

True	現在時刻で有効
False	現在時刻で無効

7.8 対象バージョンのデータサイズ

対象バージョンのデータサイズを取得します。

PDFファイルの先頭からこのサイズ分をバイトデータで抽出すると、体操バージョンのPDF文書となります。このサイズは、署名の領域（ByteRange）の正当性を確認するうえで重要な要素です。

```
プロパティ(get)：  long DataSize
```

値：

データのバイトサイズ

7.9 署名領域（Byte Range）

電子署名されたPDFファイルの署名対象となる領域値を取得します。

署名領域は ByteRange（四つの値をもつ配列）として記載されています。プロパティはその値です。

ByteRange 各データの詳細は「ISO32000-2 12.8 Digital signature」を参照してください。

署名領域はPDFファイルの全部（署名ディクショナリの“Contents”を除く）を対象にしなければなりません。しかし、それを検出するためにはPDFデータを読み解く以外にありません。そのため、不正な署名データを作成できてしまいます。この ByteRange データと、対象バージョンのデータサイズを比較することは署名の正当性を確認するうえで重要です。

```
プロパティ(get)：  long[] ByteRange
```

値：

[0]	PDFファイルの先頭からのオフセット値
[1]	オフセットからのバイト数
[2]	PDFファイルの先頭からのオフセット値
[1]	オフセットからのバイト数

7.10 署名オブジェクト

PDFファイルの電子署名ではその詳細はSignatureディクショナリに記載されます。このディクショナリの情報文字列で取得します。

```
プロパティ(get)：  string SigObj
```

値：

全Signatureディクショナリの文字列

7.11 署名タイトル

関連するAnnotディクショナリに記載された電子署名のタイトルを取得します。
詳細は、「ISO32000-2 12.5 Annotations」を参照してください。

```
プロパティ(get)：  string Title
```

値：
署名のタイトル

7.12 署名者の名前

署名ディクショナリに記載された署名者の名前を取得します。

```
プロパティ(get)：  string Name
```

値：
署名者の名前

7.13 署名時刻

署名ディクショナリに記載された署名時刻を取得します。
ISO32000-2 12.8Didital signature ではタイムスタンプや CMS 署名に署名時刻がある場合はこの時刻を無視するように定められています。

```
プロパティ(get)：  string M
```

値：
署名時刻

7.14 Filter 名

署名ディクショナリに記載された署名Filterを取得します。

```
プロパティ(get)：  string Filter
```

値：
Filter 名

7.15 SubFilter 名

署名ディクショナリに記載されたSubFilterを取得します。

```
プロパティ(get)：  string Filter
```

値：
SubFilter 名

7.16 署名値(Contents)

署名ディクショナリに記載された署名値(Contents 要素)を取得します。

```
プロパティ(get)：  byte[] Contents
```

値：

Contents 要素のバイナリデータ

7.17 ダイジェストアルゴリズム

CMS署名で使用されたダイジェストアルゴリズムを取得します。

```
プロパティ(get)：  string DigestAlgorithm
```

値：

ダイジェストアルゴリズムのオブジェクトID文字列

7.18 ProBuild情報

署名ディクショナリに記載されたPropBuild情報を取得します。

```
プロパティ(get)：  string Filter
```

値：

PropBuild情報

7.19 署名者の証明書情報

署名者証明書をエンコードされたバイナリデータで取得します。

電子証明書バイナリデータの取り扱いについてはヘルパーメソッド「8.x　XXXXX」または System.Security.Cryptography.X509Certificates.X509Certificate2 をご利用ください。

詳細は、「ISO32000-2 12.8.4.4Validation-related information」を参照してください。

```
プロパティ(get)：  byte[] SignerCert
```

値：

証明書のバイナリデータ

7.20 署名者証明書のシリアル番号

署名者証明書のシリアル番号をバイナリデータで取得します。

```
プロパティ(get)：  byte[] SignerCertSn
```

値：

証明書シリアル番号のバイナリデータ

7.21 署名者証明書のサブジェクトデータ

署名者証明書のサブジェクト名をエンコードされたバイナリデータで取得します。
エンコードされたサブジェクト名は System.Security.Cryptography.X509Certificates.X509DistinguishedName などでデコードできます。

```
プロパティ(get)：  byte[] SignerCertSubj
```

値：
証明書サブジェクト名のエンコードされたバイナリデータ

7.22 署名者証明書の発行者名

署名者証明書の発行者名をエンコードされたバイナリデータで取得します。
エンコードされた発行者名は System.Security.Cryptography.X509Certificates.X509DistinguishedName などでデコードできます。

```
プロパティ(get)：  byte[] IssuerName
```

値：
証明書発行者名のエンコードされたバイナリデータ

7.23 署名アルゴリズム ID

署名アルゴリズム ID のオブジェクト ID を取得します。

```
プロパティ(get)：  string SignatureAlgorithmId
```

値：
署名アルゴリズム ID のオブジェクト ID 文字列

7.24 署名者証明書の利用開始日時

署名者証明書の利用を開始できる日時を取得します。
電子証明書はこの時刻以前に使用してはいけません。

```
プロパティ(get)：  DateTime NotBefore
```

値：
電子証明書の利用を開始できる日時

7.25 署名者証明書の利用終了日時

署名者証明書の利用ができなくなる日時を取得します。
電子証明書はこの時刻以降で使用してはいけません。

```
プロパティ(get)：  DateTime NotAfter
```

値：
電子証明書の利用を終了する日時

7.26 電子証明書の不具合（エラー）

電子証明書の不正利用などの場合にエラーコードが設定されます。

電子署名や検証のメソッドは必ずしも電子証明書の不正があったことを返しません。そのため、ののポロパティの値をチェックすることは重要です。

```
プロパティ(get)：  int SignerCert_err
```

値：

0　　証明書に不正はありません

負数　　エラーコード

7.27 タイムスタンプ関連情報

署名内のタイムスタンプトークンの関連情報を取得します。

タイムスタンプトークン詳細は「RFC3161」を参照してください。

7.27.1 タイムスタンプ機関のポリシーID

タイムスタンプを発行した機関のポリシーID を取得します。

```
プロパティ(get)：  string TstPolicyId
```

値：

ポリシーオブジェクト ID 文字列

7.27.2 ハッシュ ID

タイムスタンプのハッシュアルゴリズムのオブジェクト ID を取得します。

```
プロパティ(get)：  string HashOid
```

値：

ハッシュアルゴリズムのオブジェクト ID 文字列

7.27.3 ハッシュ値

タイムスタンプのハッシュ値（MassageImprint 値）を取得します。

```
プロパティ(get)：  byte[] HashMessage
```

値：

ハッシュ値

7.27.4 生成日時

タイムスタンプが認証する日時を取得します。

```
プロパティ(get)：  string GenTime
プロパティ(get)：  DateTime GenTm
```

値：

認証される日時を表す文字列または DateTime 値

7.27.5 Accuracy（時刻精度）

タイムスタンプの Accuracy 値を取得します。
詳細は「RFC3161 2.4.2Response Format」を参照してください。

```
プロパティ(get)：  int AccuracySec
プロパティ(get)：  int AccuracyMilli
プロパティ(get)：  int AccuracyMicro
```

値：
精度値

7.27.6 タイムスタンプ機関名

タイムスタンプ機関の名前（GeneralName）を文字列で取得します。

```
プロパティ(get)：  string TsaName
```

値：
Tsa をデコードした文字列

7.28 VRI ディクショナリに記載された証明書

VRI（Validation-related information）ディクショナリに記載された電子署名情報を取得します。
このディクショナリには0個以上の証明書データが格納されます。
電子証明書バイナリデータの取り扱いについてはヘルパーメソッド「8.x XXXXX」をまたは System.Security.Cryptography.X509Certificates.X509Certificate2 ご利用ください。
詳細は、「ISO32000-2 12.8.4.4Validation-related information」を参照してください。

```
プロパティ(get)：  byte[][] VRI_Certs
```

値：
証明書のバイナリデータ

7.29 VRI ディクショナリに記載された CRL

VRI（Validation-related information）ディクショナリに記載された証明書失効リスト（Certificate Revocation List）を取得します。
このディクショナリには0個以上の証明書失効リストデータが格納されます。
電子証明書バイナリデータの取り扱いについてはヘルパーメソッド「8.x XXXXX」をご利用ください。
詳細は、「ISO32000-2 12.8.4.4Validation-related information」を参照してください。

```
プロパティ(get)：  byte[][] VRI_CRLs
```

値：
エンコードされた証明書失効リストデータ

7.30 VRI ディクショナリに記載された OCSP

VRI（Validation-related information）ディクショナリに記載された OCSP（Online Certificate Status Protocol）レスポンスを取得します。
このディクショナリには0個以上の OCSP レスポンスデータが格納されます。

詳細は、「ISO32000-2 12.8.4.4Validation-related information」を参照してください。

```
プロパティ(get)：  byte[][] VRI_OCSPs
```

値：
エンコードされたOCSPレスポンスデータ

7.31 CMS 署名に格納された証明書失効データ

署名ディクショナリのSubFilter値が“ETSI.CAdES.detached”以外の場合にCMS署名に埋め込まれた失効情報（CRLまたはOCSP）を取得します。
このディクショナリには0個以上の失効データが格納されます。
詳細は、「ISO32000-2 12.8.3.3.2Revocation of CMS-based signatures」を参照してください。

```
プロパティ(get)：  byte[][] cms_CRLs
プロパティ(get)：  byte[][] cms_OCSPs
```

値：
エンコードされた失効データ

7.32 検証における署名時刻の基準

電子署名時点での証明書の有効性を確認するためには。署名時刻を保証する情報がPDF文書に格納されていなければならない。ここでは、最も信頼性の高い時刻の種類をPDF文書から取得する。

```
プロパティ(get)：  SecKindOfSigningTime SigningTime_source
```

値：
SEC_SIGNING_TIME_SYS(0)　検証を実施したシステムの時刻
SEC_SIGNING_TIME_M(1)　PDF文書の署名ディクショナリに記載された時刻
SEC_GIGNING_TIME_SIGN(2)　CMS署名データに記載された時刻
SEC_SIGNING_TIME_TS(3)　タイムスタンプの時刻

7.33 署名時刻

最も信頼性の高い署名時刻を取得します。

```
プロパティ(get)：  DateTime SigningTime
```

値：
署名時刻

8.0 ヘルパーメソッド

電子署名データを扱うために有用なメソッドを用意してあります。
これらのメソッドは、Windows API をラップしたものです。詳細は MSDN を参照してください。

8.1 BynaryToString

バイト配列を書式化された文字列に変換します。
Windows API の `CryptBinaryToString` をラップしてあります。

```
メソッド: static string SignCloseInterface(
              byte[]           data,
              CryptStringFlag  flag,
          )
```

引数

`data`	バイト配列のデータ
`flag`	書式化の形式を指定するフラグ

戻り値

書式化された文字列が戻ります。

Windows API の `CryptUIDlgViewContext` をラップしてあります。

8.2 ViewCertificate

指定された証明書をダイアログで表示します。
Windows API の `CryptUIDlgViewContext` を証明書が表示されるようラップしてあります。

```
メソッド: static bool ViewCertificate(
              byte[]           data,
          )
```

引数

`data`	バイト配列の証明書データ

戻り値

`True`	成功
`False`	失敗

8.3 ViewCRL

指定されたCRL（Certificate Revocation List；失効リスト）をダイアログで表示します。
Windows API の `CryptUIDlgViewContext` をCRLが表示されるようラップしてあります。

```
メソッド: static bool ViewCertificate(
              byte[]           data,
          )
```

引数

`data`	バイト配列のCRLデータ

戻り値

`True`	成功
`False`	失敗

9.0 著作権