PDF Structure

Version 1.10.3

PDFメタデータ編集

C# C/C++ **開発環境編**

株式会社トラスト・ソフトウェア・システム

2026 年 3 月

目次

1.0 はじめに 1
2.0 利用環境および PDF のバージョンと画像フォーマット 1
2.1 開発環境と使用説明書 1
3.0 インストール 2
3.1 ファイルの概要 2
3.2 .NET インターフェース 2
3.3 C/C++ インターフェース 3
4.0 PDF 文書メタデータのプロパティ 4
4.1 メタデータへのプロパティ記載手順 5
4.1.1 C#開発環境での開始手順 5
4.1.2 C/C++開発環境での開始手順 6
5.0 関数 – .NET、C/C++開発環境 1
5.1 インスタンス化および初期化 1
5.2 ライセンス情報の表示または取得 2
5.3 入力 PDF ファイルを開く 2
5.3.1 PDF ファイルの許可フラグを無視して開く 2
5.3.2 PDF ファイルを指定モードで開く 3
5.4 PDF 文書の情報 3
5.4.1 PDF 文書のページ数取得 3
5.4.2 PDF 文書のパーミッション（許可）フラグ 4
5.4.3 PDF 文書の暗号化レビジョン 4
5.4.4 PDF 文書のメタデータ 5
5.5 メタデータ 5
5.6 ビュワーのタイトルバーにタイトルを表示 5
5.7 DisplayDocTitle フラグを削除 6
5.8 PDF 文書を格納 6
5.9 PDF 文書処理の終了 6
5.10 ライブラリの使用終了 7
6.0 XmpInterface インターフェース 8
6.1 XmpInterface の取得 8
6.2 XmpInterface の終了 8
6.3 メタデータを更新 9
6.4 XMP プロパティの有無および削除 9
6.5 Simple(単純)プロパティ 10
6.5.1 プロパティの有無 10

6.5.2 Simple の空プロパティの作成 10
6.5.3 プロパティ値の取得 11
6.5.4 プロパティ値を記載 11
6.5.5 プロパティの削除 11
6.6 Array(**配列**)**プロパティ** 12
6.6.1 Array プロパティまたは Array アイテムの有無 13
6.6.2 Array の空プロパティを作成 14
6.6.3 Array アイテム値の取得 14
6.6.4 Array プロパティへの要素を記載 15
6.6.5 アイテム数 15
6.6.6 プロパティまたはアイテムの削除 16
6.6.7 **Alternate Array**(代替配列)プロパティ 17
6.6.8 **Alternate** Array(代替配列)プロパティのアイテム読み書きおよび削除 18
6.7 Structure(**構造**)**プロパティ** 19
6.7.1 プロパティ、フィールドの有無 19
6.7.2 Structure の空プロパティを作成 20
6.7.3 フィールド値の取得 20
6.7.4 フィールド値の記載 21
6.7.5 フィールドの数 21
6.7.6 プロパティまたはフィールドの削除 21
6.8 PDF **文書のプロパティ(概要)が記載されるメタデータ** 22
6.8.1 文書の「タイトル(title)」 23
6.8.2 「タイトル(title)」の取得 24
6.8.3 「タイトル(title)」の記載 25
6.8.4 「タイトル(title)」の削除 26
6.8.5 文書の「作成者(creator)」 27
6.8.6 「作成者(creator)」の取得 28
6.8.7 「作成者(creator)」の記載 29
6.8.8 「作成者(creator)」の削除 29
6.8.9 文書の「説明(description)」 30
6.8.10 「説明(description)」の取得 31
6.8.11 「説明(description)」の記載 32
6.8.12 「説明(description)」の削除 33
6.8.13 文書の「キーワード(keywords)」 34
6.8.14 「キーワード(keywords)」の取得 35
6.8.15 「キーワード(keywords)」の記載 36
6.8.16 「キーワード(keywords)」の削除 37
6.8.17 文書の「著作権情報」 38

6.8.18 「著作権情報」の記載 39
6.9 **日付データ** 40
6.9.1 日付データを Simple プロパティから取得 41
6.9.2 日付データを Simple プロパティに記載 42
6.9.3 文字列の日付を日付データに変換 43
6.9.4 日付データを文字列の日付に変換 44
6.10 **独自名前空間** 45
6.11 XMP **データをダンプする** 45
6.12 **エラーの詳細** 45
7.0 **著作権** 46

1.0 はじめに

PDF Metadata は、PDF (Portable Document Format) 文書のメタデータを読み書きするライブラリです。メタデータの読み書きができるのはPDFデータに限ります。

PDF Metadata は PDF Structure の一部ですので、PDF文書の画像化、情報抽出、内容変更、押印機能などの機能も使用でいきますがそれぞれのライセンスが必要です。

2.0 利用環境および PDF のバージョンと画像フォーマット

PDF Metadata は、以下の環境で利用できます。

利用環境	Windows 10、11 Windows Server 2026、2019、2022、2025
開発環境	C#、C/C++、VB.NET
画像フォーマット	PNG (Portable Network Graphics)、JPEG (Joint Photographic Experts Group)、TIFF (Tagged Image File Format) 形式、BMP(Device Independent Bitmap)形式 TIFF形式は、非圧縮、Deflate圧縮、JPEG 圧縮、LZW 圧縮を生成します。 BMP形式は非圧縮 (WindowsまたはOS/2) を生成します。

PDFのバージョン PDF1.4 から PDF1.7 および PDF2.0 (全てではありません) を対象とします。

PDF Metadata は、パスワードで暗号化されたPDF文書のメタデータを読み書きできます (パスワードが必要です) が、暗号化したPDFファイルを作成しません。

2.1 開発環境と使用説明書

本書は、PDF 文書のページを画像に変換する関数 (メソッド) を C#および C/C++開発環境で利用するための説明書です。他の機能は以下の説明書を参照してください。

・PDF 文書のページを画像に変換する C#および C/C++開発環境編
・PDF 文書のメタデータを解析 C#および C/C++開発環境編 (本書)
・PDF 文書のプリミティブなオブジェクトを抽出 C#および C/C++開発環境編

3.0 インストール

PDF Metadata には、以下のフォルダーおよびファイルが含まれます。

doc	「PDF Metadata 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#/VB.NET、 C/C++ (Visual Studio プロジェクト) サンプル コード

開発環境に応じて適切なフォルダーでご利用ください。

なお、PDF Metadata を使用するためには、適切なライセンスキーが必要です。評価用のライセンスキーで全ての機能を確認できますが、出力したPDFファイルには透かしが追加されます。

3.1 ファイルの概要

PDF Metadata に含まれるファイルの概要です。

libs/x64/PdfStructure.dll	PDF画像変換のためのネイティブDLL(x64環境専用)です。
libs/win32/PdfStructure.dll	PDF画像変換のためのネイティブDLL(win32環境専用)です。
libs/x64/PdfStructure.lib	C/C++(x64)開発環境の場合にリンクして使用します。
libs/win32/PdfStructure.lib	C/C++(win32)開発環境の場合にリンクして使用します。
libs/StructureNet.dll	PDF画像変換機能を C# (または VB.NET) で利用するためのラッパーDLLです。
libs/ConstantsNet.dll	メタデータのための定数群 (C#用)
include/Structure.h	C/C++用のヘッダファイルです。C/C++での開発時に使用します。

3.2 .NET インターフェース

PDF画像変換ライブラリ (PdfStructure.dll) は、.NETアセンブリではありません。C#またはVB.NETから利用するための.NETアセンブリDLL (StructureNet.dll) を参照して画像変換します。開発時においてはStructureNet.dll を「参照設定」に追加する必要があります。

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。

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

ネイティブDLL (PdfStructure.dll) の32ビット用と64ビット用をサブフォルダ「Win32」と「x64」に配置できます。これらをサブフォルダに配置すると、.NETアセンブリDLL (StructureNet.dll) は利用環境に合った適切なネイティブDLLを動的に使用します。

名前空間:

```
PDFTools.PdfStructure
```

クラス名:

```
Structure
```

以下の手順で Structure をインスタンス化してください。

```
Structure stc = new Structure();
```

さらに、Primitive インターフェースを取得します。

```
PrimitiveInterface prm = stc.GetPrimitiveInterface
```

3.3 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（Structure.h）を利用できるようにし、ライブラリ（PdfStructure.lib）をリンクしてください。実行環境では PdfStructure.dll を適切なフォルダーに配置してください。

メソッドやプロパティと同じ機能の関数をC/C++開発環境で使用するために以下の接頭辞が追加されて用意されています。なお、各開発環境専用の関数やメソッドを用意している場合がありますので注意してください。

PDF Structure の機能	接頭辞
PDFの画像化機能 (PDF Imager-LP)	Mlp
PDFにスタンプ機能 (PDF Stamp)	Stm
PDFの編集(文字･画像･図形の追加)	Mod
PDFのメタデータ編集 (PDF Metadata)	Xmp
PDFから構成オブジェクト抽出	Stc
PDFの電子署名と検証 (PDF Sign)	Stc

4.0 PDF 文書メタデータのプロパティ

PDF文書のメタデータ [1]には一般的に以下のようなプロパティがあります。

PDF1.7 以前ではこれらのプロパティを文書情報 (Document Information) ディクショナリ [2]に記載していましたが、PDF2.0 ではその記載は非推奨となりました。しかしながら、PDF Metadata は互換性のために文書情報ディクショナリにも記載します。

- タイトル
- 作成者
- サブタイトル
- キーワード
- 作成日
- 更新日
- PDFを作成したアプリケーション

など

メタデータの例:

```
<?xpacket begin="Ôªø" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="PDF Metadata ver1.10.2">
   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
     <rdf:Description rdf:about="" xmlns:xmp="http://ns.adobe.com/xap/1.0/">
     <xmp:CreateDate>2026-03-14T12:42:11+01:00</xmp:CreateDate>
     <xmp:ModifyDate>2026-09-24T21:23:03+02:00</xmp:ModifyDate>
     <xmp:CreatorTool>My Word Processor v10.7</xmp:CreatorTool>
     <xmp:MetadataDate>2026-09-24T21:23:03+02:00</xmp:MetadataDate>
   </rdf:Description>
   <rdf:Description rdf:about="" xmlns:pdf="http://ns.adobe.com/pdf/1.3/">
      <pdf:Producer>PDF Structure ver1.10.2</pdf:Producer>
   </rdf:Description>
   <rdf:Description rdf:about="" xmlns:dc="http://purl.org/dc/elements/1.1/">
      <dc:format>application/pdf</dc:format>
         <dc:title>
            <rdf:Alt>
               <rdf:li xml:lang="x-default">Annual report 2026</rdf:li>
               <rdf:li xml:lang="en">Annual report 2026</rdf:li>
               <rdf:li xml:lang="de">Jahresbericht 2026</rdf:li>
            </rdf:Alt>
         </dc:title>
         <dc:creator>
            <rdf:Seq>
               <rdf:li>John Doe</rdf:li>
               <rdf:li>Mary Miller</rdf:li>
            </rdf:Seq>
         </dc:creator>
      </rdf:Description>
   </rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>
```

PDF Metadata はこれらすべてのデータの読み書きができ、さらに独自の項目を追加しその項目の読み書きもできます (「6.6 PDF文書のプロパティ (概要) が記載されるメタデータ」参照)。以下では、一般的なプロパティの追加や変更の手順を説明します。メタデータの読み取りは XmpInterface (「6.0 XmpInterface」参照) を使って読み取ります。

[1] ISO32000-2:2020 PDF 2.0 「14.3 Metadata」参照
[2] ISO32000-2:2020 PDF 2.0 「14.3.3 Document information dictionary」参照

4.1 メタデータへのプロパティ記載手順

4.1.1 C#開発環境での開始手順

メタデータにプロパティを記載するには、PDF文書オープンしてから XmpInterface を介して実行します。
以下に手順を示します。

```
//初期化
using (var stc = new Structure("ライセンスキー"))
{
    //ファイルオープン
    stc.OpenDoc("input.pdf")

    //XMPインターフェース取得
    XmpInterface xmp = stc.GetXmpInterface();

    /*プロパティ書き込みや変更の処理*/
    ...;

    //PDF文書更新
    xmp.UpdateDocument();

    //XMPインターフェース終了
    xmp.CloseInterface();

    //PDF文書出力
    stc.SavePDF("out.pdf");
}
```

注意:

PDF文書更新(UpdateDocument メソッド)を実行すると、それまでの書き込みや変更がPDF文書に反映されます。これらの処理をキャンする場合は、このメソッドを実行せずにXMPインターフェースを終了します。

4.1.2 C/C++開発環境での開始手順

メタデータにプロパティを記載するには、PDF文書オープンしてから XmpInterface を介して実行します。
以下に手順を示します。

```
//初期化
MlpInitialize("ライセンスキー");

//ファイルオープン
MlpOpenDoc("input.pdf", NULL);

//XMPインターフェース取得
XMP_HANDLE h = MlpGetXmpInterface();

/*プロパティ書き込みや変更の処理*/
...;

//PDF文書更新
XmpUpdateDocument(h);

//XMPインターフェース終了
XmpCloseInterface(h);

//PDF文書出力
MlpSavePDF(_T("out.pdf"));

//ファイルクローズ（省略可能）
MlpCloseDoc();

//PDF Metadata 終了
MlpUninitialize();
```

注意：

PDF文書更新(XmpUpdateDocument 関数)を実行すると、それまでの書き込みや変更がPDF文書に反映されます。これらの処理をキャンする場合は、この関数を実行せずにXMPインターフェースを終了します。

5.0 関数 – .NET、C/C++開発環境

C#/VB.NET、およびC/C++で利用する関数です。C/C++で利用する場合は、機能に則した接頭辞がついていますので各関数名を読み替えてください。接頭辞は「3.3 C/C++インターフェース」を参照してください。

5.1 インスタンス化および初期化

PDF Metadata ライブラリはインスタンス化およびライセンスキーを使った初期化が必要です。まず PDFTool.PdfStructure.Structure クラスをインスタンス化します。続いて、以下のメソッドで初期化します。ライブラリはその使用後に Uninitialize メソッドを使って開放します。

メソッド

```
int Initialize(string license)
void InitializeWException(string license)
```

引数

license	PDF Metadata 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）が戻る。それ以外はエラーコードです。
InitializeWException メソッドは失敗すると Exception をスローします。

使用例

```
using PDFTools.PdfStructure;

using (var stc = new Structure())
{
  try
  {
     stc.InitializeWException("ライセンスキー文字列");
  }
  catch (Exception e)
  {
     Console.WriteLine(e.Message);
  }
  ... // ここで変換などを実施
  stc.Uninitialize();
}
```

5.2 ライセンス情報の表示または取得

PDF Metadata の初期化の後はライセンスキー内容を文字列で表示または取得ができます。
ShowLicenseInfo は結果をコンソールに出力し、LicenseInfoStr は結果を文字列で戻します。

メソッド
```
void ShowLicenseInfo()
void LicenseInfoStr()
```

引数

なし

プロパティ (get)
```
string LicenseInfoString
```

戻り値

ShowLicenseInfo は、戻り値がありません。
LicenseInfoStr は、成功するとライセンスを説明する文字列が戻ります。

5.3 入力 PDF ファイルを開く

PDF文書を開く際は、そのPDF文書が印刷する場合の解像度を低解像度に指定されている場合や、印刷そのものが禁止されている場合があります。そのような場合PDF文書はパスワードなどで暗号化されています。Metadata は、パスワードで暗号化されたPDF文書を復号して画像に変換できます。また、開くモードを「表示」や「印刷」に指定して適切に開くことができます。

PDF Metadata は、PDF文書以外に画像データを入力として開くことができます。入力画像はそのファイルの拡張子やデータの内容によって判別され適切に解釈されますが、PDFファイル以外ではメタデータを取り扱えません。

5.3.1 PDF ファイルの許可フラグを無視して開く

画像に変換するPDFファイルを開きます。
この関数はPDFファイルが暗号化されている場合にPDF文書に指定された許可フラグを無視します。なお、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します。

メソッド
```
int OpenDoc(string filename, string ownerPassword, string userPassword)
```

引数

filename	PDF のファイルパス
ownerPassword	所有者パスワード [3]
userPassword	ユーザーパスワード [4]

戻り値

0	成功
負数	エラーコード

[3] 所有者パスワードは「権限パスワード」と呼ばれることがあります。
[4] ユーザーパスワードは「開くパスワード」と呼ばれることがあります。

5.3.2 PDF ファイルを指定モードで開く

画像に変換するPDFファイルを開くモード（「表示」または「印刷」）に従って開きます。

この関数はPDFファイルが暗号化されている場合にPDF文書に指定された許可フラグに従って開きます。そのため、印刷が許可されないPDFファイルを「印刷」モードで開くと失敗します。また、低解像度印刷指定の場合は失敗しませんので、適切に画像化解像度を指定するため許可フラグを確認する必要があります。（許可フラグは、「4.5.2 PDF文書のパーミッション（許可）フラグ」を参照してください。）

なお、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します。

メソッド

```
int OpenDocUsage(string filename, string password, int mode)
```

引数

filename	PDF文書のファイルパス名
password	パスワード オーナーパスワード、ユーザーパスワードのいずれかを指定
mode	1（表示）または、2（印刷）を指定

戻り値

0	成功
負数	エラーコード

5.4 PDF 文書の情報

PDF Metadata は、開いたPDF文書の情報を取得できます。

5.4.1 PDF 文書のページ数取得

現在開いているPDF文書の総ページ数を取得します。

メソッド

```
int PageCount()
```

引数

なし

戻り値

0または正数	成功
負数	エラーコード

5.4.2 PDF 文書のパーミッション (許可) フラグ

PDF文書には、その文書を開いたユーザーに変更や印刷の可否を指定するフラグがあります。PDF Metadata はこの値を現在開いているPDF文書からパーミッション (許可) フラグ値として取得します。このフラグを評価するには、このPDF文書の暗号化レビジョンも必要です。暗号化レビジョンを取得するのは「4.5.3PDF文書の暗号化レビジョン」を参照してください

メソッド

```
int GetDocumentInfo(DocumentOpt.PERMISSION_FLAG, out int flag)
```

引数

flag	パーミッション (許可) フラグ値

戻り値

0	成功
負数	エラーコード

パーミッションフラグの詳細は、「PDF Reference」 3.5.2 Standard Security Handler を参照してください。
パーミッションフラグ値は、PDF文書の「Standard Security Handler」ディクショナリ内のPキーに指定された整数値です。

5.4.3 PDF 文書の暗号化レビジョン

PDF文書が暗号化されている場合は、印刷の許可などのフラグ (パーミッションフラグ) を確認しなければならない場合があります。以下ではこのフラグの評価に必要な暗号化レビジョン [5]を取得します。(パーミッションフラグの取得は「4.5.2 PDF文書のパーミッションフラグ」を参照)

メソッド

```
int GetDocumentInfo(DocumentOpt.ENCRYPT_REVISION, out int rev)
```

引数

rev	暗号化のレビジョン番号

戻り値

0	成功
負数	エラーコード

暗号化レビジョンは、PDF文書の「Standard Security Handler ディクショナリ [6]」のRキーに指定された値です。

[5] ISO 32000-2 PDF2.0 「7.6.4 Standard Security Handler」参照
[6] ISO 32000-2 PDF2.0 「7.6.4.2 Standard encryption dictionary」参照

5.4.4 PDF 文書のメタデータ

PDF文書に記載されたメタデータ[7]をバイトデータまたは文字列で取得します

```
メソッド
  int GetMetadata(out byte[] metadata)
  int GetMetadataString(out string metadataString)
```

引数

metadata	バイト列のメタデータ
metadataString	Unicode に変換されたメタデータ

戻り値

0 または正数	バイトサイズまたは文字数
負数	エラーコード

5.5 メタデータ

PDF文書のメタデータをXMP (Extensible Metadata Platform) として解析および変更します。
メタデータに記載されたプロパティの取得および変更は XmpInterface クラス (C#の場合)または XMP_TK_HANDLE を介して実行します。

```
C#開発環境
    Imr = new PdfImager();
    XmpInterface xmp = imr.GetXmpInterface();
```

```
C/C++開発環境
    XMP_TK_HANDLE handle = imr.GetXmpInterface();
```

戻り値

0 はエラーです。それ以外は値が取得できていますので、取得や編集の処理を行えます。

5.6 ビュワーのタイトルバーにタイトルを表示

PDFビュワーのタイトルバーにメタデータの dc:title (文書タイトル) の文字列を表示するように指定します。
文書タイトルが指定されていない場合はファイル名が表示される指定です。

```
メソッド
  int SetDisplayTitleFlag(bool flag)
```

引数

flag	True：メタデータの dc:title に指定された文字列をタイトルバーに表示 ただし、dc:title の指定が無い場合はファイル名を表示 False: タイトルバーにファイル名を表示

戻り値

0	成功
負数	エラー

[7] ISO 32000-2 PDF2.0 「14.3 Metadata」および ISO16684-1 XMP 参照

5.7 DisplayDocTitle フラグを削除

メタデータの dc:title (文書タイトル) に指定された文字列をPDFビュワーのタイトルバーに表示するための、Viewer Preferences[8]ディクショナリの DisplayDocTitle キーを削除します。

メソッド

```
int DeleteDisplayTitleFlag()
```

引数

なし

戻り値

0	成功
負数	エラー

5.8 PDF 文書を格納

指定されたパス名で PDF データを格納します。

このメソッドを XmpInterface 内で実行すると、メタデータが強制的に更新 (「6.3 メタデータを更新」参照) され、それまでのメタデータ変更をキャンセルできなくなりますので、注意してください。

メソッド

```
int SavePDF(string file)
```

引数

file	ファイルパス名

戻り値

0	成功
負数	エラー

5.9 PDF 文書処理の終了

PDF文書の画像変換処理を終了します。

メソッド

```
void CloseDoc()
```

引数

なし

戻り値

ありません。

[8] ISO 32000-2 PDF2.0 「12.2 Viewer preferences」参照

5.10 ライブラリの使用終了

ライブラリの使用を終了します。

メソッド

```
void Uninitialize()
```

引数

なし

戻り値

ありません。

6.0 XmpInterface インターフェース

ＰＤＦ文書のメタデータはＸＭＬ（Extensible Markup Language）で記載され、その内容はＸＭＰ（Extensible Metadata Platform）に従って記載されることが推奨されています。
PDF Metadata このようなメタデータを解析または変更します。

XMPメタデータは XmpInterface クラスを介して解析および変更します。
XMP規格は、ISO 16684-1 XMP を参照してください。

6.1 XmpInterface の取得

C#開発環境
メソッド

```
XmpInterface GetXmpInterface()
```

C/C++開発環境
関数

```
XMP_TK_HANDLE MlpGetXmpInterface()
```

引数

なし

戻り値

０	エラー
０以外	成功

6.2 XmpInterface の終了

XmpInterface は現在文書をクローズすると終了しますが、以下の手順でも終了させることができます。

メソッド

```
void CloseInterface()
void CloseInterface(bool flag)
```

引数

flag	True の場合はそれまでに変更されたメタデータ PDF データに追加してから終了します。False の場合はそれまでの変更をキャンセルして終了します。 XmpInterface 実行中に UpdateDocument () を実行すると、それまでの操作が確定し、キャンセルできません。 既定値： True

戻り値

ありません。

6.3 メタデータを更新

PDF文書のメタデータを変更された XMP データで更新します。

更新されたPDFファイルを作成するためにはこの処理が必須でが、PDFを出力する SavePDF()メソッドはこの更新処理を含みます。そのため、PDF データの出力前に変更されたメタデータの編集をキャンセルできません。

更新時には以下のように、メタデータに記載された文書情報が PDF 文書の Info ディクショナリにコピーされます。

- dc:title に記載された先頭データが Info ディクショナリの Title キーに記載されます。
- dc:creator に記載された先頭データが Info ディクショナリの Author キーに記載されます。
- dc:description に記載された先頭データが Info ディクショナリの Subject キーに記載されます。
- dc:subject に記載されたデータを結合して pdf:Keyword および Info ディクショナリの Keyword キーに記載されます。

メソッド

```
int UpdateDocument()
```

引数

なし

戻り値

0	成功
負数	エラーコード（指定された名前空間URIが不正な場合など）

6.4 XMP プロパティの有無および削除

この関数は近い将来に廃止されます。

Simple（単純）プロパティ、Array（配列）プロパティ、Structure（構造）プロパティに検定されたメソッドを利用してください。「6.5Simple プロパティ」、「6.6Array プロパティ」、「6.7Structure プロパティ」を参照して下さい。

メソッド

```
int DoesPropertyExist(string uri, string property)
int DeleteProperty(string uri, string property)
```

引数

uri	名前空間 URI
property	プロパティ名

戻り値

0	正しく削除された、または property が存在しません
正数	property が存在します
負数	エラーコード（指定された名前空間URIが不正な場合など）

6.5 Simple(単純)プロパティ

Simpleプロパティは URI 以外のテキストを値に持つことができます。なお、要素にはネストされた XMP 要素を含められません。

以下は、Simpleプロパティの例です。

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMP Core 6.0.0">
   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <rdf:Description rdf:about=""
            xmlns:xmp="http://ns.adobe.com/xap/1.0/">
         <xmp:Rating>3</xmp:Rating>
         <xmp:CreateDate>2026-02-19T11:14:02+09:00</xmp:CreateDate>
      </rdf:Description>
   </rdf:RDF>
</x:xmpmeta>
```

Simple プロパティの詳細は ISO16684-1:2019 XMP 「7.5 Simple valued XMP properties」を参照してください。

6.5.1 プロパティの有無

指定されたプロパティがSimpleとして存在するかを確認します。プロパティの値の有無は無関係です。

メソッド

```
int DoesSimplePropertyExist(string namespaceUri, string propertyName)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名

戻り値

0	名前空間またはプロパティが存在しない。もしくは、プロパティは存在するがSimpleプロパティではありません。
正数	Simpleプロパティが存在します（値が空の場合も含む）。
負数	エラーコード（指定された名前空間URIが不正な場合など）

6.5.2 Simple の空プロパティの作成

Simpleの空プロパティを作成します。

メソッド

```
int CreateSimple(string namespaceUri, string propertyName)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名

戻り値

0	成功
負数	エラーコード

6.5.3 プロパティ値の取得

Simpleプロパティの値を取得または設定します。設定する場合にそのプロパティが存在しない場合は新たに追加されます。

メソッド

```
int GetSimpleProperty(string namespaceUri, string propertyName,
                                                    out string value)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
value	プロパティ値

戻り値

正数	プロパティが存在し、値を取得できた場合に文字列のバイト数
負数	エラーコード

6.5.4 プロパティ値を記載

Simpleプロパティに値を記載します。記載する場合にそのプロパティが存在しない場合は新たに追加されます。

メソッド

```
int SetSimpleProperty(string namespaceUri, string propertyName,
                                                      string value)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
value	プロパティ値

戻り値

正数	プロパティが存在し、値を取得できた場合に文字列のバイト数
負数	エラーコード

6.5.5 プロパティの削除

Simpleプロパティを削除します。
プロパティがSimpleでない場合は失敗します。

メソッド

```
int DeleteSimpleProperty(string namespaceUri, string propertyName)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名

戻り値

0	成功
負数	エラーコード

6.6 Array(配列)プロパティ

Arrayプロパティはただ一つのネストされた要素を持ちます。ネストされた要素は以下のいずれかです。

- 順序の無い配列で、その要素は rdf:Bag です。
- 順序付けられた配列で、その要素は rdf:Seq です。
- 代替配列で、その要素は rdf:Alt です。

ネストされた要素には0個以上の rdf:li 要素が含まれます。

以下は順序の無いArrayプロパティの例です。

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMP Core 6.0.0">
   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <rdf:Description rdf:about=""
            xmlns:dc=”http://purl.org/dc/elements/1.1/”>
         <dc:UnorderedSubject>
            <rdf:Bag>
               <rdf:li>XMP</rdf:li>
               <rdf:li>metadata</rdf:li>
               <rdf:li>ISO standard</rdf:li>
               </rdf:Bag>
            </rdf:Bag>
         </dc:UnorderedSubject>
      </rdf:Description>
   </rdf:RDF>
</x:xmpmeta>
```

Array プロパティの詳細は ISO16684-1:2019 XMP 「7.7 Array valued XMP properties」を参照してください。

6.6.1 Array プロパティまたは Array アイテムの有無

指定されたプロパティが「Array（配列）プロパティ」として存在するか、またはアイテムが存在するかを確認します。値の有無は無関係です。

メソッド

```
int DoesArrayPropertyExist(string namespaceUri, string propertyName)
int DoesArrayItemExist(string namespaceUri, string propertyName,
                                                    int itemIndex)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
itemIndex	アイテムのインデックス番号（先頭の番号は１）

戻り値

０	Arrayプロパティではない、がプロパティArrayに存在しない、もしくはインデックス番号のアイテムがありません。
１	指定されたUnordered（順序無）またはOrdered（順序付）プロパティのいずれかで存在します。（プロパティの有無）
	そのプロパティの指定されたインデックスにアイテムが存在します。（アイテムの有無）
３	指定されたArrayプロパティが存在します。（プロパティの有無）
	そのプロパティの指定されたインデックスにアイテムが存在します。（アイテムの有無）
負数	エラーコード （指定された名前空間URIが不正な場合など）

6.6.2 Array の空プロパティを作成

指定されたプロパティでアイテムの無いArray（配列）プロパティを新たに作成します。作成する前に同名プロパティが存在していると、以前のプロパティのアイテムは削除されます。CreateUnorderedArrayProperty、CreateOrderedArrayProperty、CreateAlternateArrayProperty はそれぞれ順序無し、順序付き、代替のArrayを作成します。

メソッド

```
int CreateUnorderedArrayProperty(string namespaceUri,
                                 string propertyName)
int CreateOrderedArrayProperty(string namespaceUri,
                               string propertyName)
int CreateAlternateArrayProperty(string namespaceUri,
                                 string propertyName)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名

戻り値

0	成功
負数	エラーコード（指定された名前空間URIが不正な場合など）

6.6.3 Array アイテム値の取得

既存Array（配列）プロパティのアイテム値を取得します。
以下のメソッドはUnordered Array（順序無配列）およびOrdered Array（順序付配列）プロパティで使用できます。Alternate Array（代替配列）では「6.6.6 Alternateプロパティのアイテム読み書きおよび削除」を参照してください。

メソッド

```
string GetArrayItem(string namespaceUri, string propertyName,
                    int itemIndex)
int GetArrayItem(string namespaceUri, string propertyName,
                 out string value)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
itemIndex	アイテムのインデックス番号（先頭のアイテム番号は１）
value	アイテム値

戻り値

正数	プロパティが存在していて、取得した文字列のバイト数
0	プロパティが存在しているが値が空文字
文字列	プロパティがArrayの場合のプロパティ値
負数	エラーコード

6.6.4 Array プロパティへの要素を記載

Array（配列）プロパティの種類（順序有、順序無、代替）を変更せずにアイテム値を書き込みます。
以下のメソッドはUnordere Array（順序無配列）およびOrdered Array（順序付配列）プロパティで使用できます。Alternate Array（代替配列）では「6.6.5 Alternateプロパティのアイテム読み書き及び削除」を参照してください。
AppendArrayItem は最後のアイテムとして値と共に追加され、SetArrayItem は指定されたインデックス位置のアイテムの値を上書きします。InsertArrayItems は指定されたインデックス位置に新たなアイテムとして追加されます。これらの関数は既存Arrayへの記載となりますので、Arrayが存在しないと失敗します。

メソッド

```
int AppendArrayItem(string namespaceUri, string propertyName,
                                                        string value)
int SetArrayItems(string namespaceUri, string propertyName,
                                              int itemIndex, string value)
int InsertArrayItems(string namespaceUri, string propertyName,
                                              int itemIndex, string value)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
itemIndex	アイテムのインデックス番号（先頭のアイテム番号は１） アイテム数＋１とすると、Array の最後のアイテムとして追加される。
value	作成された Array プロパティのアイテム値

戻り値

０	成功
負数	エラーコード

6.6.5 アイテム数

Array（配列）プロパティのアイテム数を取得します。

メソッド

```
int CountArrayItems(string namespaceUri, string propertyName)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名

戻り値

正数	Array（配列）プロパティのアイテム数
負数	エラーコード

6.6.6 プロパティまたはアイテムの削除

Array (配列) プロパティまたはそのアイテムを削除します。
プロパティがArrayでない場合は失敗します。

メソッド

```
int DeleteArrayProperty(string namespaceUri, string propertyName)
int DeleteArrayItem(string namespaceUri, string propertyName,
                                                    int itemIndex)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
itemIndex	アイテムのインデックス番号 (先頭のアイテム番号は 1)

戻り値

0	成功
負数	エラーコード

6.6.7 Alternate Array(代替配列)プロパティ

Alternate Array (代替配列) プロパティには代替言語のアイテムを記載します。

PDF Metadata ではLanguage Alternative Array (言語代替配列) として言語の指定と共に読み書きおよび削除するメソッドがあります。言語は IETF RFC3066 に従ってアイテムの修飾子 xml:lang と共に記載されます。言語は"ja-JP"や"en-US","en-UK"などのようにマイナス文字 ("-";0x2D) で区切られた複数の部分で構成されます。この言語は"ja"や"en"などの小文字で表した主サブタグと、大文字2文字のサブタグをマイナス文字で区切って構成されます (この後にさらにサブタグが続く言語名もあります) 。なお、既定言語として"x-default"を指定できますが、これを指定されたアイテムはAlternative Arrayの先頭要素になります。

以下はAlternateプロパティの例です。

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMP Core 6.0.0">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about=""
        xmlns:dc="http://purl.org/dc/elements/1.1/">
      <dc:title>
        <rdf:Alt>
          <rdf:li xml:lang="x-default">文書タイトル</rdf:li>
          <rdf:li xml:lang="ja-JP">文書タイトル</rdf:li>
          <rdf:li xml:lang="en-US">Document Title</rdf:li>
        </rdf:Alt>
      </dc:title>
    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
```

6.6.8 Alternate Array(代替配列)プロパティのアイテム読み書きおよび削除

GetArrayAltLangText は指定言語のアイテム値を読み出します。SetArrayAltLangText では指定言語のアイテムがあればそれを更新しますが、指定言語がない場合は指定言語の新たなアイテムを追加します。DeleteArrayAltLangText は指定された言語のアイテムを削除します。アイテムの削除では部分マッチで複数のアイテムが検索された場合は削除されずエラーとなります。

各アイテム値が変更されるとAlternate Arrayの構成が適切に変更されます。

メソッド

```
int GetArrayAltLangText(string namespaceUri, string propertyName,
                                string genericLang, string specificLang,
                                 out string actualLang, out string value)

int SetArrayAltLangText(string namespaceUri, string propertyName,
                    string genericLang, string specificLang, string value)

int DeleteArrayAltLangText(string namespaceUri, string propertyName,
                                 string genericLang, string specificLang)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
genericLang	言語名の主サブタグ（言語名をマイナス文字で区切られた先頭の文字列）または完全な言語名 null または空文字を指定できます。
specificLang	完全言語名
actualLang	読み出したアイテムの言語名 null または空文字を指定できます。
value	アイテム値

戻り値

0	成功
負数	エラーコード

各要素は以下の手順に従って検索されます。

1. spesificLang で指定された完全言語名のアイテムを検索
2. genericLang が指定されている場合はそれと部分的にマッチする語名のアイテムを検索
3. x-default のアイテムを検索
4. 先頭のアイテムを選択

6.7 Structure(構造)プロパティ

構造プロパティの詳細は ISO16684-1:2019 XMP 「7.6 Structure valued XMP properties」を参照してください。

以下は Structure プロパティの例です。

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMP Core 6.0.0">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about=""
        xmlns:xmp="http://ns.adobe.com/xap/1.0/"
        xmlns:dc="http://purl.org/dc/elements/1.1/">
      <dc:StructureSubject rdf:parseType="Resource">
        <xmp:field1>構造化されたフィールド１</xmp:field1>
        <xmp:field2>構造化されたフィールド２</xmp:field2>
      </dc:StructureSubject>
    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
```

6.7.1 プロパティ、フィールドの有無

指定されたStructureプロパティやそのフィールドが存在するかを確認します。プロパティの値の有無は無関係です。

メソッド

```
int DoesStructurePropertyExist(string namespaceUri,
                                                string propertyName)

int DoesStructureFiedExist(string namespaceUri, string propertyName,
                                    string fieldUri, string fieldName)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
fieldUri	Structure プロパティのフィールド名前空間 URI
fieldName	Structure プロパティのフィールド名

戻り値

0	プロパティまたはフィールドが存在しません。
正数	プロパティまたはフィールドが存在しています。
負数	エラーコード

6.7.2 Structure の空プロパティを作成

アイテムの無いStructureプロパティまたはその要素フィールドを新たに作成します。作成する前に同名プロパティが存在していると、そのプロパティまたはその要素フィールドは削除されます。

メソッド

```
int CreateStructure(string namespaceUri, string propertyName)

int CreateStructureField(string namespaceUri, string propertyName,
                                    string fieldUri, string fieldName)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
fieldUri	Structure プロパティのフィールド名前空間 URI
fieldName	Structure プロパティのフィールド名

戻り値

0	成功
負数	エラーコード

6.7.3 フィールド値の取得

Structureプロパティのフィールド値を取得します。

メソッド

```
string GetStructureField(string namespaceUri, string propertyName,
                                    string fieldUri, string fieldName)

int GetStructureField(string namespaceUri, string propertyName,
                    string fieldUri, string fieldName, out string value)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
fieldUri	フィールド名前空間 URI
fieldName	フィールド名
value	フィールド値

戻り値

0 または正数	取得した文字のバイト数
負数	エラーコード

6.7.4 フィールド値の記載

Structureプロパティのフィールドに値を記載します。
値を書き込む場合でプロパティが無い場合やフィールドが無い場合は新たに作成されます。

```
メソッド
  int SetStructureField(string namespaceUri, string propertyName,
                        string fieldUri, string fieldName, string value)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
fieldUri	フィールド名前空間 URI
fieldName	フィールド名
value	フィールド値

戻り値

0	成功
負数	エラーコード

6.7.5 フィールドの数

Structureのフィールドを計数します。

```
メソッド
  int CountStructures(string namespaceUri, string propertyName)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名

戻り値

正数	フィールドの数
負数	エラーコード

6.7.6 プロパティまたはフィールドの削除

Structureプロパティまたは、そのフィールドを削除します。
プロパティがStructureでない場合は失敗します。

```
メソッド
  int DeleteStructureProperty(string namespaceUri, string propertyName)
  int DeleteStructureField(string namespaceUri, string propertyName,
                           string fieldUri, string fieldName)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
fieldUri	フィールド名前空間 URI
fieldName	フィールド名

戻り値

0	成功
負数	エラーコード

6.8 PDF 文書のプロパティ(概要)が記載されるメタデータ

PDF文書には以下のようなPDF文書のプロパティを記載するメタデータ[9]があります。

PDF1.7 以前ではこれらのプロパティを文書情報（Document Information）ディクショナリ[10]に記載していましたが、PDF2.0 ではその記載は非推奨となりました。しかしながら、PDF Metadata は互換性のために文書情報ディクショナリにも記載します。

- タイトル
- 作成者
- 説明（または、サブタイトル）
- キーワード
- 著作権情報
- 作成日
- 更新日
- PDFを作成したアプリケーション

など

以下では「タイトル」、「作成者」、「説明」、「キーワード」、「著作権情報」を記載もしくは変更するための PDF Metadata のメソッド（または関数）について説明します。なお、プロパティ「作成日」、「更新日」、「PDFを作成したアプリケーション」は PDF Metadata によって自動的に更新されます。

[9] ISO32000-2:2020 PDF 2.0 「14.3 Metadata」参照
[10] ISO32000-2:2020 PDF 2.0 「14.3.3 Document information dictionary」参照

6.8.1 文書の「タイトル(title)」

文書のタイトルはメタデータプロパティの Dublin Core 名前空間 title に順番のある配列(Ordered Array)プロパティに複数の項目として記載され、それぞれの項目には代替(Alternative)の言語修飾子(Language Qualifier)が付加されます。

文書のタイトルは文書情報(Document Information)辞書の Title キーに記載される場合がありますが、PDF 2.0 ではその記載が非推奨となりました。

XMP 修飾子

XMP 修飾子は任意の XMP 値にその値の形式を変更せずに注釈を付加するものです。XMP 修飾子の値は任意の XMP 形式にすることができ、複数の修飾子を含めることができます。

中でも、xml:lang（言語修飾子）は特別で XML の構成が違ってきます。しかし、RDF/XML 仕様では「xml:lang は任意のノード要素またはプロパティ要素で使用し、その内容が指定された言語であることを示せる」と規定されています。PDF Metadata はこの言語修飾子の付いた要素を内包する「文書タイトル」のような Alternate Array（代替配列）プロパティの要素選択を容易に行えます。

言語修飾子には x-default（既定言語）を指定でき、この修飾子を持った要素は Array の先頭に置かれます。

PDF Metadata で言語指定された Array の要素を選択するには Generic Language（一般言語）と Specific Language（特定言語）を使います。特定の言語（例えば、ja-JP や en-US など）で選択する場合は Specific Language で指定するだけで十分ですが、en-US もしくは en-UK のいずれかといった選択には Generic Language も使った選択が有用です。言語指定された Alternate Array 内の要素は以下の手順で検索されます。

1. Specific Language（特定言語）で指定された言語と全く同じ言語修飾子を持った要素を検索
2. Generic Language（一般言語）で指定された言語と部分的に適合する言語修飾子を持った要素を選択
   ただし、適合するのは言語修飾子の先頭でありマイナス文字（U+002D）で区切られた部分で、Generic Language が指定された場合
3. x-default を言語修飾子に持つ要素を選択
4. Array の先頭に置かれた要素を選択

この指定方法は、「文書タイトル」に限らず、xml:lang が付加された Alternate Array の要素でも同じ手順で選択できます。

以下は Title メタデータの例です。

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMP Core 6.0.0">
   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <rdf:Description rdf:about=""
            xmlns:dc="http://purl.org/dc/elements/1.1/">
         <dc:title>
            <rdf:Alt>
               <rdf:li xml:lang="x-default">文書タイトル</rdf:li>
               <rdf:li xml:lang="ja-JP">文書タイトル</rdf:li>
               <rdf:li xml:lang="en-US">Document Title</rdf:li>
            </rdf:Alt>
         </dc:title>
      </rdf:Description>
   </rdf:RDF>
</x:xmpmeta>
```

6.8.2 「タイトル(title)」の取得

PDF文書のタイトルを取得します。

文書タイトルがメタデータに記載されていない場合は、文書情報(Document Information)ディクショナリ[11]から取得します。文書タイトルの取得元の違いは、戻り値の違いで知ることができます。

GetTitle()は指定された言語のタイトルを取得でき、GetTitleIndex()では指定されたインデックスの要素を取得できます。CountTitles()でタイトル配列の要素数を取得できます。

C#開発環境
メソッド

```
int GetTitle(string genericLang, string specificLang,
                                out string actualLang, out string title)
int GetTitle(string specificLang, out string actualLang,
                                                    out string title)
int GetTitle(string specificLang, out string title)
int CountTitles()
```

C/C++開発環境
関数

```
int XmpGetTitle(XMP_HANDLE h, TCHAR* genericLang,
                TCHAR* spacificLang, TCHAR** actualLang TCHAR** title)
int XmpCountTitles(XMP_HANDLE h)
```

引数

h	XmpInterface のハンドル
genericLang	RFC3066 プライマリサブタグとしての汎用言語の名称または null、""を指定 特定言語(specificLang)が一致しない場合に使用される言語 既定値： null
specificLang	RFC3066 タグ、または"x-default"で表された特定言語の名称 言語が完全に一致する場合に使用される言語 空文字の場合は x-default が指定されたものとみなされる。
title	PDF文書に記載されたタイトル(Title)文字列
actualLang	メタデータに記載言語修飾子文字列が戻る。

戻り値

負数	エラーコード
正数	CountTitles()ではタイトル要素の総数
正数	タイトル文字列のバイト数
STC_NO_DATA	文書にタイトルの記載がありません。
STC_XMP_NO_TITLE または STC_XMP_ARRAY_HAS_NO_ITEMS	タイトルがメタデータではなく、文書情報辞書だけに記載されています。 これらの場合 actualLang は意味を持ちませんが、文書情報辞書のタイトル値が戻ります。
負数	エラーコード （すべての戻り値は意味を持ちません。）

[11] ISO32000-2:2020 PDF 2.0 「14.3.3 Document information dictionary」参照

6.8.3 「タイトル(title)」の記載

PDF文書にタイトルを記載します。

ReplaceTitle()は既存のタイトルすべてを削除してから指定された文字列をメタデータに記載します。指定された文字列は文書情報(Document Information)ディクショナリ[12]の Title 要素にも記載されます。

SetTitle()は既存の指定言語の項目を変更、または、指定された言語の項目が追記されます。このメソッドの実行によって既定言語のタイトル値が適切に変更され、既定言語のタイトル値は文書情報の Title 要素に転記されます。

C#開発環境
メソッド

```
int ReplaceTitle(string title)
int ReplaceTitle(string specificLang, string title)
int ReplaceTitle(string genericLang, string specificLang, string title)
int SetTitle(string title)
int SetTitle(string specificLang, string title)
int SetTitle(string genericLang, string specificLang, string title)
```

C/C++開発環境
関数

```
int XmpReplaceTitle(XMP_HANDLE h, TCHAR* genericLang,
                                  TCHAR* spacificLang, TCHAR* title)
int XmpSetTitle(XMP_HANDLE h, TCHAR* genericLang,
                                  TCHAR* spacificLang, TCHAR* title)
```

引数

h	XmpInterface のハンドル
title	タイトル(Title)を表す文字列
genericLang	RFC3066 プライマリサブタグとしての汎用言語の名称または null、""を指定 特定言語(specificLang)が一致しない場合に使用される言語 既定値： null
specificLang	RFC3066 タグ、または"x-default"で表された特定言語の名称 言語が完全に一致する場合に使用される言語 既定値：x-default

戻り値

0	成功
負数	エラーコード

汎用言語および特定言語は「4.1.3 代替言語」を参照ください。

[12] ISO32000-2:2020 PDF 2.0 「14.3.3 Document information dictionary」参照

6.8.4 「タイトル(title)」の削除

PDF文書からタイトルを削除します。
DeleteTiteleItem はタイトルの指定された要素を削除し、DeleteTitle はタイトルをすべて削除します。

メソッド

```
int DeleteTitleItem(string genericLang, string specificLang)
int DeleteTitle()
```

引数

genericLang	RFC3066 プライマリサブタグとしての汎用言語の名称または null、""を指定 特定言語(specificLang)が一致しない場合に使用される言語 既定値： null
specificLang	RFC3066 タグ、または"x-default"で表された特定言語の名称 言語が完全に一致する場合に使用される言語 既定値：x-default

戻り値

0	成功
負数	エラーコード

6.8.5 文書の「作成者(creator)」

作成者を記載するプロパティは Dublin Core 名前空間 creator (dc:creator) にその値が順番を持つ配列(Ordered Array)プロパティに複数の項目として記載されます。

文書の作成者は文書情報(Document Information)辞書の Author キーに記載される場合がありますが、PDF 2.0 ではその記載が非推奨となりました。

以下は Creator メタデータの例です。

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMP Core 6.0.0">
   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <rdf:Description rdf:about=""
            xmlns:dc="http://purl.org/dc/elements/1.1/">
         <dc:creator>
            <rdf:Seq>
               <rdf:li>作成者 1</rdf:li>
               <rdf:li>作成者 2</rdf:li>
               <rdf:li>作成者 3</rdf:li>
            </rdf:Seq>
         </dc:creator>
      </rdf:Description>
   </rdf:RDF>
</x:xmpmeta>
```

6.8.6 「作成者(creator)」の取得

PDF文書の作成者を取得します。文書の作成者がメタデータに記載されていない場合は、文書情報(Document Information)ディクショナリ [13]から取得します。文書タイトルの取得元の違いは、戻り値の違いで知ることができます。CountCreatorItems は作成者要素の総数を戻します。

C#開発環境
メソッド

```
int CountCreatorItems()
int GetCreator(uint itemIndex, out string creator)
int GetCreator(out string creators)
```

C/C++開発環境
関数

```
int XmpGetCreator(XMP_HANDLE h, TCHAR** creators)
int XmpCountCreatorItems(XMP_HANDLE h)
int XmpGetCreatorIndex(XMP_HANDLE h, int itemIndex, TCHAR** creator)
```

引数

h	XmpInterface のハンドル
itemIndex	タイトル配列の要素インデックス 先頭のインデックスは１ インデックス番号を省略すると、先頭の要素を取得
creator	PDF文書に記載された作成者(Creator)文字列
creators	メタデータに記載されたすべての作成者(Creator)を結合した文字列

戻り値

正数	CountCreatorItems()では作成者要素の総数です。
正数	メタデータからの読み取りに成功すると作成者文字列のバイト数が戻ります。
STC_NO_DATA	文書に作成者の記載がありません。
STC_XMP_NO_CREATOR または STC_XMP_ARRAY_HAS_NO_ITEMS	作成者がメタデータではなく、文書情報辞書だけに記載されています。 これらの戻り値は文書情報辞書の作成者値です。
他の負数	エラーコード（すべての戻り値は意味を持ちません。）

[13] ISO32000-2:2020 PDF 2.0「14.3.3 Document information dictionary」参照

6.8.7 「作成者(creator)」の記載

ReplaceCreator()は既存の作成者すべてを削除してから指定された文字列をメタデータに記載します。指定された文字列は文書情報(Document Information)ディクショナリ[14]の Author 要素にも記載されます。

SetCreator()は既存の配列の最終項目として追記されます。このメソッドの実行後に dc:creator 配列の先頭項目（アイテム番号１の項目）に記載された値が文書情報辞書の Creator 要素に転記されます。

C#開発環境
メソッド

```
int ReplaceCreator(string creator)
int SetCreator(string creator)
int SetCreatorInsex(int itemIndex, string creator)
```

C/C++開発環境
関数

```
int XmpReplaceCreator(XMP_HANDLE h, TCHAR* creator)
int XmpSetCreator(XMP_HANDLE h, TCHAR* creator)
int XmpSetCreator(XMP_HANDLE h, int itemIndex, TCHAR* creator)
```

引数

h	XmpInterface のハンドル
itemIndex	作成者要素のインデックス番号（先頭=1）
creator	作成者を表す文字列

戻り値

0	成功
負数	エラーコード

6.8.8 「作成者(creator)」の削除

PDF文書から作成者を削除します。
DeleteCreatorItem はタイトルの指定された要素を削除し、DeleteCreator は作成者をすべて削除します。

メソッド

```
int DeleteCreatorItem(int itemIndex)
int DeleteCreator()
```

引数

itemIndex	作成者要素のインデックス番号（先頭=1）

戻り値

0	成功
負数	エラーコード

[14] ISO32000-2:2020 PDF 2.0 「14.3.3 Document information dictionary」参照

6.8.9 文書の「説明(description)」

文書の内容の説明を記載するプロパティは Dublin Core 名前空間 description（dc:description）に代替言語(Alternate Language)値が順番のある配列(Ordered Array)プロパティに複数の項目として記載されます。

文書の説明は文書情報(Document Information)辞書の Subject キーに記載される場合がありますが、PDF 2.0 ではその記載が非推奨となりました。

以下は Description メタデータの例です。

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMP Core 6.0.0">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about=""
        xmlns:dc="http://purl.org/dc/elements/1.1/">
      <dc:description>
        <rdf:Alt>
          <rdf:li xml:lang="x-default">説明</rdf:li>
          <rdf:li xml:lang="ja-JP">説明</rdf:li>
          <rdf:li xml:lang="en-UK">Description</rdf:li>
        </rdf:Alt>
      </dc:description>
    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
```

6.8.10 「説明(description)」の取得

PDF文書の説明を取得します。

文書の説明がメタデータに記載されていない場合は、文書情報(Document Information)ディクショナリ [15]から取得します。文書説明の取得元の違いは、戻り値の違いで知ることができます。

GetDescription()は指定された言語のタイトルを取得でき、GetTitleIndex()では指定されたインデックスの要素を取得できます。CountDescriptions()でタイトル配列の要素数を取得できます。

C#開発環境
メソッド

```
int GetDescriotion(string genericLang, string specificLang,
                           out string actualLang, out string description)
int GetDescription(string specificLang, out string actualLang,
                                                   out string description)
int GetDescription(string specificLang, out string title)
int CountDescriptions()
```

C/C++開発環境
関数

```
int XmpGetDescription(XMP_HANDLE h, TCHAR* genericLang,
           TCHAR* spacificLang, TCHAR** actualLang TCHAR** description)
int XmpCountDescriptions(XMP_HANDLE h)
```

引数

h	XmpInterface のハンドル
genericLang	RFC3066 プライマリサブタグとしての汎用言語の名称または null、""を指定 特定言語(specificLang)が一致しない場合に使用される言語 既定値：null
specificLang	RFC3066 タグ、または"x-default"で表された特定言語の名称 言語が完全に一致する場合に使用される言語 空文字の場合は x-default が指定されたものとみなされる。
description	PDF文書に記載された説明(Description)文字列
actualLang	メタデータに記載言語修飾子文字列が戻る。

戻り値

負数	エラーコード
正数	CountDescriptins()では説明要素の総数
正数	説明文字列のバイト数
STC_NO_DATA	文書にタイトルの記載がありません。
STC_XMP_NO_TITLE または STC_XMP_ARRAY_HAS_NO_ITEMS	説明がメタデータではなく、文書情報辞書だけに記載されています。 これらの場合 actualLang は意味を持ちませんが、文書情報辞書の説明値が戻ります。
負数	エラーコード（すべての戻り値は意味を持ちません。）

[15] ISO32000-2:2020 PDF 2.0「14.3.3 Document information dictionary」参照

6.8.11 「説明(description)」の記載

ReplaceDescription()は既存の説明すべてを削除してから指定された文字列をメタデータに記載します。指定された文字列は文書情報(Document Information)ディクショナリ[16]の Subject 要素にも記載されます。

SetDescription()は既存の指定言語の項目を変更、または、指定された言語の項目が追記されます。このメソッドの実行で、dc:description に記載された既定言語の値が文書情報の Subject 要素に記載されます。

C#開発環境
メソッド

```
int ReplaceDescription(string description)
int ReplaceDescription(string specificLang, string description)
int ReplaceDescription(string genericLang, string specificLang,
                                                    string description)
int SetDescription(string description)
int SetDescription(string specificLang, string description)
int SetDescription(string genericLang, string specificLang,
                                                    string description)
```

C/C++開発環境
関数

```
int XmpReplaceDescription(XMP_HANDLE h, TCHAR* genericLang,
                                  TCHAR* spacificLang, TCHAR* description)
int XmpSetDescription(XMP_HANDLE h, TCHAR* genericLang,
                                  TCHAR* spacificLang, TCHAR* description)
```

引数

h	XmpInterface のハンドル
description	詳細(Description)を表す文字列
genericLang	RFC3066 プライマリサブタグとしての汎用言語の名称または null、""を指定 特定言語(specificLang)が一致しない場合に使用される言語 既定値： null
specificLang	RFC3066 タグ、または"x-default"で表された特定言語の名称 言語が完全に一致する場合に使用される言語 既定値： x-default

戻り値

0	成功
負数	エラーコード

汎用言語および特定言語は「4.1.3 代替言語」を参照ください。

[16] ISO32000-2:2020 PDF 2.0 「14.3.3 Document information dictionary」参照

6.8.12 「説明(description)」の削除

PDF文書から説明を削除します。
DeleteDescriptionItem は説明の指定された要素を削除し、DeleteDescription は説明をすべて削除します。

メソッド

```
int DeleteDescriptionItem(string genericLang, string specificLang)
int DeleteDescription()
```

引数

genericLang	RFC3066 プライマリサブタグとしての汎用言語の名称または null、""を指定 特定言語(specificLang)が一致しない場合に使用される言語 既定値： null
specificLang	RFC3066 タグ、または"x-default"で表された特定言語の名称 言語が完全に一致する場合に使用される言語 既定値：x-default

戻り値

0	成功
負数	エラーコード

6.8.13 文書の「キーワード(keywords)」

キーワードを記載するプロパティは Dublin Core 名前空間 description (dc:subject) にその値が順番を持たない配列(Unordered Array)プロパティに複数の項目として記載されます。さらに PDF 名前空間 Keywords (pdf:Keywords) に単純(Simple)なプロパティとしてすべてのキーワードが連結して記載されます。

PDF Metadata は、指定されたキーワードを双方のプロパティに記載します。

文書のキーワードは文書情報(Document Information)辞書の Keywords キーに記載される場合がありますが、PDF 2.0 ではその記載が非推奨となりました。

以下はキーワードメタデータの例です。

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMP Core 6.0.0">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about=""
        xmlns:pdf="http://ns.adobe.com/pdf/1.3/"
        xmlns:dc="http://purl.org/dc/elements/1.1/">
      <dc:subject>
        <rdf:Bag>
          <rdf:li>キーワード 1</rdf:li>
          <rdf:li>キーワード 2</rdf:li>
        </rdf:Bag>
      </dc:subject>
      <pdf:Keywords>キーワード 1, キーワード 2</pdf:Keywords>
    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
```

6.8.14 「キーワード(keywords)」の取得

GetKeywords()は連結されたキーワードすべてを取得し、GetSubjectIndex()は指定されたアイテムのキーワードを取得します。

C#開発環境
メソッド

```
int GetKeywords(out string keywords)
int GetSubjectIndex(int itemIndex, out string keyword)
```

C/C++開発環境
関数

```
int XmpGetKeywords(out string keywords)
int XmpGetSubjectIndex(int itemIndex, TCHAR* keyword)
```

引数

keywords	連結されたキーワード文字列
keyword	単一のキーワード文字列

戻り値

0	成功
負数	エラーコード

6.8.15 「キーワード(keywords)」の記載

ReplaceKeywords()は既存のキーワードすべてを削除してから指定された文字列をメタデータに記載し、SetKeywords()は項目を追加します。

Keywords として渡された文字列は区切り文字で分割されますが、Word として渡された文字列は分割されません。いずれの場合でもキーワードは dc:subject および pdf:Keywords に記載されます。

連結された文字列の区切り文字は“, ”(U+002C U+0020)に正規化されます。そして、文書情報(Document Information)辞書にも記載されます。

C#開発環境
メソッド

```
int ReplaceKeywords(string keywords)
int SetKeywords(string keywords)
int ReplaceSubject(string word)
int SetSubject(string word)
```

C/C++開発環境
関数

```
int XmpReplaceKeywords(XMP_HANDLE h, TCHAR* keywords)
int XmpSetKeywords(XMP_HANDLE h, TCHAR* keywords)
int XmpReplaceSubjrct(XMP_HANDLE h, TCHAR* word)
int XmpSetSubject(XMP_HANDLE h, TCHAR* word)
```

引数

h	XmpInterface のハンドル
keywords	キーワード文字列 (区切り文字を含む場合は複数ワードに分割される)
word	キーワード文字列 (すべての文字がキーワードとなり、分割されない)

戻り値

0	成功
負数	エラーコード

キーワード分割について

Keywords で指定された文字列は、区切り文字で分割され、順番の無い配列(Unordered)プロパティの各項目に記載されます。

Keywords に含められる区切り文字は改行(U+000D)、ラインフィード(U+000A),コンマ(U+002C)、コロン(U+003A)、セミコロン(U+003B)、タブ(U+0009)、ダブルスペース(U+0020 U+0020)で、以下のように分割されます。

- 連続した区切り文字は1つの区切り文字として扱われます。
- Keywords に指定された文字列の先頭および末尾の区切り文字は無視されます。
- 区切り文字で分割された文字列の先頭および末尾にある1つ以上の空白は無視されます。
- 区切り文字に隣接するか末尾のダブルクォート(U+0022)で囲まれた文字列内の区切り文字は通常の(区切り文字ではない)文字として扱われます。
- 文字列を囲むダブルクォートが不足の場合は末尾にダブルクォートを追加して分割されます。
- ダブルクォートで囲まれた文字列が空の場合は無視されます。

6.8.16 「キーワード(keywords)」の削除

DeleteKeyword() または、DeleteSubject() ですべてのキーワードを削除し、DeleteSubjectIndex () で指定の要素を削除します。

C#開発環境
メソッド

```
int DeleteKeywords()
int DeleteSubjects()
int DeleteSubjectIndex(int itemIndex)
```

C/C++開発環境
関数

```
int XmpDeleteKeywords()
int XmpDeleteSubjects()
int XmpDeleteSubjectIndex(int itemIndex)
```

6.8.17 文書の「著作権情報」

著作権情報は Dublin Core 名前空間 rights（dc:rights）に代替言語(Alternate Language)値が順番のある配列(Ordered Array)プロパティに複数の項目として記載されます。また、XMP Rights 名前空間 WebStatement（xmpRights:WebStatement）には単純(Simple)プロパティの著作権URL情報が記載されます。著作権の有無は xmpRights:Marked に「著作権取得済み」(“True”)、「著作権フリー」（“False”）、「不明」（未記載）と指定します。

以下は著作権情報メタデータの例です。

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMP Core 6.0.0">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about=""
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:xmpRights="http://ns.adobe.com/xap/1.0/rights/"
      <dc:rights>
        <rdf:Alt>
    <rdf:li xml:lang="x-default">Copyright 2026 サンプル（株）</rdf:li>
    <rdf:li xml:lang="ja-JP">Copyright 2026 サンプル（株）</rdf:li>
    <rdf:li xml:lang="en-US">Copyright 2026 Sample Co.,Ltd.</rdf:li>
        </rdf:Alt>
      </dc:rights>
<xmpRights:WebStatement>https//www.sample.com/</xmpRights:WebStatement>
<xmpRights:Marked>True</xmpRights:Marked>
    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
```

6.8.18 「著作権情報」の記載

ReplaceRights()および ReplaceRightsWebStatement()はそれぞれの既存著作権情報を削除してから指定された文字列をメタデータに記載します。RepaceRightsMarked では権限管理の有無を示し、OmitRightsMarked で「不明」にします。

SetRights()は既存の指定言語の項目を変更、または、指定された言語の項目が追記されます。

C#開発環境
メソッド

```
int ReplaceRights(string rights)
int ReplaceRights(string specificLang, string rights)
int ReplaceRights(string genericLang, string specificLang,
                                                    string rights)
int SetRights(string rights)
int SetRights(string specificLang, string rights)
int SetRights(string genericLang, string specificLang, string rights)

int ReplaceRightsWebStatement(string rightsWebStatemant)
int ReplaceRightsMarked(bool flag)
int OmitRightsMarked()      // RightsMarked のフラグを削除し「不明」とします
```

C/C++開発環境
関数

```
int XmpReplaceRights(XMP_HANDLE h, TCHAR* genericLang,
                                   TCHAR* spacificLang, TCHAR* rights)
int XmpSetRights(XMP_HANDLE h, TCHAR* genericLang,
                                   TCHAR* spacificLang, TCHAR* rights)

int XmpReplaceRightsWebStatement(TCHAR* webStatemant)
int XmpReplaceRightsMarked(BOOL flag)
int XmpOmitRightsMarked()   // RightsMarked のフラグを削除し「不明」とします
```

引数

h	XmpInterface のハンドル
rights	著作権情報文字列
webStatement	著作権情報の URL を表す文字列
genericLang	RFC3066 プライマリサブタグとしての汎用言語の名称または null、""を指定 特定言語(specificLang)が一致しない場合に使用される言語 既定値： null
specificLang	RFC3066 タグ、または"x-default"で表された特定言語の名称 言語が完全に一致する場合に使用される言語 既定値： x-default
flag	真：権限管理されたリソースであることを示す。 偽：パブリックドメインのリソースであることを示す。

戻り値

0	成功
負数	エラーコード

汎用言語および特定言語は「4.1.3 代替言語」を参照ください。

6.9 日付データ

日付データは以下の形式が定められています。
先頭の数字は、日付を扱うメソッドで指定または取得する日付形式の番号です。

```
1 YYYY
2 YYYY-MM
3 YYYY-MM-DD
4 YYYY-MM-DDThh:mmTZD
5 YYYY-MM-DDThh:mm:ssTZD
6 YYYY-MM-DDThh:mm:ss.sTZD
```

ここで、

- YYYY は4桁の西暦年を表す整数です
- MM は2桁の月を表す整数（1 月は 01）です
- DD は2桁の日を表す整数（01 から 31 まで）です
- hh は2桁の時を表す整数（00 から 23 まで）です
- mm は2桁の分を表す整数（00 から 59 まで）です
- ss は2桁の秒を表す整数（00 から 59 まで）です
- s は１桁以上で秒の少数部を表します
- TZD はタイムゾーン指定子（Time Zone Designator）で、Z、+hh:mm、-hh:mm のいずれかで表します。
  ライブラリではタイムゾーンの地理的位置を以下のように+1（正数）,0（ゼロ）,-1（負数）で指定します。

  +1　経度 0 度よりも東側（East of UTC）に位置する地域、その時差を+hh:mm と表し UTC 時刻より進んでいます。
  0　経度 0 度の地域の時刻で UTC 時刻です。
  -1　経度 0 度よりも西側（West of UTC）に位置する地域、その時差を-hh:mm と表し UTC 時刻より遅れています。

以下はメタデータに記載された日付の例です。

```
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMP Core 6.0.0">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about=""
        xmlns:xmp=”http://ns.adobe.com/xap/1.0/”>
      <xmp:ModifyDate>2026-03-21T10:23:07+09:00</xmp:ModifyDate>
      <xmp:CreateDate>2026-01-23T12:34:56Z</xmp:CreateDate>
      <xmp:MetadataDate>2026-03-21T10:23:07+09:00</xmp:MetadataDate>
    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
```

6.9.1 日付データを Simple プロパティから取得

一般に日付データは Simple プロパティに記載されます。他のプロパティに記載された日付は文字列で読み出した後に日付データに変換します。

C#開発環境
メソッド

```
int GetSimpleUtcDate(string namespaceUri, string propertyName,
                                                    long utcTime)

int GetSimpleUtcDate(string namespaceUri, string propertyName,
                                                DateTime utcDateTime)

int GetSimpleLocalDate(string namespaceUri, string propertyName,
                                                   long localTime)

int GetSimpleLocalDate(string namespaceUri, string propertyName,
                                               DateTime localDateTime)
```

C/C++開発環境
関数

```
int XmpGetSimpleUtcDate(XMP_HANDLE h, TCHAR* namespaceUri,
                                    TCHAR* propertyName, time_t* utcTime)

int XmpGetSimpleLocalDate(XMP_HANDLE h, TCHAR* namespaceUri,
                                   TCHAR* propertyName, time_t* localTime)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
utcTime	1970 年 1 月 1 日午前 0 時から経過した秒数で表した UTC 時刻
utcDateTime	UTC 時刻
localTime	1970 年 1 月 1 日午前 0 時から経過した秒数で表した日本時刻
localDateTime	日本時刻

戻り値

0	成功
負数	エラーコード

6.9.2 日付データを Simple プロパティに記載

一般に日付データは Simple プロパティに記載されます。他のプロパティでは日付データを文字列に変換してから記載します。

C#開発環境
メソッド

```
int SetSimpleCurrentDate(string namespaceUri, string propertyName)

int SetSimpleUtcDate(string namespaceUri, string propertyName,
                                                    long utcTime)

int SetSimpleUtcDate(string namespaceUri, string propertyName,
                                                DateTime utcDateTime)

int SetSimpleLocalDate(string namespaceUri, string propertyName,
                                                    long localTime)

int SetSimpleLocalDate(string namespaceUri, string propertyName,
                                                  DateTime localDateTime)

int SetSimpleDateFmt(string namespaceUri, string propertyName,
                  long zonetime, int nanosecond, int formatNumber,
                                  int tzSign, int tzHour, int tzMinute)

int SetSimpleDateFmt(string namespaceUri, string propertyName,
          DateTime zoneDateTime, int nanosecond, int formatNumber,
                                   int tzSign, int tzHour, int tzMinute)
```

C/C++開発環境
関数

```
int XmpSetSimpleDate(XMP_HANDLE h, TCHAR* namespaceUri,
                                    THCAR* propertyName, time_t utcTime)

int XmpSetSimpleCurrentDate(XMP_HANDLE h, THCAR* namespaceUri,
                                                    THCAR* propertyName)

int XmpSetSimpleLocalDate(XMP_HANDLE h, THCAR* namespaceUri,
                                  THCAR* propertyName, time_t localTime)

int XmpSetSimpleDateTz(XMP_HANDLE h, THCAR* namespaceUri,
                THCAR* propertyName, time_t localDateTime, int tzSign,
                                                int tzHour, int tzMinute)
```

引数

namespaceUri	名前空間 URI
propertyName	プロパティ名
utcTime	1970 年 1 月 1 日午前 0 時から経過した秒数で表した UTC 時刻
utcDateTime	UTC 時刻
localTime	1970 年 1 月 1 日午前 0 時から経過した秒数で表した日本時刻
localDateTime	日本時刻
zoneTime	1970 年 1 月 1 日午前 0 時から経過した秒数で表した現地（タイムゾーン）の時刻
zoneDateTime	現地（タイムゾーン）時刻
nanosecond	時刻秒の少数部 単位[ナノ秒] （1 ナノ秒 = 0.000000001 秒）
formatNumber	時刻を表す文字列の形式番号 「6.9 日付データ」参照
tzSign	タイムゾーンの地理的位置 「6.9 日付データ」参照
tzHour	タイムゾーン時差の時
tzMinute	タイムゾーン時差の分

戻り値

0	成功
負数	エラーコード

6.9.3 文字列の日付を日付データに変換

様々なプロパティに記載された日付文字列を日付データに変換します。日付文字列は「6.9 日付データ」に示された形式でなければなりません。

C#開発環境
メソッド

```
int ConvertToLocalDate(string dateString, out long localTime)
int ConvertToLocalDate(string dateString, out DateTime localTime)
int ConvertToUtcDate(string dateString, out long utcTime)
int ConvertToUtcDate(string dateString, out DateTime utcTime)
int ConvertToFmtDate(string dateString, int formatNumber,
                            out long localTime, out int nanosecond,
                        out int tzSign, out int tzHour, out int tzHour)
int ConvertToFmtDate(string dateString, int formatNumber,
                          out DateTime localTime, out int nanosecond,
                        out int tzSign, out int tzHour, out int tzHour)
```

C/C++開発環境
関数

```
int XmpConvertToLocalDate(XMP_HANDLE h, TCHAR* dateString,
                                                        long* localTime)
int XmpConvertToUtcDate(XMP_HANDLE h, TCHAR* dateString,
                                                        time_t* utcTime)
int XmpConvertToFmtDate(XMP_HANDLE h, string dateString,
                int* formatNumber, time_t* zoneTime, int* nanosecond,
                                    int* tzSign, int* tzHour, int* tzHour)
```

引数

utcTime	1970 年 1 月 1 日午前 0 時から経過した秒数で表した UTC 時刻
utcDateTime	UTC 時刻
localTime	1970 年 1 月 1 日午前 0 時から経過した秒数で表した日本時刻
localDateTime	日本時刻
zoneTime	1970 年 1 月 1 日午前 0 時から経過した秒数で表した現地（タイムゾーン）の時刻
zoneDateTime	現地（タイムゾーン）時刻
nanosecond	時刻秒の少数部 単位[ナノ秒] （1 ナノ秒 = 0.000000001 秒）
formatNumber	時刻を表す文字列の形式番号 「6.9 日付データ」参照
tzSign	タイムゾーンの地理的位置 「6.9 日付データ」参照
tzHour	タイムゾーンの時
tzMinute	タイムゾーンの分

戻り値

0	成功
負数	エラーコード

6.9.4 日付データを文字列の日付に変換

様々なプロパティに記載するために日付データを日付文字列に変換します。日付文字列は「6.9 日付データ」に示された形式に限られます。

C#開発環境
メソッド

```
int ConvertCurrentDateToStr(out string dateString)
int ConvertLocalDateToStr(long localTime,  out string dateString)
int ConvertLocalDateToStr(DateTime localDateTime,
                                                  out string dateString)
int ConvertUtcDateToStr(long utcTime, out string dateString)
int ConvertUtcDateToStr(DateTime utcDateTime, out string dateString)
int ConvertFmtDateToStr(long zoneTime, int nanosecond,
                        out int tzSign, out int tzHour, out int tzHour)
int ConvertFmtDateToStr(DateTime zoneDateTime, int nanosecond,
                        out int tzSign, out int tzHour, out int tzHour)
```

C/C++開発環境
関数

```
int XmpConvertCurrentDateToStr(XMP_HANDLE h, TCHAR** dateString)
int XmpConvertLocalDateToStr(XMP_HANDLE h, time_t localTime,
                                                  TCHAR** dateString)
int XmpConvertUtcDateToStr(XMP_HANDLE h, time_t* utcTime,
                                                    TCHAR* dateString)
int XmpConvertFmtDateToStr(XMP_HANDLE h, time_t* zoneTime,
                                  int* nanosecond,  int* formatNumber,
                                   int* tzSign, int* tzHour, int* tzHour)
```

引数

utcTime	1970 年 1 月 1 日午前 0 時から経過した秒数で表した UTC 時刻
utcDateTime	UTC 時刻
localTime	1970 年 1 月 1 日午前 0 時から経過した秒数で表した日本時刻
localDateTime	日本時刻
zoneTime	1970 年 1 月 1 日午前 0 時から経過した秒数で表した現地（タイムゾーン）の時刻
zoneDateTime	現地（タイムゾーン）時刻
nanosecond	時刻秒の少数部 単位[ナノ秒] （1 ナノ秒 = 0.000000001 秒）
formatNumber	時刻を表す文字列の形式番号 「6.9 日付データ」参照
tzSign	タイムゾーンの地理的位置 「6.9 日付データ」参照
tzHour	タイムゾーンの時
tzMinute	タイムゾーンの分

戻り値

0	成功
負数	エラーコード

6.10 独自名前空間

独自の名前空間を使用する場合に XmpInterface へ登録します。

```
メソッド
  string RegisterNamespace (string namespaceUri, string suggestedPrefix)
  int RegisterNamespace(string namespaceUri, string suggestedPrefix,
                                              out string registeredPrefix)
```

引数

namespaceUri	名前空間 URI
suggestedPrefix	名前空間に関連付ける希望のプリフィックス
registeredPrefix	名前空間に関連付けられた実際のプリフィックス

戻り値

負数	エラーコード
0	成功
文字列	名前空間 URI に関連付けられたプロパティのプリフィックス

6.11 XMP データをダンプする

現在文書のXMPデータをダンプします。

```
メソッド
  int DumpObjectToConsole()              //コンソールに出力
  int DumpObjectToFile(string fileName) //ファイルに出力
  int DumpObjectToChar(out string data) //文字列で出力
```

引数

fileName	出力ファイル名
data	出力文字列

戻り値

0	成功
負数	エラーコード

6.12 エラーの詳細

エラーコードが戻された場合に、より詳細なエラー内容を取得できる場合があります。
このメッソドで取得できるメッセージは必ずしも直前のものではありません。

```
メソッド
  string GetLastError()
  int GetLastError(out string message)
```

引数

message	エラーのより詳細な内容

戻り値

0	成功
負数	エラーコード

7.0 著作権

本書「PDF Structure 説明書」は株式会社トラスト・ソフトウェア・システムの著作物です。他媒体への転載を禁止します。