XMP Specification

2004 年 1 月

アドビ システムズ 株式会社
Adobe Solutions Network
http://partners.adobe.com/asn/

目次

目次

はじめに

このマニュアルについて

XMP（Extensible Metadata Platform）は、さまざまなアプリケーションでメタデータを作成、処理および交換するための標準フォーマットとして開発されたものです。

この章では、各章の構成や表記規則など、このマニュアルに関連する情報を提供します。また、追加情報の入手先についても説明します。

対象読者

このマニュアルは、XMP メタデータが含まれているファイルを生成、処理、管理するアプリケーションの開発者を対象にしています。

このマニュアルの構成

このマニュアルは、次の章に分かれています。

- 第 1 章「XMP の概要」では、メタデータとは何かについて説明し、XMP モデルの概要を簡単に説明します。
- 第 2 章「XMP データモデル」では、XMP でサポートされているデータについて、その概念を簡単に説明します。また、メタデータで使用されるスキーマ（多数のプロパティの集合体）についても説明します。
- 第 3 章「XMP ストレージモデル」では、ファイル内での XMP データの構造について説明します。
- 第 4 章「XMP スキーマ」では、XMP メタデータでよく使用されるスキーマについて説明し、各プロパティの値の型などを列記します。また、新しいスキーマを定義して、既存のモデルでサポートされていないプロパティを必要に応じて定義する方法についても説明します。
- 第 5 章「アプリケーションファイルへの XMP メタデータの埋め込み」では、さまざまなアプリケーションファイルに XMP メタデータを埋め込む方法について説明します。

このマニュアルの表記規則

テキストの種類に応じて、次の書体を使用しています。

書体	使用対象：
サンセリフ、標準	XMP のプロパティ名。xmp:CreationDate など。
固定幅、標準	すべての XML コード

追加情報の入手先

XMP メタデータで準拠している各種のインターネット標準および勧告については、次の各サイトを参照してください。

Dublin Core Metadata Initiative	http://purl.org/DC/
Extensible Markup Language（XML）	http://www.w3.org/XML/
IETF Standard for Language element values（RFC 1766）	http://www.ietf.org/rfc/rfc1766.txt?number=1766
ISO 639 Standard for Language Codes	http://www.loc.gov/standards/iso639-2/
ISO 3166 Standard for Country Codes	http://www.iso.ch/iso/en/prods-services/iso3166ma/index.html
Naming and Addressing: URIs, URLs, and so on	http://www.w3.org/Addressing/
Resource Description Framework（RDF）：	http://www.w3.org/RDF/
Resource Description Framework（RDF）Model and Syntax Specification	http://www.w3.org/TR/REC-rdf-syntax/
Unicode	http://www.unicode.org/
XML Namespaces	http://www.w3.org/TR/REC-xml-names/

1 XMP の概要

メタデータとは

メタデータとは、文書の特性やプロパティを記述したデータのことです。メタデータは、文書の主となるコンテンツとは区別されます。例えばワープロ文書の場合、実際のテキストデータやフォーマット情報などはコンテンツに当たります。これに対して、作成者、変更日、著作権情報などのプロパティはメタデータになります。

情報の種類によっては、コンテンツとして扱うかメタデータとして扱うか、ワークフローによって判断が分かれる場合もあります。一般的には、コンテンツから独立してそれ単独で利用価値のある情報が、メタデータになります。例えば、文書で使用されているすべてのフォントのリストは、メタデータとして用意すれば役に立つ場合があるでしょう。それに対して、あるページの特定の段落で使用されているフォントの情報は、論理的にはコンテンツとして扱うことになります。

メタデータを利用すれば、ユーザやアプリケーションはより効率的に文書を扱えるようになります。アプリケーションは、ファイルの中からメタデータを取り出して、さまざまな方法で活用することができます（その文書のネイティブファイル形式に対応していなくても、メタデータは取り出せます）。何人かが共同して製作を進めていくワークフローでは、メタデータを利用することで、管理資産の有用性を大幅に高めることができます。例えば画像ファイルの場合は、作業用タイトル、説明、サムネール画像、知的財産権に関するデータなどのメタデータを含めておくことが考えられます。これらのメタデータにアクセスすれば、画像とファイル名の関連付け、イメージキャプションの検索、画像を使用するための著作権情報の確認などが簡単に行えるようになります。

たいていの場合は、ファイルの変更日付やサイズなどのメタデータがファイルシステムによって提供されています。また、他のアプリケーションやユーザによって、その他のメタデータが提供されている場合もあります。メタデータは、関連するファイルの一部として保存される場合もありますし、そうでない場合もあります。

XMP とは

複数のアプリケーションで効果的にメタデータを扱えるようにするには、各アプリケーション共通の標準が必要になります。XMP（Extensible Metadata Platform）は、そのような標準として開発されたものです。

XMP では、次のような仕組みを導入して、メタデータの定義、作成、および処理方法を標準化しています。

- データモデル：文書のメタデータを構成するための概念で、高い柔軟性を持っています。第 2 章「XMP データモデル」を参照してください。
- ストレージモデル：データモデルの実装です。第 3 章「XMP ストレージモデル」を参照してください。これには、XML ストリームへのメタデータのシリアル化や、XMP パケット（ファイル内にデータをパッケージ化する方法）が含まれています。第 5 章「アプリケーションファイルへの XMP メタデータの埋め込み」で、さまざまなファイル形式に XMP パケットを埋め込む方法を説明しています。

- スキーマ：定義澄みメタデータプロパティ定義のセット。アドビのすべての編集および出版用の製品や、さまざまなベンダのアプリケーションなどを含め、広範なアプリケーションから利用することができます。第 4 章「XMP スキーマ」を参照してください。XMP には、スキーマを拡張および追加するためのガイドラインも設けられています。

XMP の次の機能については、別のマニュアルで説明しています。

- 『The Adobe XMP Toolkit』では、開発者のために、アドビのオープンソースツールキットの API について説明しています。
- 『XMP Custom Panels』では、カスタムパネル記述ファイルの作成方法を説明しています。このファイルを使用すると、XMP をサポートしているアドビアプリケーションの標準の**ファイル情報**ダイアログをカスタマイズして、カスタムのメタデータプロパティを定義、作成および管理することができます。

XMP は、さまざまなワークフローやツール環境に対応できるように設計されています。XMP はローカリゼーションが可能で、Unicode をサポートしています。

XMP メタデータは、W3C の標準である Resource Description Framework（RDF）を使用して、XML 形式のテキストとしてエンコードされます。これについては、第 3 章「XMP ストレージモデル」を参照してください。

注意： このマニュアルや保存される XMP データで使用されている名前空間、キーワードおよび関連する名前の中には、「XAP」または「xap」という文字列が含まれているものがあります。これは、当初使用されていた XMP の内部コード名です。互換性を維持するために、引き続き使用されています。

XMP でサポートされていないこと

アプリケーションでは、XMP メタデータの保存機能や生成機能を提供したり、ユーザがメタデータにアクセスできるようにしたり、拡張機能をサポートしたりして、XMP をサポートできます。

XMP の処理に関連するさまざまな領域のうち、XMP 自身で処理できないものは数多くあります。そうした領域については、XMP メタデータをサポートするアプリケーションおよびツールによって処理を行う必要があります。このマニュアルではそのいくつかについて推奨する方法も説明しています。XMP でサポートされていない領域には、次のものが含まれます。

- 各アプリケーション固有のメタデータ。
- メディア管理システムの操作。
- メタデータに関するユーザインタフェイス。
- XMP で定義されている範囲外のスキーマ定義。
- メタデータプロパティの検証や整合性のチェック。
- ユーザがメタデータを設定または編集する際の要件。

このマニュアルで説明されている XMP のスキーマやガイドラインに従っても、メタデータやメタデータフローの整合性までは保証されません。このような整合性は、特定のアプリケーションやツールを使用して検証および維持する必要があります。

2 XMP データモデル

この章では、XMP でどのようなデータがサポートされているかについて説明します。

- 「メタデータのプロパティ」では、メタデータアイテムがプロパティという形でどのように文書に関連付けられるのかを説明します。
- 10 ページの「スキーマと名前空間」では、プロパティ名の記述方法を説明し、スキーマというグループがプロパティによってどのように構成されているかを説明します。
- 11 ページの「プロパティ値」では、XMP の各プロパティで使用できるデータ型について説明します。

メタデータのプロパティ

XMP のメタデータは、一連のプロパティから構成されています。プロパティは、リソースと呼ばれる特定のエンティティに関連付けられています。つまり、プロパティは、リソースに関連する情報を表しています。リソースには、次のものがあります。

- ファイル。JPEG 画像などの単純なファイルもあれば、PDF 文書などの複雑なファイルもあります。
- ファイル内の、特定の意味を持った一部分。この部分は、ファイルの構造や、アプリケーションがどのようにファイルを扱うかによって決まります。例えば、PDF ファイルにインポートされている画像は意味のあるエンティティであり、関連するメタデータを持たせることができます。しかし、ページの範囲はそうではありません。それに対応する特定の PDF 構造は存在しないからです。一般的に、XMP は、単語や文字などの微小なサブコンポーネントに対して使用するようには設計されていません。

プロパティには、名前と値があります。プロパティは、概念的には、リソースを説明する次のような形の文を構成します。

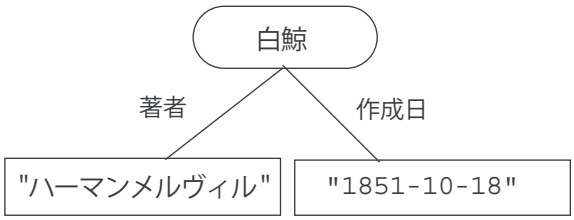
「リソースのプロパティ名はプロパティ値です。」

例えば、次のようになります。

『白鯨』の著者は「ハーマンメルヴィル」です。

この章では、多数の図を使用してデータモデルを表現しています。次の図のように、メタデータツリーの一番上（ルート）は、メタデータが適用される文書または文書の要素を表します。

注意： どのプロパティ名も、有効な XML 名であることが必要です。



スキーマと名前空間

スキーマとは、プロパティのセットです。一般に、スキーマを構成するのは、特定の種類の文書に関連するプロパティのみ、あるいは、ワークフローの特定の段階に関係するプロパティのみです。第 4 章「XMP スキーマ」では、いくつかの標準的なメタデータスキーマについて説明し、新しいスキーマを定義する方法を説明しています。

それぞれのスキーマは、名前空間（XML 名前空間の使用法に準じます）によって識別されます。複数のスキーマで、同じ名前のプロパティが異なる意味で定義されていても、名前空間を使用すれば、それらの競合を回避できます。例えば、別々に設計された 2 つのスキーマに、Creator というプロパティがあるとします。このプロパティは、一方のスキーマではそのリソースを作成した人を表し、もう一方のスキーマでは、そのリソースを作成するのに使用したアプリケーションを表しているかもしれません。こうした名前の競合は、スキーマ固有の名前空間接頭辞をプロパティ名に付けることで回避できます（下記参照）。

スキーマには、次のものが含まれています。

- スキーマ名。スキーマを一意に識別するために使用する URI です。スキーマ名は、単なる一意の文字列です （多くの場合、URL のように見える文字列が使用されますが、その URI に実際に Web ページが存在するとは限りません。アドビの名前空間の場合、対応する Web ページは現時点では存在していません）。この URI は、XML 名前空間の規則に従っている必要があります。また、RDF に準拠するよう、最後の文字は「/」または「#」にする必要があります。

注意： スキーマの URI は、全体で一意の文字列であることに意味があり、その構成要素は重要ではありません。例えば、`foo:/schema/1.0/` と `foo:/schema/2.0/` は全く異なるスキーマであり、両者の間に関係があるとは限りません。

- スキーマの名前空間接頭辞。完全なスキーマ名の短い省略表現です。このマニュアルでは、各スキーマに対して特定の名前空間接頭辞を使用していますが、それらは公式に定められたものではありません。XML 名前空間の規則に準じ、スキーマの名前空間接頭辞はスキーマの URI の単なる省略表現であり、その接頭辞を宣言している `xmlns` 属性のスコープ内でのみ有効です。

例えば、次のコードでは、XMP Basic スキーマの名前空間接頭辞を `xmp` と定義しています。

```
xmlns:xmp="http://ns.adobe.com/xap/1.0/"
```

XML の修飾名の規則に準じ、スキーマのプロパティは *`prefix:name`* と記述します。

ここで、*`prefix`* はスキーマの名前空間接頭辞を表し、*`name`* は有効で単純な XML 名を表します。例えば、xmp:CreateDate のようになります。`

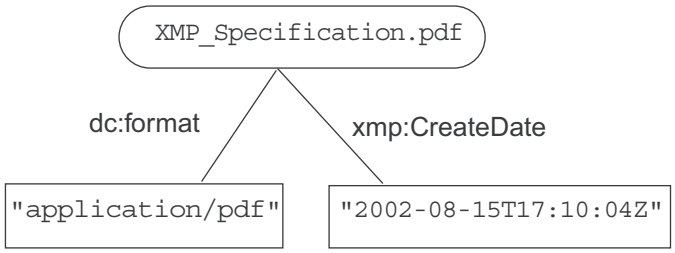
プロパティ値

XMP のプロパティ値に使用できるデータ型は、以下で説明する 3 つの基本カテゴリである単純型、構造体、および配列に分類されます。XMP メタデータは XML として保存されるので、どの型であっても値は Unicode 文字列として記述されます。

この節では、XMP の各データ型の概念を、例を示しながら説明していきます。17 ページの「XMP のシリアル化」では、これらの例が XML でどのように表現されているかが説明されています。あらかじめ定義されているすべてのプロパティおよび値の型については、第 4 章「XMP スキーマ」を参照してください。

単純型

単純型は、単一のリテラル値からなります。単純型には、文字列、ブーリアン、整数、実数などのよく知られている型の他に、Choice などの型もあります。

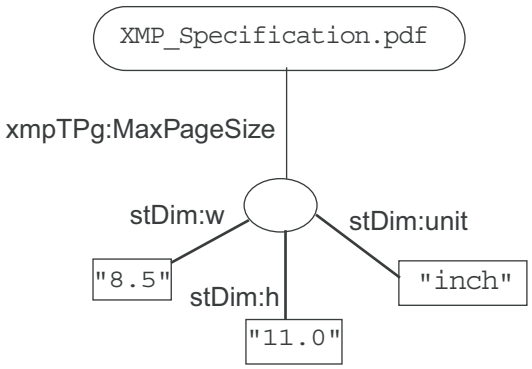


この図では、`XMP_Specification.pdf` という文書に関連付けられている 2 つの単純なプロパティが示されています。

- プロパティ dc:format の値は、「`application/pdf`」という MIMEType の値です。
- プロパティ xmp:CreateDate の値は、「`2002-08-15T17:10:04Z`」という Date 型の値です。

構造体

構造化されているプロパティは、1 つまたは複数の名前付きフィールドからなります。

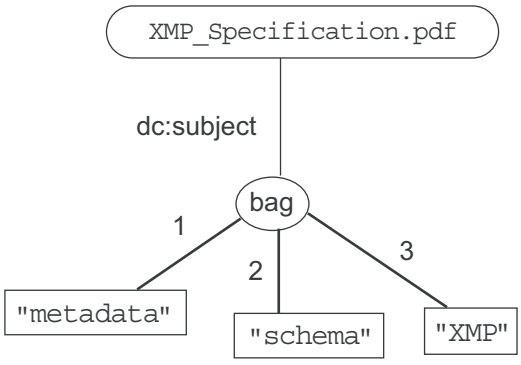


この例では、Dimensions という型の構造化されたプロパティが 1 つ示されています。この構造体は独自の XML 名前空間接頭辞（`stDim`）を持っていますが、これは構造化されたフィールドの必要条件というわけではありません。フィールドは stDim:w（幅）、stDim:h（高さ）および stDim:unit（単位）の 3 つで、それぞれの値は「`8.5`」、「`11.0`」および「`inch`」です。

構造体のフィールド自身が、構造体や配列であることもあります。

配列

配列は、一連の値からなります。これは、次の図に示すように、フィールド名が序数である構造体と考えることができます。



配列の個々の要素は、同じ型になるようにしてください （この例では、要素の型は Text になっています）。配列の要素は、単純型以外にも、構造体や配列にすることもできます。

XMP では、次で説明する無順序配列、順序配列、および択一配列の 3 種類がサポートされています。

無順序配列

無順序配列とは、値の順序に重要性がないリストのことです。例えば、文書に関連するキーワードの場合、その順序には重要性がないのが普通です。そのため、dc:subject プロパティは無順序配列として定義されています。

スキーマ定義では、無順序配列は bag と呼ばれます。例えば、dc:subject は「bag Text」と定義されています。

順序配列

順序配列とは、値の順序に重要性があるリストのことです。例えば、文書の著者の順序は重要な意味を持つことがあります（学術誌など）。そのため、dc:creator プロパティは順序配列として定義されています。

スキーマ定義では、順序配列は seq と呼ばれます。例えば、dc:creator は「seq ProperName」と定義されています。

択一配列

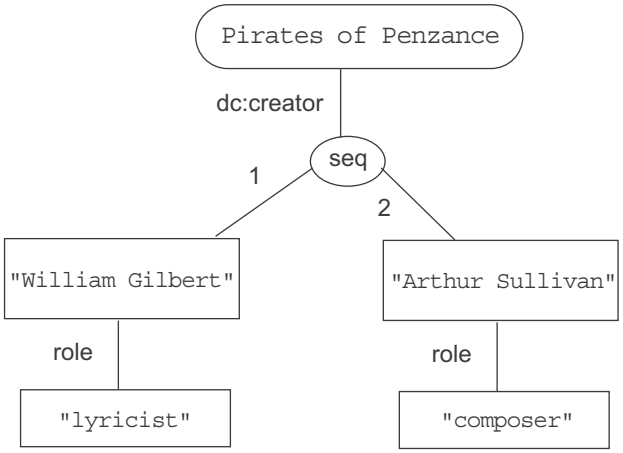
択一配列とは、1 つまたは複数の値の集合で、その中から 1 つが選択されます。スキーマ定義では、択一配列は alt と呼ばれます。例えば、xmp:Thumbnails は「alt Thumbnail」と定義されています。値の選択方法に、決まった規則はありません。アプリケーションが選択する場合もあれば、ユーザが選択する場合もあります。配列の最初の項目は、RDF の規則によってデフォルト値と見なされます。

この配列は、たとえば、同じ内容のテキスト（タイトルや著作権表示など）を複数の言語で用意する場合によく使用されます。この言語別の値については、14 ページの「言語別の値」で詳しく説明しています。

プロパティ修飾子

プロパティ値には、別のプロパティが関連付けられている場合があります。この関連付けられているプロパティをプロパティ修飾子と呼びます。これは言わば「プロパティのプロパティ」であり、プロパティ値の追加情報を提供するのに使用されます。例えば、ミュージカル作品が収録されているデジタルリソースに、dc:creator という配列型のプロパティを使用して、1 人または複数人の作者を指定するとします。配列のそれぞれの値に、role（役割）というプロパティ修飾子を設定して、「作曲家」や「作詞家」などの値を設定することができます。

注意： 現時点では、単純値のプロパティにのみ修飾子を設定することができ、その修飾子自身も単純値である（構造体や配列でない）必要があります。これは、Adobe XMP Toolkit の初期のバージョンの制限に起因しています。



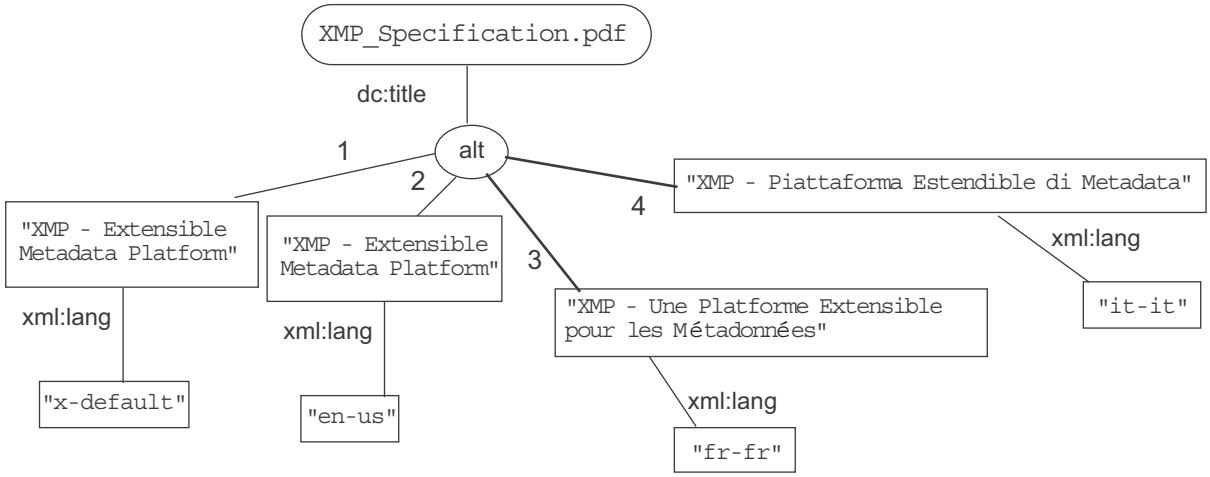
プロパティ修飾子を使用すれば、既存の使用方法を変更せずに値を拡張できます。例えば、図の role 修飾子は、dc:creator の名前を知りたいだけの場合にも邪魔になりません。別の方法として、dc:creator の値を、name フィールドと role フィールドを持つ構造体に変更することも考えられますが、そうすると、単純値を取得しようとする古いソフトウェアが正常に動作しなくなります。

プロパティ修飾子は、言語別の値を保持している配列で最もよく使用されます（次の節を参照してください）。

言語別の値

言語別の値を用意すれば、使用する言語に基づいてプロパティのテキスト値を選択できるようになります。言語別の値を保持している配列の各要素は単純なテキスト値であり、言語修飾子が関連付けられている必要があります。言語修飾子は、前の節で説明したプロパティ修飾子です。そのプロパティ名は xml:lang で、その値には RFC 1766 の表記規則に準拠した文字列を使用します（http://www.ietf.org/rfc/rfc1766.txt を参照してください）。

XMP では、「x-default」言語コードをデフォルトとして設定する必要があります。XMP を認識できない RDF 対応アプリケーションでも使用できるように、この言語コードは配列の最初の要素にしてください。次の図に例を示します。



3 XMP ストレージモデル

この章では、前の章で説明したデータモデルを持つ XMP メタデータが、どのようにファイルに保存されるか（シリアル化されるか）について説明します。

- XMP のプロパティは、XML （具体的には RDF）としてシリアル化されます（次の「XMP のシリアル化」を参照してください）。
- シリアル化されたデータは、ファイルに埋め込めるようパケットの中にラッピングされます。これらのパケットの構造および機能については、26 ページの「XMP パケット」で説明します。
- パケットは、各ファイル形式での規則に従ってファイルに格納されます。個々のファイル形式については、第 5 章「アプリケーションファイルへの XMP メタデータの埋め込み」で説明します。
- 31 ページの「メタデータの外部ストレージ」では、XMP データが関連付けられている文書とは別のファイルにその XMP データを保存する方法を説明します。

XMP のシリアル化

XMP では、文書に関連付けられているメタデータプロパティを表現する方法として（つまり、ファイルにシリアル化する方法として）、XML に基づいて策定された Resource Description Framework （RDF）という標準を採用しています。RDF 標準の採用により、XMP では W3C 標準で利用可能なマニュアル、ツール、共有実装のノウハウの恩恵を受けることができます。RDF については、http://www.w3.org/TR/REC-rdf-syntax/ にある W3C のマニュアル『Resource Description Framework (RDF) Model and Syntax Specification』で説明されています。

XMP パケットの中に含まれている XMP データの高レベル構造は、次のようになっています。このそれぞれについて、以降の各節で説明していきます。

- 一番外側にある要素は、x:xmpmeta 要素です。これは省略可能です。
- この要素には、1 つの rdf:RDF 要素が含まれています。
- この rdf:RDF 要素には、1 つまたは複数の rdf:Description 要素が含まれています。
- それぞれの rdf:Description 要素には、1 つまたは複数の XMP のプロパティが含まれています。

このマニュアルでは、RDF 構文で例を示しています。RDF では、データモデルをシリアル化する方法として、「標準的」な長い表記方法の他に、簡略な表記方法がいくつか用意されています。ここに示す例では、標準的な方法に加えて、Adobe XMP Toolkit で利用されるいくつかの省略表記を使用しています。これは、保存された XMP を人間が読みやすいように設計されたものです。有効な省略表記はすべて使用できます。

XMP では、RDF の一部分のみをサポートしています。詳しくは、25 ページの「RDF に関する問題」を参照してください。

注意： テキストのデフォルトのエンコーディングは、すべての Unicode 文字をサポートする UTF-8 です。マルチバイトの Unicode エンコーディングも使用できます。

x:xmpmeta 要素

シリアル化された XMP データの一番外側の XML 要素は、`x:xmpmeta` 要素にすることをお勧めします。こうすれば、一般的な XML ストリームから XMP メタデータを検索しやすくなります。そのフォーマットは次のとおりです。

```
<x:xmpmeta xmlns:x='adobe:ns:meta/'>
  ... シリアル化された XMP メタデータ
</x:xmpmeta>
```

`xmpmeta` 要素では、属性はいくつでも任意の順序で使用できます。認識できない属性はすべて無視されます。必須の属性はありません。現在定義されている属性は、Adobe XMP Toolkit で定義されている `x:xmptk` のみです。その値は、ツールキットのバージョンを表します。

注意： 初期のバージョンの XMP では、`x:xapmeta` 要素を使用することが推奨されていました。入力をフィルタリングするアプリケーションでは、どちらも認識できる必要があります。

rdf:RDF 要素

`x:xmpmeta` 要素のすぐ内側は、1 つの `rdf:RDF` 要素である必要があります。

```
<x:xmpmeta xmlns:x='adobe:ns:meta/'>
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      ...
  </rdf:RDF>
</x:xmpmeta>
```

rdf:Description 要素

`rdf:RDF` 要素の中には、1 つまたは複数の `rdf:Description` 要素を含めることができます。次の例では、`rdf:Description` 要素が 1 つだけ含まれています。

```
<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/">
      ... Dublin Core のプロパティを記述
  </rdf:Description>
</rdf:RDF>
```

慣例として、1 つの `rdf:Description` 要素の中には、1 つのスキーマのみを対象として、そのプロパティがすべてリストされます （これは必須条件ではなく、読みやすくするための単なる慣習です）。この例では、`rdf:Description` 要素の中に Dublin Core スキーマのプロパティが指定されています。`xmlns:dc` 属性では、名前空間接頭辞（`dc:`）の使用が定義されています。他のスキーマのプロパティがある場合は、新たに `rdf:Description` 要素が追加され、その中に指定されます。

注意： `rdf:Description` 要素は、構造化されたプロパティの指定にも使用されます（21 ページの「構造体」を参照してください）。

rdf:about 属性

rdf:Description 要素の rdf:about は必須の属性で、その XMP がどのリソースについてのメタデータであるかを表します。この属性の値は、URI の構文に従う必要があり、次のいずれかになります。

- 空の文字列（前述の例）。記述対象のリソースに対してその XMP が物理的にローカルであることを表します。XMP をリソースに正しく関連付けるためには、リソースのファイル形式をアプリケーションが理解している必要があります。
- ファイルが保存されるたびに生成される一意のインスタンス ID。インスタンス ID を作成するためのガイドラインは、次の節で説明します。

インスタンス ID

コンピュータファイルへの参照は、多くの場合、不確定要素を含んでいます。つまり、ファイルのコンテンツは、知らないうちに変更される可能性があります。従って、状況に応じて次のいずれかを参照することが望まれます。

- ある時点におけるファイルの特定の状態。
- 内容が変化し得る永続的コンテナとしてのファイル全般。

ファイルの特性の中には、変化しないと思われるもの（作成したアプリケーションなど）もありますし、ファイル内容の変更に伴って変化すると思われるもの（単語数など）もありますし、変化するかどうかわからないもの（著作権情報や作成者の名前など）もあります。

同様に、ファイルのそのような特性を表す XMP プロパティも、ファイルの現時点での内容を表しているのか、それとも、ファイル全般のことを表しているのかは、あいまいです。XMP 自体には、それらを区別するメカニズムは用意されていません。スキーマを使用して、この点が明確になるようにプロパティを定義することが望まれます（が、必須ではありません）。

このマニュアルでは、「メタデータが説明している対象」のことをリソースと呼んでいます。文脈によっては、上で説明した特定の側面または永続的な側面のことをリソースと呼んでいる場合もあります。「ファイルの特定の状態」のことを明言したい場合には、インスタンスという用語を使用しています。

注意： このような用語の使い方は、HTTP での用語の使い方とは異なります。HTTP の場合、リソースは「コンテナ」の意味で最もよく使用されます。一方、エンティティまたはエンティティ部分という用語は、常に「特定の時点における、リソースの全部または一部のコンテンツ」という意味で使用されます。

上で説明したインスタンス ID は、ファイルが保存されるたびに作成されるので、固有の ID になっています。インスタンス ID は、文書の各バージョン間の関係を示すものではありません。しかし、参照されているインスタンスが、特定の時点でのリソースのコンテンツであるなら、インスタンス ID はその時点でのリソースも示しているので、多くの場合、インスタンス ID はリソースを特定するのにも使用できます。したがって、rdf:about 属性でインスタンス ID を使用すると、メタデータが生成または格納された時点でのリソースおよびその特定のコンテンツを識別することができます。

注意： 状況によっては、より永続的な ID が必要な場合もあります。その場合は、XMP Media Management スキーマにある xmpMM:DocumentID プロパティを使用できます。

インスタンス ID は、GUID/UUID スタイルの ID であることが必要です。これは、グローバルに一意であることが保証されている（正確には、競合することは事実上ないと言ってよいくらい競合する確率が低い）大きな整数値のことです。通常は、128 ビットまたは 144 ビットの整数が使用され、22 または 24 文字の Base-64 文字としてエンコードされます。

XMP では、一意の数字を生成する方法に指定はありません。この目的で使用できる一般的な方法にはさまざまなものがあります。例えば、次のようなものがあります。

- ローカルのイーサネットアドレスや、高精度クロックなどの、物理情報を使用する。

  注意： 一意の ID を作成する方法を決める際には、プライバシーの保護を重視するか、それとも監査記録の必要性を重視するかを考える必要があります。アドビアプリケーションでは、プライバシーの保護を重視しているので、イーサネットアドレスは使用していません。

- ローカルで一意であるランダムデータを用意して、その MD5 ハッシュ値を計算する。この方法を使用すれば、イーサネットアドレスを使用した場合のプライバシーの問題を回避できます。また、場合によっては ID を再生成することも可能です。例えば、リソースがデジタル写真であり、その画像コンテンツを使用して MD5 ハッシュを計算している場合には、再生成が可能です。

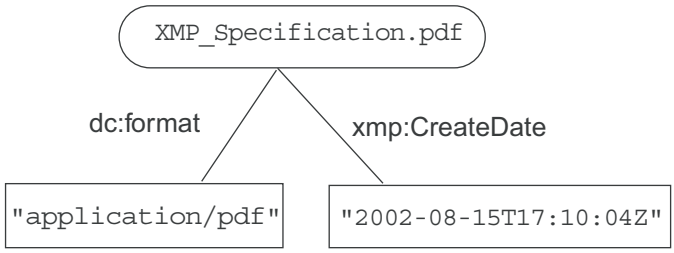
RDF の観点からは、リソースを特定する唯一の ID は `rdf:about` 属性なので、その値は標準的な方法でフォーマットすると便利です。そのようにすれば、RDF 対応のソフトウェアでは、どのような URI が使用されているか（特に、それが URL でないことが）認識できます。抽象 UUID に基づいた URI に関する正式な W3C 勧告はありません。関連がありそうな 2 つの提案を次に示します。

- http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-01.txt
- http://www.ietf.org/internet-drafts/draft-king-vnd-urlscheme-03.txt

XMP のプロパティ

この節では、11 ページの「プロパティ値」に図示したプロパティを XMP にシリアル化する方法を説明します。参照しやすいように、データダイアグラムを再掲します。

単純型



XMP では、これらのプロパティは次のように指定されます。

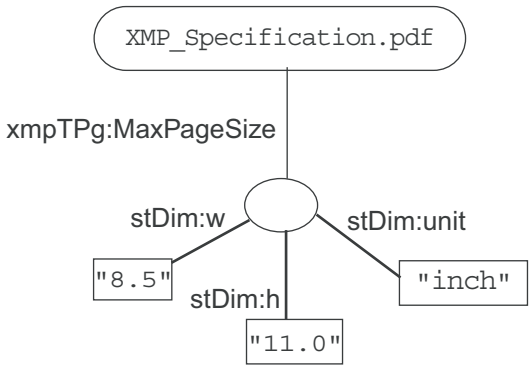
```
<rdf:Description rdf:about="" xmlns:dc="http://purl.org/dc/elements/1.1/">
  <dc:format>application/pdf</dc:format>
</rdf:Description>

<rdf:Description rdf:about="" xmlns:xmp="ns.adobe.com/xap/1.0/">
  <xmp:CreateDate>2002-08-15T17:10:04Z</xmp:CreateDate>
</rdf:Description>
```

別の方法として、`rdf:Description` 要素の属性として単純なプロパティを記述する、RDF の省略表記があります。上の 2 つ目の `rdf:Description` 要素は、次のように指定できます。

```
<rdf:Description rdf:about="" xmlns:xmp="ns.adobe.com/xap/1.0/"
  xmp:CreateDate="2002-08-15T17:10:04Z"/>
```

注意： すべてのプロパティ名は、有効な XML 名であることが必要です。

構造体



この例では、3 つのフィールドを持つ構造体のプロパティが示されています。これは次のように XML にシリアル化できます。

```
<rdf:Description rdf:about=""
        xmlns:xmpTPg="http://ns.adobe.com/xap/1.0/t/pg/">
  <xmpTPg:MaxPageSize>
     <rdf:Description
           xmlns:stDim="http:ns.adobe.com/xap/1.0/sType/Dimensions#">
        <stDim:w>4</stDim:w>
        <stDim:h>3</stDim:h>
        <stDim:unit>inches</stDim:unit>
     </rdf:Description>
  </xmpTPg:MaxPageSize>
</rdf:Description>
```

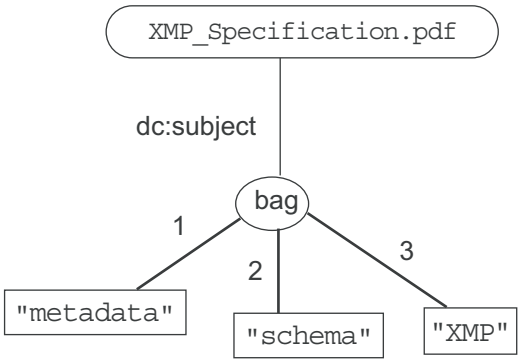
この要素階層は、次のもので構成されています。

- 上で説明した rdf:Description 要素。プロパティの名前空間を指定します。
- xmpTPg:MaxPageSize 要素。Dimensions 型のプロパティです。
- 内側の rdf:Description 要素。構造体の存在を宣言するために必要です。この要素では、構造体のフィールドで使用される名前空間も定義されています。内側の rdf:Description 要素には、rdf:about 属性はありません。

注意： 構造体のフィールドでは、スキーマの名前空間を使用する必要はありませんが、XML の修飾名の規則には従う必要があります。

- Dimensions 構造体のフィールド。

配列



この例（12 ページの「配列」の例）は次のようにシリアル化されます。

```
<rdf:Description rdf:about="" xmlns:dc="http://purl.org/dc/elements/1.1/">
  <dc:subject>
     <rdf:Bag>
        <rdf:li>metadata</rdf:li>
        <rdf:li>schema</rdf:li>
        <rdf:li>XMP</rdf:li>
     </rdf:Bag>
  </dc:subject>
</rdf:Description>
```

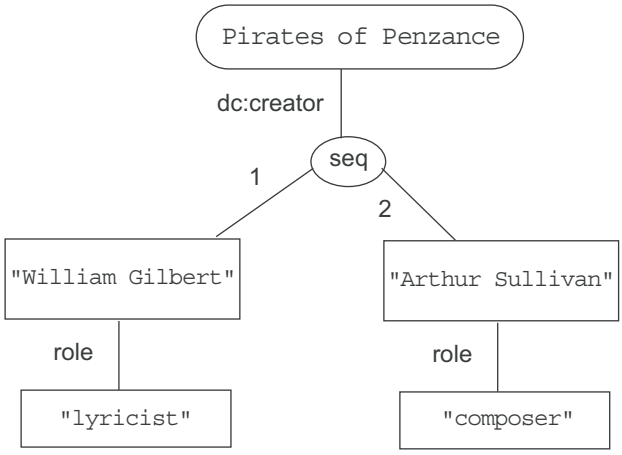
dc:subject プロパティは無順序配列で、rdf:Bag という型で表されます。このプロパティには、配列の要素に対応する rdf:li 要素があります。順序配列と択一配列は、その型がそれぞれ rdf:Seq および rdf:Alt であること以外はよく似ています。択一配列の例は、次の 24 ページの「言語別の値」で説明します。

プロパティ修飾子

プロパティ修飾子は、次の 2 つのいずれかの方法でシリアル化できます。

- 次の図のような一般的な表現を使用する。
- xml:lang 修飾子の場合の特別な表現を使用する（24 ページの「言語別の値」を参照してください）。

一般的な例として、13 ページの「プロパティ修飾子」の図を再掲します。



上の図は、2 つの要素を持つ配列を示しており、各要素には role というプロパティ修飾子が指定されています。これは、次のようにシリアル化できます。

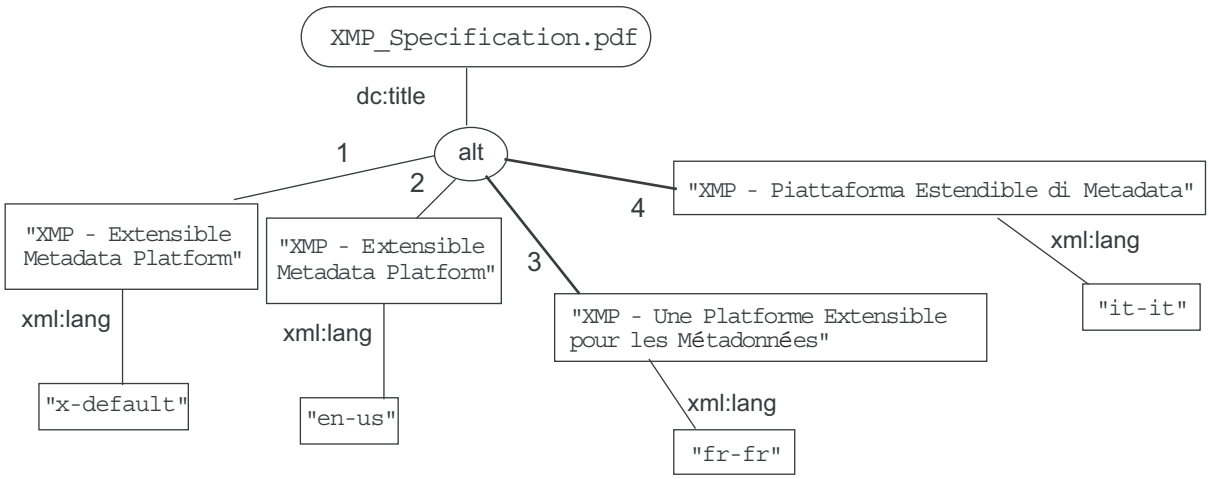
```
<rdf:Description rdf:about=""
        xmlns:dc="http://purl.org/dc/elements/1.1/">
  <dc:creator>
     <rdf:Seq>
        <rdf:li>
           <rdf:Description>
              <rdf:value>William Gilbert</rdf:value>
              <role>lyricist</role>
           </rdf:Description>
        </rdf:li>
           <rdf:Description >
              <rdf:value>Arthur Sullivan</rdf:value>
              <role>composer</role>
           </rdf:Description>
        </rdf:li>
     </rdf:Seq>
  </dc:creator>
</rdf:Description>
```

プロパティ修飾子が存在することは、`rdf:Description` 要素の特別な使い方によって示されています。この例では、それぞれの `rdf:li` 配列要素に、次の内容を保持している `rdf:Description` 要素が含まれています。

- プロパティの値を表す rdf:value という特別な要素。
- 値の修飾子を表す、0 個以上の他の要素。この例では、role というプロパティ修飾子が 1 つ使用されています。

言語別の値

テキストのプロパティには、テキストの言語を指定する xml:lang プロパティ修飾子が指定されている場合があります。これは、言語別の値を保持している配列でよく使用されます。



言語別の値は、rdf:Alt 配列の一種で、Lang Alt 型と呼ばれます。この例では、配列の各要素は単純なテキスト値で、値はプロパティ修飾子を持っています。プロパティ修飾子は xml:lang プロパティとして指定されており、その値の言語を表しています。

この配列の XMP は、次のようになります。

```
<xmp:Title>
  <rdf:Alt>
     <rdf:li xml:lang="x-default">
        XMP - Extensible Metadata Platform </rdf:li>
     <rdf:li xml:lang="en-us">XMP - Extensible Metadata Platform</rdf:li>
     <rdf:li xml:lang="fr-fr">
        XMP - Une Platforme Extensible pour les Metadonnees</rdf:li>
     <rdf:li xml:lang="it-it">
        XMP - Piattaforma Estendible di Metadata</rdf:li>
  </rdf:Alt>
</xmp:Title>
```

xml:lang 修飾子は、文字データを値として持つ XML 要素（この場合は、rdf:li 要素）の属性として記述されています。「x-default」という特別な言語値にも注目してください。これは、デフォルトタイトルを指定します。

RDF に関する問題

サポートされていない機能

XMP では、RDF の一部分のみを使用しています。XMP として有効なのは、前の節で説明した RDF と、それに相当するすべての省略表記のみです。すべての XMP は有効な RDF ですが、RDF 仕様の多くの機能は、XMP としては有効ではありません。特に、次の点に注意してください。

- rdf:RDF は、XMP では必須です（RDF では省略可能です）。
- 最上位レベルの要素は、rdf:Description 要素であることが必要です。
- rdf:ID 属性は無視されます。
- rdf:bagID 属性は無視されます。
- rdf:aboutEach 属性または rdf:aboutEachPrefix 属性はサポートされていません（rdf:Description 全体が無視されます）。
- rdf:parseType='Literal' 属性はサポートされていません。

検証

DTD 検証や XML Schema 検証を行う必要がある場合は、RDF では同じモデルを多くの方法で表現できる、という点に注意してください。また、XMP のオープン性を考えると、使用可能な XML の要素および属性を予測するのは一般に不可能で、制限するのも望ましくありません。任意の要素および属性を使えるように DTD を記述する方法はないようです。ANY を使用するにしても、子要素の宣言が必要になります（XML 仕様の第 3 節にある、検証の制限事項第 4 番を参照してください）。

DTD 検証を使用して XML に XMP を配置する場合は、CDATA セクションに XMP パケットをラッピングすることをお勧めします。その場合、パケット内で使用されている「]]>」はすべてエスケープする必要があります。

XMP パケット

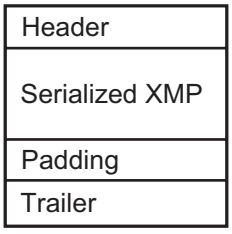
XMP パケット形式では、XMP メタデータをファイルに埋め込む方法が定義されています。これは、前の節で説明したシリアル化された XMP データの周りに「ラッパー」がかぶさったものです。XMP パケットには、次のような特徴があります。

- ネイティブの XML ファイルを含め、さまざまなバイナリ形式およびテキスト形式のファイルに埋め込めます。
- スキャンしやすいマーカを使用して区切られます。これらのマーカは XML 構文と互換性があるので、追加のフィルタ処理を行うことなく XML パーサに転送できます。
- バイトストリーム中の任意の位置を扱えます（マシン固有のワード境界などに依存しません）。
- メタデータのインプレース編集（拡張を含む）ができます。
- 1 つのデータファイルに複数のパケットを埋め込めます。

第 5 章「アプリケーションファイルへの XMP メタデータの埋め込み」では、個々のファイル形式でどのように XMP パケットが埋め込まれているか説明しています。アプリケーションは、認識できないファイル形式であっても、そのファイルの XMP パケットをスキャンできます。ただし、これは最後の手段として使用してください（29 ページの「ファイル内の XMP パケットのスキャン」を参照してください）。

XMP パケットを図式化したものを図 3.1 に示します。これは、ヘッダ、XML データ、パディング、およびトレーラで構成されています。

図 3.1　　XMP パケットの構成



XMP パケットの大まかな構造を次に示します（ヘッダとトレーラのテキストが示されています）。

```
<?xpacket begin=' ■ ' id='W5M0MpCehiHzreSzNTczkc9d'?>
  ... 前述のシリアル化された XMP ...
        <x:xmpmeta xmlns:x='adobe:ns:meta/'>
           <rdf:RDF xmlns:rdf= ...>
              ...
           </rdf:RDF>
        </x:xmpmeta>
  ... パディングとして使用する XML の空白文字 ...
<?xpacket end='w'?>
```

「■」は、Unicode の「幅がゼロで、改行禁止のスペース文字」（U+FEFF）を表し、バイトオーダーのマーカとして使用されます。

XMP パケットは、最初に XML 宣言がないこと以外は、XML 仕様の整形式の要件に準拠している必要があります。ファイル内の各パケットには、それぞれ異なる文字エンコーディングを使用できます。パケットをネストすることはできません。

次の各節で、図 3.1 に示したパケットの各部分を説明します。

ヘッダ

ヘッダでは、フォームの処理命令が指示されています。

```
<?xpacket ... ?>
```

処理命令は、XML 属性として記述されており、パケットに関する情報を表します。必須属性は `begin` 属性と `id` 属性の 2 つで、この順序で指定する必要があります。その他の属性は、そのあとに任意の順序で指定できます。認識できない属性は無視されます。各属性は、空白文字（U+0020）1 つのみで区切る必要があります。

属性：begin

この必須属性は、新しいパケットの開始を表します。その値は、適切に（UTF-8、UTF-16、または UTF-32 で）エンコードされた、Unicode の「幅がゼロで、改行禁止のスペース文字」（U+FEFF）です。この文字は、アプリケーションで記述されたときのオーダー（XML データエンコーディングと同じバイトオーダー）になるため、バイトオーダーのマーカとして機能します。

初期のバージョンの XMP パケット仕様との下位互換性を保つために、この属性の値が空の文字列である場合は、UTF-8 エンコーディングを意味するようになっています。

29 ページの「ファイル内の XMP パケットのスキャン」では、XMP パケットの処理プログラムで 1 バイトずつ読み込んで、バイトオーダーおよびエンコーディングを正しく判定する方法を説明しています。

属性：id

必須属性である `id` は、`begin` の後ろに指定する必要があります。このバージョンの構文で定義されているすべてのパケットでは、`id` の値として次の 7 ビット ASCII の文字列を設定します。

```
W5M0MpCehiHzreSzNTczkc9d
```

この文字列は、パケット全体の文字エンコーディングでエンコードする必要があります。例えば、全体のエンコーディングがビッグエンディアンの UTF-16 である場合は、ヌルを挿入して、`id` 値を 7 ビット ASCII から UTF-16 に変換する必要があります。.

属性：bytes

注意： この属性は推奨されていません。

`bytes` 属性は省略可能で、パケット全体の長さをバイト単位で指定するのに使用します。これを指定すれば、XMP パケットをより速くスキャンできるようになります。この長さが、トレーラの処理命令の末端までの長さよりも長い場合、余った部分のバイトは、適切にエンコードされた Unicode の空白文字で埋める必要があります。これはパディング文字と見なされます。

bytes 属性は、XMP パケットがバイナリファイルに埋め込まれる場合にのみ使用してください。テキストファイルに埋め込まれる場合には使用しないでください。テキストの長さは、コンピュータによって変化することがあるからです。例えば、Macintosh または UNIX システムのテキストファイルを Windows に移動すると、シングルバイトの行末文字（CR や LF）は、通常、すべて 2 バイト（CRLF）になります。したがって、bytes 属性で指定した長さは正しくなくなります。

属性：encoding

注意： この属性は推奨されていません。

この属性は省略可能で、XML 宣言の encoding 属性と同じです（XML 仕様の［23］と［80］を参照してください）。これは、パケット全体の文字エンコーディングを指定します。これは、begin 属性で示されている Unicode エンコーディングと一致している必要があります。

XMP データ

XMP のバイトデータはここに配置します。そのエンコーディングは、ヘッダの begin 属性で示されているエンコーディングと一致している必要があります。このデータの構造は、前述の「XMP のシリアル化」で説明されています。

XMP データには、XML 宣言は含めないでください。XML 仕様では、XML 宣言を「エンティティの最初に置く」ことになっていますが、これはファイルに埋め込まれる XMP パケットには当てはまりません。

注意： XMP パケットには、XMP に準拠していない他の XML を含めることはできません。

パディング

パケットには、2 KB から 4 KB のパディングを挿入することをお勧めします。そうすれば、既存のアプリケーションデータを上書きせずにインプレースで XMP を編集し、必要に応じて拡張することができます。パディングには、XML に対応した空白を使用する必要があります。適切なエンコーディングの空白文字（U+0020）を使用し、約 100 文字ごとに改行を挿入することをお勧めします。

トレーラ

これは必須の処理命令で、XMP パケットの終了を示します。

```
<?xpacket end='w'?>
```

属性：end

end 属性は必須で、他の属性よりも前に指定する必要があります。

注意： このあとに認識できない属性があっても構いませんが、無視されます。各属性は、空白（U+0020）文字 1 つのみで区切る必要があります。

end の値は、そのファイル形式を認識できないアプリケーションに XMP パケットの更新を許可するかどうかを示します。

- r は、パケットが「読み取り専用」であることを表しており、インプレースで更新しないようにする必要があります。

  注意： r は、ファイル形式を認識できファイルを正しく書き直せるアプリケーションの動作まで制限するものではありません。

- w は、十分なスペースがあれば、パケットをインプレースで更新できることを表します。パディングを必要に応じて調整し、パケット全体の長さが変わらないようにしてください。XMP を含むテキストファイルを破損したり、元のアプリケーションの他の制限に違反することを避けるために、元のエンコーディングおよびバイトオーダーは維持する必要があります。

ファイル内の XMP パケットのスキャン

この節では、ファイル内の XMP パケットをスキャンする方法を説明します。また、スキャンに注意が必要な理由を説明します。

注意事項

アプリケーションが個々のファイル形式を認識した上で XMP パケットにアクセスするのが、最も良い方法です。個々のファイル形式で XMP パケットがどのように埋め込まれているかについて詳しくは、第 5 章「アプリケーションファイルへの XMP メタデータの埋め込み」を参照してください。

認識できないファイル形式の場合は、ファイルをスキャンして XMP パケットを見つけることができます。ただし、この方法は最後の手段としてください（データを変更する必要がある場合は特に）。ファイル形式を認識できない場合、単にパケットを見つけるだけでは不十分な場合があります。次のような問題が考えられます。

- XMP がどのリソースに関連付けられているか判断できない場合があります。XMP に対応していないアプリケーションのページレイアウトファイルの中に、XMP が設定された JPEG 画像が配置されている場合、そのファイルにある XMP パケットは、レイアウト全体ではなく、その画像のみを参照しています。
- ファイルに複数の XMP パケットがある場合、どれが「メイン」の XMP か判断できず、複合文書のリソースの包含階層全体を判断できない場合があります。
- すでに使用されていないパケットが存在している可能性があります。例えば、PDF ファイルでは増分保存が可能です。そのため、変更が加えられた文書には複数のパケットが存在していることがありますが、ファイルの現状を反映しているのはそのうちの 1 つのみです。

スキャニングのヒント

有効なヘッダが見つかるまで、ファイルを 1 バイトずつスキャンする必要があります。スキャンプログラムでは、まず、次のテキストを表すバイトパターンを探す必要があります。

```
<?xpacket begin=
```

そのバイトパターンは、次のいずれかになります。

- 8 ビットエンコーディング（UTF-8、ASCII 7 ビット、ISOLatin-1）：

```
0x3C 0x3F 0x78 0x70 0x61 0x63 0x6B
0x65 0x74 0x20 0x62 0x65 0x67 0x69 0x6E 0x3D
```

- 16 ビットエンコーディング（UCS-2、UTF-16）：（ビッグエンディアンおよびリトルエンディアン）

```
0x3C 0x00 0x3F 0x00 0x78 0x00 0x70 0x00 0x61 0x00
0x63 0x00 0x6B 0x00 0x65 0x00 0x74 0x00 0x20 0x00 0x62 0x00
0x65 0x00 0x67 0x00 0x69 0x00 0x6E 0x00 0x3D [0x00]
```

- 32 ビットエンコーディング（UCS-4）：上の UCS-2 のパターンと似ていますが、UCS-2 では 0x00 バイトが 1 つずつであったのに対し、このパターンでは 0x00 バイトが 3 つになります。

16 ビットエンコーディングの場合、0x00 という値が文字の上位バイトなのか下位バイトなのかは、バイトオーダーのマーク（begin 属性の値）を読み込むまで判断できません。バイトオーダーがどちらであってもこのパターンは最初が非ゼロ値から始まるので、上にも示されているように、終わりが 0x00 値になる場合とそうでない場合があります。

スキャンプログラムでは、0x00 の値を単純に無視して、8 ビットパターンを検索することもできます。バイトオーダーが判別できたら、バイト単位ではなく文字単位で読み取るようにしてください。

一致するバイトパターンを見つけたら、スキャンプログラムは引用符を 1 つ読み込む必要があります。これは、一重引用符（アポストロフィ）（U+0027）か二重引用符（U+0022）のいずれかになります。

注意： 処理命令の個々の属性値には、一重引用符か二重引用符のいずれかが付いています。次のヘッダは整形式です。

```
<?xpacket begin="■" id='W5M0MpCehiHzreSzNTczkc9d'?>
```

これで、begin 属性の値を読み込む準備ができました。この値の後には閉じる引用符が続きます。

UTF-8：	0xEF 0xBB 0xBF
UTF-16、ビッグエンディアン：	0xFE 0xFF
UTF-16、リトルエンディアン：	0xFF 0xFE
UTF-32、ビッグエンディアン：	0x00 0x00 0xFE 0xFF
UTF-32、リトルエンディアン：	0xFF 0xFE 0x00 0x00

注意： 属性に値がない場合のエンコーディングは UTF-8 です。

これで、ヘッダの残りの部分を適切な文字エンコーディングで処理するための情報が得られました。

メタデータの外部ストレージ

これは必須の要件ではありませんが、XMP メタデータはそのメタデータの対象ファイルの中に（XMP パケットとして）埋め込むことをお勧めします。ただし、データベースストレージモデルである場合や、ファイルサイズが極端に大きくなる場合、フォーマットやアクセスの問題がある場合など、メタデータの埋め込みが不適切であったり不可能であったりすることもあります。インターネットで頻繁に転送することを想定している小さなコンテンツの場合は、埋め込みメタデータによる負担が大きすぎることもあります。ビデオや音声のアーカイブシステムには、メタデータを表現する方法がないこともあります。また、ハイエンドデジタルカメラの中には、拡張不能な独自のファイル形式で画像の生データを保持し、別のファイルとして EXIF メタデータを保存しているものもあります。

メタデータをコンテンツとは別の場所に保存すると、メタデータをなくす危険性があります。また、コンテンツが保存されているファイルとメタデータとを関連付ける方法も検討する必要があります。アプリケーションでは、次のことを行う必要があります。

- 整形式の XML 文書として外部ファイルを書き出します。その際、最初の XML 宣言も含めます。
- ファイルの拡張子は .xmp にします。Mac OS の場合は、必要に応じてファイルのタイプを「TEXT」に設定します。
- MIME タイプが必要な場合は、application/rdf+xml を使用します。
- 埋め込まれていた XMP パケットを後処理プログラムで抽出して連結したのと同じような状態になるように、外部メタデータを書き込みます。
- 可能であれば、rdf:about 属性のインスタンス ID を XMP の対象ファイル中に配置して、そのフォーマットに対応したアプリケーションがメタデータを確認できるようにします。

アプリケーションで外部 XMP ファイルを検索する場合は、メイン文書と同じ名前で拡張子が .xmp のファイルを、同じディレクトリ内で検索します （このファイルはサイドカー XMP ファイルと呼ばれます）。

3 XMP ストレージモデル
メタデータの外部ストレージ

4 XMP スキーマ

この章の内容は、次のとおりです。

- 標準 XMP スキーマの定義
  - 35 ページの「Dublin Core スキーマ」
  - 36 ページの「XMP Basic スキーマ」
  - 38 ページの「XMP Rights Management スキーマ」
  - 39 ページの「XMP Media Management スキーマ」
  - 42 ページの「XMP Basic Job Ticket スキーマ」
  - 43 ページの「XMP Paged-Text スキーマ」
- 特定のものを対象としているスキーマの定義
  - 44 ページの「Adobe PDF スキーマ」
  - 45 ページの「Photoshop スキーマ」
  - 46 ページの「EXIF スキーマ」
- スキーマで使用されるプロパティ値の定義および説明（60 ページの「プロパティ値の型」）
- XMP を拡張するためのガイドライン（68 ページの「スキーマの拡張」）

XMP メタデータでは、一般に、1 つまたは複数のスキーマのプロパティが使用されます。例えば、多くのアドビアプリケーションで共通してよく使用されるプロパティには、次のようなものがあります。

- Dublin Core スキーマ：dc:title、dc:creator、dc:description、dc:subject、dc:format、dc:rights
- XMP Basic スキーマ：xmp:CreateDate、xmp:CreatorTool、xmp:ModifyDate、xmp:MetadataDate
- XMP Rights Management スキーマ：xmpRights:WebStatement、xmpRights:Marked
- XMP Media Management スキーマ：xmpMM:DocumentID

XMP スキーマの定義

この章のそれぞれのスキーマ定義では、そのスキーマを識別するための名前空間文字列と、そのスキーマの名前空間接頭辞として推奨するものを示したあとに、そのスキーマで定義されているすべてのプロパティの一覧表を掲載します。それぞれの表には、次の列があります。

- **プロパティ**：プロパティの名前。推奨される名前空間接頭辞も含めて記載しています。
- **値の型**：プロパティの値の型。60 ページの「プロパティ値の型」に記載されているそれぞれの型へのリンクが付いています。配列型の場合は、値の前にコンテナの種類（alt、bag、または seq）が付きます （詳しくは、12 ページの「配列」を参照してください）。
- **カテゴリ**：スキーマのプロパティには、内部と外部の 2 つがあります。
  - 内部メタデータは、アプリケーションが管理する必要があります。内部メタデータには、システムレベルの情報（変更日など）や、編集アプリケーションによってアクセスされる情報（文書内の単語数など）があります。xmp:ModifyDate はその一例です。このようなプロパティは、ユーザには変更させないようにしてください。アプリケーションは、ファイルを保存する際に、すべての内部プロパティに対して有効な値を設定する必要があります。内部プロパティの値を設定しない場合は、既存の値をすべて廃棄してください。
  - 外部メタデータは、ユーザが設定する必要があります。これは、文書のコンテンツからは独立しています。編集アプリケーションは、外部データへの変更を表示する必要がありますが、外部データに影響を与えることはありません。ユーザが変更しない限り、外部プロパティは出力にそのまま保持されます。dc:creator はその一例です。
- **説明**：プロパティの説明。

注意： いくつかの XMP プロパティは、以前のバージョンから推奨されなくなっています。それらは互換性を保つために定義されていますが、今後は使用しないようにしてください。

注意： この仕様の以前のバージョンでは、エイリアスプロパティも記載していました。特定の XMP 実装では、あるスキーマのプロパティを別のスキーマのプロパティと同等なものとして扱っている場合があります。しかし、交換性を促進するため、アプリケーションでは常に標準の「基本」形式のプロパティを記述する必要があります。このバージョンの仕様では、基本プロパティのみをリストしています。

スキーマには、さまざまなプロパティが定義されています。XMP のプロパティは、次のいずれかの状態になります。

- 未指定、つまり値を持っていない状態です。初めて値が定義されるまで、プロパティは未指定の状態になります。
- 指定済み、つまり値が定義されている状態です。

  注意： 指定済みのプロパティは、その値が空の文字列であることもありますが、これは未指定のプロパティとは異なります。ただし、プロパティに空の文字列値は設定しないようお勧めします。

どの XMP でも、特定のスキーマのすべてのプロパティを指定済みにする必要はありません。構造化されているプロパティも、すべてのフィールドを指定済みにする必要はありません（スキーマで特に要請されていない限り）。

Dublin Core スキーマ

Dublin Core スキーマには、よく使用される一連のプロパティが含まれています。

- このスキーマの名前空間の URI は、`http://purl.org/dc/elements/1.1/` です。
- このスキーマの推奨される名前空間接頭辞は、`dc` です。

プロパティ	値の型	カテゴリー	説明
dc:contributor	bag ProperName	外部	リソースの内容に協力した人（作成者は除きます）。
dc:coverage	Text	外部	リソースの範囲またはスコープ。
dc:creator	seq ProperName	外部	リソースの作成者（序列が重要な場合は、その順に従ってリストします）。
dc:date	seq Date	外部	リソースに関して何らかの重要な事が起こった日付。
dc:description	Lang Alt	外部	リソースのコンテンツの説明。複数の言語に対応するために、複数の値が用意されることもあります。
dc:format	MIMEType	内部	リソースの保存時に使用したファイル形式。ツールやアプリケーションを使用して、このプロパティにデータの保存形式を設定します。適切な修飾子を含めることもできます。
dc:identifier	Text	外部	リソースの一意の識別子。
dc:language	bag Locale	内部	リソースで使用されている言語を示す無順序配列。
dc:publisher	bag ProperName	外部	提供者
dc:relation	bag Text		他の文書との関連。
dc:rights	Lang Alt	外部	非公式の権利の言明。言語別の値。
dc:source	Text	外部	このリソースの派生元のリソースを示す一意の識別子。
dc:subject	bag Text	外部	リソースのコンテンツの主題を表すフレーズまたはキーワードの無順序配列。
dc:title	Lang Alt	外部	文書のタイトル、またはリソースに与えられた名前。通常は、リソースの公式な名前を表します。
dc:type	bag open Choice	外部	文書の種類。小説、詩、研究報告書など。

XMP Basic スキーマ

XMP Basic スキーマには、基本的な説明情報を提供するプロパティが含まれています。

- このスキーマの名前空間の URI は、`http://ns.adobe.com/xap/1.0/` です。
- このスキーマの推奨される名前空間接頭辞は、`xmp` です。

プロパティ	値の型	カテゴリー	説明
xmp:Advisory	bag XPath	外部	オーサリングアプリケーションの外部で編集されたプロパティを示す無順序配列。 それぞれのアイテムには、1 つの ASCII スペース（U+0020）で区切られた名前空間と XPath が 1 つ含まれます。
xmp:BaseURL	URL	内部	文書のコンテンツで使用されている相対 URL のベース URL。この文書にインターネットのリンクが含まれており、それらが相対リンクの場合は、このベース URL に対する相対リンクとなります。 このプロパティは、埋め込まれた相対 URL をツールが解釈するための標準的な方法を提供します。Web オーサリングツールでは、URL をどのように解釈してもらいたいかに基づいてこの値を設定する必要があります。
xmp:CreateDate	Date	外部	リソースが最初に作成された日時。
xmp:CreatorTool	AgentName	内部	リソースの作成に使用されたツールのうち、最初に確認されたものの名前。メタデータに履歴がある場合、この値は xmpMM:History の softwareAgent プロパティの値と等しくなる必要があります。
xmp:Identifier	bag Text	外部	指定のコンテキストにおいて、リソースの明確な識別を可能にするテキスト文字列の無順序配列。配列要素が xmpidq:Scheme で修飾され、その識別子が従っている正式な識別システムを示している場合もあります。 **注意：** dc:identifier プロパティにはスキーム修飾子が定義されておらず、XMP 仕様では単純型の（単一値の）プロパティとして定義されているので、このプロパティは使用されていません。
xmp:MetadataDate	Date	内部	このリソースのメタデータが最後に変更された日時。このプロパティには xmp:ModifyDate と同じか、より最近の値が設定されている必要があります。

プロパティ	値の型	カテゴリー	説明
xmp:ModifyDate	Date	内部	リソースが最後に変更された日時。 **注意：** このプロパティの値は、ファイルが保存される前に設定されるので、システムのファイル変更日時とは必ずしも一致しません。
xmp:Nickname	Text	外部	リソースの短い非公式名。
xmp:Thumbnails	alt Thumbnail	内部	ファイルのサムネール画像の択一配列。それぞれのサムネール画像は、サイズや画像エンコーディングなどの特性が異なる場合があります。

xmp:Identifier 配列の要素が xmpidq:Scheme で修飾され、その識別子が従っている正式な識別システムを示している場合もあります。

- この修飾子の名前空間の URI は、`http://ns.adobe.com/xmp/Identifier/qual/1.0/` です。
- この修飾子の推奨される名前空間接頭辞は、`xmpidq` です。

修飾子	値の型	カテゴリー	説明
xmpidq:Scheme	Text	外部	関連する xmp:Identifier アイテムの値で使用されている、正式な識別体系の名前。

XMP Rights Management スキーマ

このスキーマには、権利の管理に関するプロパティが含まれています。これらのプロパティでは、リソースに関する法律上の制限事項に関連する情報を指定します。

注意： XMP は、権利の行使を行うメカニズムではありません。

- このスキーマの名前空間の URI は、`http://ns.adobe.com/xap/1.0/rights/` です。
- このスキーマの推奨される名前空間接頭辞は、xmpRights です。

プロパティ	値の型	カテゴリー	説明
xmpRights:Certificate	URL	外部	オンラインの権利管理証明書。
xmpRights:Marked	Boolean	外部	このリソースで権利が管理されているかどうか。
xmpRights:Owner	bag ProperName	外部	リソースの法律上の所有者を示す無順序配列。
xmpRights:UsageTerms	Lang Alt	外部	リソースを合法的に使用する方法を説明するテキスト。
xmpRights:WebStatement	URL	外部	このリソースの所有者や権利に関する説明が掲載されている Web ページの場所。

XMP Media Management スキーマ

XMP Media Management スキーマは、主にデジタル資産管理（DAM）システムで使用されます。

xmpMM:ManagedFrom、xmpMM:Manager、xmpMM:ManageTo、xmpMM:ManageUI、xmpMM:ManagerVariant の各プロパティは、DAM システムによって「所有」されており、それぞれ指示に従ってアプリケーションで設定する必要があります。これらのプロパティは、管理対象外のファイルでは使用しないようにしてください。

xmpMM:DerivedFrom、xmpMM:DocumentID、xmpMM: RenditionClass、xmpMM: RenditionParams、xmpMM: VersionID、xmpMM:Versions の各プロパティは、管理対象ファイルのために DAM システムによって所有されていますが、アプリケーションが管理対象外のファイルに対して使用しても構いません。

xmpMM:History プロパティは、常にアプリケーションが所有しています。

- このスキーマの名前空間の URI は、`http://ns.adobe.com/xap/1.0/mm/` です。
- このスキーマの推奨される名前空間接頭辞は、`xmpMM` です。

プロパティ	値の型	カテゴリー	説明
xmpMM:DerivedFrom	ResourceRef	内部	この文書の派生元の文書への参照。これは最小限の参照で、指定されていない要素は変更されていないものと見なされます。例えば、新しいバージョンの場合は、以前のバージョンのインスタンス ID とバージョン番号を指定すれば十分なことがあります。また、レンディション[訳注] の場合は、派生元のインスタンス ID とレンディションクラスを指定すれば十分なことがあります。
xmpMM:DocumentID	URI	内部	文書のすべてのバージョンおよびレンディションでの共通の識別子。これは、UUID に基づいたものにしてください。19 ページの「rdf:about 属性」の説明を参照してください。
xmpMM:History	seq ResourceEvent	内部	リソースが現在の状態になるまでに行われた高レベルのユーザアクションを示す順序配列。これは、前のバージョンから現在のバージョンに至るまでに行われた操作の概要を、人が読んで分かる形で提示することを目的としています。このリストは、要約レベルにしてください。キーストロークやその他の細かい操作を逐一記録することを目的としたものではありません。

プロパティ	値の型	カテゴリー	説明
xmpMM:ManagedFrom	ResourceRef	内部	管理対象になる前の文書への参照。これは、現在所有されていない文書が資産管理システムに管理対象として登録されるときに設定されます。別の管理システムへの参照が含まれていることもあります。
xmpMM:Manager	AgentName	内部	このリソースを管理している資産管理システムの名前。アプリケーションは、このプロパティと xmpMM: ManagerVariant を調べて、この文書に関してどの資産管理システムにコンタクトすればよいかを判断します。
xmpMM:ManageTo	URI	内部	資産管理システムで管理対象リソースを識別するための URI。そのリソースが管理対象になっていることを正式に示すには、このプロパティを設定します。この URI の形式およびコンテンツは、その資産管理システムに固有のものです。
xmpMM:ManageUI	URI	内部	Web ブラウザで管理対象リソースの情報にアクセスする際に使用できる URI。ブラウザにカスタムプラグインが必要な場合もあります。
xmpMM: ManagerVariant	Text	内部	その資産管理システムに特有のバリアント。このプロパティの形式と内容は、その資産管理システムに固有のものです。
xmpMM: RenditionClass	RenditionClass	内部	このリソースのレンディションクラス名。文書のバージョンがレンディションではない場合、このプロパティは未指定または `default` に設定されている必要があります。
xmpMM: RenditionParams	Text	内部	xmpMM: RenditionClass にエンコーディングするには複雑すぎたり長すぎたりする追加のレンディションパラメータを指定するのに使用できます。
xmpMM: VersionID	Text	内部	このリソースの文書バージョンの識別子。 文書の各バージョンには、新しい識別子が付けられます。通常は、1、2、3 のように、単純に整数が増加していきます。メディア管理システムでは、他の表記方法や、より複雑な分岐処理をサポートすることもできます。

プロパティ	値の型	カテゴリー	説明
xmpMM:Versions	seq Version	内部	このリソースに関連するバージョン履歴。 この文書の最も古い既知のバージョンはエントリ [1] で、最も新しいバージョンはエントリ [last()] です。 通常は、メディア管理システムがチェックイン時にメタデータにバージョン情報を設定します。 xmpMM:Versions プロパティに、最初のバージョンから現在のバージョンに至るまでの完全な履歴が保持されることは、保証されていません。内部バージョン情報は圧縮されたり削除されたりすることがあるので、バージョン履歴もある時点で切り捨てられる場合があります。
xmpMM:LastURL（非推奨）	URL	内部	プライバシー保護のために推奨されなくなりました。
xmpMM:RenditionOf（非推奨）	ResourceRef	内部	xmpMM:DerivedFrom に取って代わられたため、これは非推奨となりました。このレンディションの派生元の文書への参照です。
xmpMM:SaveID（非推奨）	Integer	内部	推奨されなくなりました。以前は、xmpMM:LastURL プロパティをサポートするためにのみ使用されていました。

訳注：元の文書から特定フォーマットへの変換結果です。同じ文書内容の一表示形式を意味します。

XMP Basic Job Ticket スキーマ

このスキーマは、非常に単純なワークフローまたはジョブ情報を表します。

- このスキーマの名前空間の URI は、`http://ns.adobe.com/xap/1.0/bj/` です。
- このスキーマの推奨される名前空間接頭辞は、`xmpBJ` です。

プロパティ	値の型	カテゴリー	説明
xmpBJ:JobRef	bag Job	外部	この文書を使用しているジョブ処理の外部ジョブ管理ファイルへの参照。ジョブ名の使用はユーザにゆだねられています。特定のジョブまたは契約に含まれているすべての文書を識別するためによく使用されます。 複数の値があるのは、特定の文書が複数のジョブで使用される場合があるからです。以前にどのジョブで文書が使用されていたかに関する履歴情報を保存しておくと便利な場合があります。

XMP Paged-Text スキーマ

Paged-Text スキーマは、文書の 1 つのページに表示されるテキストに使用されます。

- このスキーマの名前空間の URI は、`http://ns.adobe.com/xap/1.0/t/pg/` です。
- このスキーマの推奨される名前空間接頭辞は、`xmpTPg` です。

プロパティ	値の型	カテゴリー	説明
xmpTPg:MaxPageSize	Dimensions	内部	文書内で最も大きいページのサイズ（内包されている文書も含みます）。
xmpTPg:NPages	Integer	内部	文書内のページの数（内包されている文書も含みます）。

Adobe PDF スキーマ

このスキーマは、Adobe PDF ファイルで使用されるプロパティを示します。

- このスキーマの名前空間の URI は、`http://ns.adobe.com/pdf/1.3/` です。
- このスキーマの推奨される名前空間接頭辞は、`pdf` です。

プロパティ	値の型	カテゴリー	説明
pdf:Keywords	Text	外部	キーワード
pdf:PDFVersion	Text	内部	PDF ファイルのバージョン（1.0、1.3 など）。
pdf:Producer	AgentName	内部	PDF 文書を作成したツールの名前。

Photoshop スキーマ

このスキーマは、Adobe Photoshop で使用されるプロパティを示します。

- このスキーマの名前空間の URI は、`http://ns.adobe.com/photoshop/1.0/` です。
- このスキーマの推奨される名前空間接頭辞は、`photoshop` です。

プロパティ	値の型	カテゴリー	説明
photoshop:AuthorsPosition	Text	外部	作成者の役割
photoshop:CaptionWriter	ProperName	外部	記入者／編集者
photoshop:Category	Text	外部	カテゴリー。7 ビット ASCII 文字 3 文字で指定されます。
photoshop:City	Text	外部	市
photoshop:Country	Text	外部	国またはそれに準じた地名。
photoshop:Credit	Text	外部	クレジット
photoshop:DateCreated	Date	外部	文書の知的コンテンツが作成された日付（物理的な実体が作成された日付ではありません）。IIM に従って表記します。例えば、アメリカの南北戦争のときに撮影された写真の場合は、写真をアーカイブするためにデジタル化した日付ではなく、その時期（1861 年～ 1865 年）の作成日を設定します。
photoshop:Headline	Text	外部	ヘッドライン
photoshop:Instructions	Text	外部	追加の説明
photoshop:Source	Text	外部	ソース
photoshop:State	Text	外部	都道府県や州
photoshop: SupplementalCategories	Text	外部	追加カテゴリー
photoshop: TransmissionReference	Text	外部	元の伝送への参照。
photoshop:Urgency	Integer	外部	重要度。有効範囲は 1 ～ 8 です。

EXIF スキーマ

EXIF は、デジタルカメラで広く使用されている、画像ファイル用のメタデータ標準です。EXIF 2.2 の仕様は、http://www.exif.org/specifications.html で入手できます。

次の各節で、EXIF 2.2 仕様の各部分に対応する 2 つの XMP スキーマについて説明します。

- 46 ページの「TIFF プロパティ用の EXIF スキーマ」
- 48 ページの「EXIF 固有のプロパティ用の EXIF スキーマ」

プロパティの説明では、読者が EXIF メタデータについてある程度知識を持っていることを前提としています。XMP のプロパティ名は、EXIF 仕様で使われている名前と同じです。プロパティについて詳しくは、EXIF 仕様を参照してください。

また、次の各節で、詳細情報を提供します。

- 57 ページの「データの表現および変換」では、XMP 形式と EXIF 形式の間で変換を行うためのガイドラインを、例を示しながら説明します。
- 65 ページの「EXIF スキーマの値の型」では、EXIF 固有の値の型について説明します。

注意： Date 型の XMP プロパティには、1 秒未満の値も含まれています。したがって、1 秒未満の値を表す EXIF のプロパティ（`SubSecTime`、`SubSecTimeOriginal`、`SubSecTimeDigitized`）は、「メイン XMP プロパティ」に含まれています。

TIFF プロパティ用の EXIF スキーマ

次の表に、TIFF に関連するデータのプロパティを示します。表に含まれているのは、EXIF 2.2 仕様に含まれている TIFF プロパティのみです。

- このスキーマ名は、`http://ns.adobe.com/tiff/1.0/` です。
- このスキーマの推奨される名前空間接頭辞は、`tiff` です。

プロパティ	値の型	カテゴリー	説明
tiff:ImageWidth	Integer	内部	TIFF タグ 256（0x100）。画像の幅（ピクセル）。
tiff:ImageLength	Integer	内部	TIFF タグ 257（0x101）。画像の高さ（ピクセル）。
tiff:BitsPerSample	seq Integer	内部	TIFF タグ 258（0x102）。各チャンネルでの 1 成分当たりのビット数。
tiff:Compression	Closed Choice of Integer	内部	TIFF タグ 259（0x103）。圧縮方式：1 = 非圧縮、6 = JPEG。
tiff: PhotometricInterpretation	Closed Choice of Integer	内部	TIFF タグ 262（0x106）。ピクセルの構成：2 = RGB、6 = YCbCr。

プロパティ	値の型	カテゴリー	説明
tiff:Orientation	Closed Choice of Integer	内部	TIFF タグ 274（0x112）。方向： 1 = 0 行目は上端、0 列目は左端 2 = 0 行目は上端、0 列目は右端 3 = 0 行目は下端、0 列目は右端 4 = 0 行目は下端、0 列目は左端 5 = 0 行目は左端、0 列目は上端 6 = 0 行目は右端、0 列目は上端 7 = 0 行目は右端、0 列目は下端 8 = 0 行目は左端、0 列目は下端
tiff:SamplesPerPixel	Integer	内部	TIFF タグ 277（0x115）。1 ピクセル当たりの成分数。
tiff:PlanarConfiguration	Closed Choice of Integer	内部	TIFF タグ 284（0x11C）。 データの並び：1 = 点順次 (chunky)、2 = 面順次 (planar)。
tiff:YCbCrSubSampling	Closed Choice of seq Integer	内部	TIFF タグ 530（0x212）。色差成分のサンプリング率。 [2, 1] = YCbCr4:2:2 [2, 2] = YCbCr4:2:0
tiff:YCbCrPositioning	Closed Choice of Integer	内部	TIFF タグ 531（0x213）。輝度成分に対する色差成分の相対位置。1 = 中心、2 = 一致
tiff:XResolution	Rational	内部	TIFF タグ 282（0x11A）。水平解像度（ピクセル / 指定の単位）。
tiff:YResolution	Rational	内部	TIFF タグ 283（0x11B）。垂直解像度（ピクセル / 指定の単位）。
tiff:ResolutionUnit	Closed Choice of Integer	内部	TIFF タグ 296（0x128）。XResolution および YResolution で使用する単位。2 = インチ、あるいは 3 = センチメートルのいずれか。
tiff:TransferFunction	seq Integer	内部	TIFF タグ 301（0x12D）。画像のトランスファ関数。3 × 256 エントリの表形式で記述します。
tiff:WhitePoint	seq Rational	内部	TIFF タグ 318（0x13E）。白色点の色度。
tiff:PrimaryChromaticities	seq Rational	内部	TIFF タグ 319（0x13F）。3 原色の色度。
tiff:YCbCrCoefficients	seq Rational	内部	TIFF タグ 529（0x211）。RGB を YCbCr に変換する変換行列の係数。
tiff:ReferenceBlackWhite	seq Rational	内部	TIFF タグ 532（0x214）。参照黒色点の値および参照白色点の値。

プロパティ	値の型	カテゴリー	説明
tiff:DateTime	Date	内部	TIFF タグ 306（0x132）（プライマリ）および EXIF タグ 37520（0x9290）（1 秒未満）。画像の作成日時（EXIF にはタイムゾーンはありません）。元の EXIF 形式ではなく、ISO 8601 形式で保存されます。このプロパティには、EXIF SubsecTime 属性の値が含まれます。 **注意：** このプロパティは、xmp:ModifyDate として XMP に保存されます。
tiff:ImageDescription	Lang Alt	外部	TIFF タグ 270（0x10E）。画像の説明。 **注意：** このプロパティは、dc:description として XMP に保存されます。
tiff:Make	ProperName	内部	TIFF タグ 271（0x10F）。記録機器の製造元。
tiff:Model	ProperName	内部	TIFF タグ 272（0x110）。機器のモデル名または番号。
tiff:Software	AgentName	内部	TIFF タグ 305（0x131）。画像の生成に使用したソフトウェアまたはファームウェア。 **注意：** このプロパティは、xmp:CreatorTool として XMP に保存されます。
tiff:Artist	ProperName	外部	TIFF タグ 315（0x13B）。カメラの所有者、撮影者、または画像の作成者。 **注意：** EXIF 文字列中の各エントリは、dc:creator プロパティの中の個別のエントリになります。EXIF 文字列に戻す場合は、dc:creator プロパティの各エントリをセミコロンで区切る必要があります。
tiff:Copyright	Lang Alt	外部	TIFF タグ 33432（0x8298）。著作権情報。 **注意：** このプロパティは、dc:rights として XMP に保存されます。

EXIF 固有のプロパティ用の EXIF スキーマ

次の表に、EXIF でのみ定義されているプロパティを示します。

注意： EXIF 2.2 のプロパティの多くは、XMP には含まれていません。除外されているプロパティは、主に画像ストリームに直接関連するプロパティ、つまり、画像ストリームにアクセスしない場合にはほとんど利用価値がないプロパティです。XMP では、主要なファイルのコンテンツから独立してそれ単独で利用価値があるものを XMP メタデータにするのが原則となっています。除外されているプロパティには、StripOffsets、RowsPerStrip、StripByteCounts、JPEGInterchangeFormat、JPEGInterchangeFormatLength などがあります。

注意：「GPS」で始まるプロパティは、GPS プロパティです。これは DIG-35 でも使用され、JPEG-2000 標準の一部になっています。

- このスキーマの名前空間の URI は、`http://ns.adobe.com/exif/1.0/` です。
- このスキーマの推奨される名前空間接頭辞は、`exif` です。

プロパティ	値の型	カテゴリー	説明
exif:ExifVersion	Closed Choice of Text	内部	EXIF タグ 36864（0x9000）。バージョン番号。「0210」にする必要があります。
exif:FlashpixVersion	Closed Choice of Text	内部	EXIF タグ 40960（0xA000）。FlashPix のバージョン。「0100」にする必要があります。
exif:ColorSpace	Closed Choice of Integer	内部	EXIF タグ 40961（0xA001）。カラースペース情報： 1 = sRGB -32786 = 未較正
exif:ComponentsConfiguration	Closed Choice of seq Integer	内部	EXIF タグ 37121（0x9101）。データの成分構成：4 5 6 0（RGB 圧縮データの場合）、1 2 3 0（その他の場合）。 0 = 存在しない 1 = Y 2 = Cb 3 = Cr 4 = R 5 = G 6 = B
exif:CompressedBitsPerPixel	Rational	内部	EXIF タグ 37122（0x9102）。圧縮画像に使用している圧縮モード（ビット / ピクセル）。
exif:PixelXDimension	Integer	内部	EXIF タグ 40962（0xA002）。画像の有効幅（ピクセル）。
exif:PixelYDimension	Integer	内部	EXIF タグ 40963（0xA003）。画像の有効高さ（ピクセル）。
exif:MakerNote	Text	内部	EXIF タグ 37500（0x927C）。EXIF 内の未定義情報。EXIF 仕様の UNDEFINED 型です。
exif:UserComment	Lang Alt	外部	EXIF タグ 37510（0x9286）。ユーザのコメント。
exif:RelatedSoundFile	Text	内部	EXIF タグ 40964（0xA004）。関連する音声ファイルの名前（「8.3」の形式）。

プロパティ	値の型	カテゴリー	説明
exif:DateTimeOriginal	Date	内部	EXIF タグ 36867（0x9003）（プライマリ）および 37521（0x9291）（1 秒未満）。元の画像が生成された日時（ISO 8601 形式）。EXIF の `SubsecTimeOriginal` データを含みます。
exif:DateTimeDigitized	Date	内部	EXIF タグ 36868（0x9004）（プライマリ）および 37522（0x9292）（1 秒未満）。画像がデジタルデータとして保存された日時。最初からデジタル形式で保存された場合は、DateTimeOriginal と同じである場合もあります。ISO 8601 形式で保存されます。EXIF の `SubsecTimeDigitized` データを含みます。
exif:ExposureTime	Rational	内部	EXIF タグ 33434（0x829A）。露出時間（秒）。
exif:FNumber	Rational	内部	EXIF タグ 33437（0x829D）。F 値。
exif:ExposureProgram	Closed Choice of Integer	内部	EXIF タグ 34850（0x8822）。使用した露出プログラムのクラス。 0 = 未定義 1 = マニュアル 2 = 通常のプログラム 3 = 絞り優先 4 = シャッター優先 5 = 深度優先 6 = スポーツ 7 = ポートレート 8 = 風景
exif:SpectralSensitivity	Text	内部	EXIF タグ 34852（0x8824）。各チャンネルのスペクトル感度。
exif:ISOSpeedRatings	seq Integer	内部	EXIF タグ 34855（0x8827）。ISO 12232 で規定されている、入力機器の ISO Speed および ISO Latitude。
exif:OECF	OECF/SFR	内部	EXIF タグ 34856（0x8828）。ISO 14524 で規定されている光電変換機能（Opto-Electoric Conversion Function）。
exif:ShutterSpeedValue	Rational	内部	EXIF タグ 37377（0x9201）。シャッタースピード（単位：APEX）。EXIF 仕様の「Annex C」を参照してください。
exif:ApertureValue	Rational	内部	EXIF タグ 37378（0x9202）。レンズの絞り値（単位：APEX）。

プロパティ	値の型	カテゴリー	説明
exif:BrightnessValue	Rational	内部	EXIF タグ 37379（0x9203）。輝度値（単位：APEX）。
exif:ExposureBiasValue	Rational	内部	EXIF タグ 37380（0x9204）。露出補正値（単位：APEX）。
exif:MaxApertureValue	Rational	内部	EXIF タグ 37381（0x9205）。レンズ最小 F 値（単位：APEX）。
exif:SubjectDistance	Rational	内部	EXIF タグ 37382（0x9206）。被写体までの距離（メートル）。
exif:MeteringMode	Closed Choice of Integer	内部	EXIF タグ 37383（0x9207）。測光方式： 0 = 不明 1 = 平均 2 = 中央重点 3 = スポット 4 = マルチスポット 5 = 分割測光 6 = 部分測光 255 = その他
exif:LightSource	Closed Choice of Integer	内部	EXIF タグ 37384（0x9208）。光源： 0 = 不明 1 = 太陽光 2 = 蛍光灯 3 = タングステン 17 = 標準光 A 18 = 標準光 B 19 = 標準光 C 20 = D55 21 = D65 22 = D75 255 = その他
exif:Flash	Flash	内部	EXIF タグ 37385（0x9209）。ストロボ光（フラッシュ）の光源のデータ。
exif:FocalLength	Rational	内部	EXIF タグ 37386（0x920A）。レンズの焦点距離（ミリメートル）。
exif:SubjectArea	seq Integer	内部	EXIF タグ 37396（0x9214）。シーン全体における主な被写体の位置および領域。
exif:FlashEnergy	Rational	内部	EXIF タグ 41483（0xA20B）。画像撮影時のストロボの強度。
exif:SpatialFrequencyResponse	OECF/SFR	内部	EXIF タグ 41484（0xA20C）。ISO 12233 で規定されている、入力デバイスの空間周波数テーブルおよび SFR 値。

プロパティ	値の型	カテゴリー	説明
exif: FocalPlaneXResolution	Rational	内部	EXIF タグ 41486（0xA20E）。焦点面の水平解像度（ピクセル / 指定の単位）。
exif:FocalPlaneYResolution	Rational	内部	EXIF タグ 41487（0xA20F）。焦点面の垂直解像度（ピクセル / 指定の単位）。
exif: FocalPlaneResolutionUnit	Closed Choice of Integer	内部	EXIF タグ 41488（0xA210）。FocalPlaneXResolution および FocalPlaneYResolution で使用する単位。 2 = インチ 3 = センチメートル
exif:SubjectLocation	seq Integer	内部	EXIF タグ 41492（0xA214）。シーンの主な被写体の位置。最初の値は、主な被写体がある位置の水平方向のピクセル、2 番目の値は垂直方向のピクセルです。
exif:ExposureIndex	Rational	内部	EXIF タグ 41493（0xA215）。入力機器の露光指数。
exif:SensingMethod	Closed Choice of Integer	内部	EXIF タグ 41495（0xA217）。入力機器の画像センサのタイプ。 1 = 未定義 2 = 単板式カラーエリアセンサ 3 = 2 板式カラーエリアセンサ 4 = 3 板式カラーエリアセンサ 5 = 色順次エリアセンサ 7 = 3 ラインセンサ 8 = 色順次ラインセンサ
exif:FileSource	Closed Choice of Integer	内部	EXIF タグ 41728（0xA300）。画像のソースを示します。選択できるのは 3（DSC）のみです。
exif:SceneType	Closed Choice of Integer	内部	EXIF タグ 41729（0xA301）。シーンのタイプを示します。選択できるのは 1（直接撮影された画像）のみです。
exif:CFAPattern	CFAPattern	内部	EXIF タグ 41730（0xA302）。画像センサのカラーフィルタ配列の幾何学的パターン。
exif:CustomRendered	Closed Choice of Integer	内部	EXIF タグ 41985（0xA401）。画像データに対して特別な処理を使用していることを示します。 0 = 通常の処理 1 = カスタムの処理

プロパティ	値の型	カテゴリー	説明
exif:ExposureMode	Closed Choice of Integer	内部	EXIF タグ 41986（0xA402）。画像撮影時に設定されていた露出モードを示します。 0 = 自動露出 1 = マニュアル露出 2 = オートブラケット
exif:WhiteBalance	Closed Choice of Integer	内部	EXIF タグ 41987（0xA403）。画像撮影時に設定されていたホワイトバランスモードを示します。 0 = 自動ホワイトバランス 1 = マニュアルホワイトバランス
exif:DigitalZoomRatio	Rational	内部	EXIF タグ 41988（0xA404）。画像撮影時に設定されていたデジタルズーム率を示します。
exif:FocalLengthIn35mmFilm	Integer	内部	EXIF タグ 41989（0xA405）。35mm フィルムカメラに換算した焦点距離（mm）を示します。値が 0 の場合は、焦点距離が不明であることを表します。このタグは、FocalLength タグとは異なることに注意してください。
exif:SceneCaptureType	Closed Choice of Integer	内部	EXIF タグ 41990（0xA406）。撮影されたシーンのタイプを示します。 0 = 標準 1 = 風景 2 = ポートレート 3 = 夜景
exif:GainControl	Closed Choice of Integer	内部	EXIF タグ 41991（0xA407）。画像全体のゲイン制御の度合いを示します。 0 = なし 1 = ローゲインを上げる 2 = ハイゲインを上げる 3 = ローゲインを下げる 4 = ハイゲインを下げる
exif:Contrast	Closed Choice of Integer	内部	EXIF タグ 41992（0xA408）。カメラによって適用されたコントラスト処理の方向を示します。 0 = 通常 1 = ソフト 2 = ハード

プロパティ	値の型	カテゴリー	説明
exif:Saturation	Closed Choice of Integer	内部	EXIF タグ 41993（0xA409）。カメラによって適用された彩度処理の方向を示します。 0 = 通常 1 = 低彩度 2 = 高彩度
exif:Sharpness	Closed Choice of Integer	内部	EXIF タグ 41994（0xA40A）。カメラによって適用されたシャープネス処理の方向を示します。 0 = 通常 1 = ソフト 2 = ハード
exif:DeviceSettingDescription	DeviceSettings	内部	EXIF タグ 41995（0xA40B）。特定のモデルのカメラの撮影条件に関する情報を示します。
exif:SubjectDistanceRange	Closed Choice of Integer	内部	EXIF タグ 41996（0xA40C）。被写体までの距離を示します。 0 = 不明 1 = マクロ 2 = 近景 3 = 遠景
exif:ImageUniqueID	Text	内部	EXIF タグ 42016（0xA420）。各画像に一意に割り当てられた識別子。この識別子は 128 ビット固定長であり、16 進数表記で 32 文字の ASCII 文字列として記録されます。
exif:GPSVersionID	Text	内部	GPS タグ 0（0x00）。4 つの EXIF バイトを 10 進数で表し、それぞれをピリオドで区切ったもの。現在の値は「2.0.0.0」です。
exif:GPSLatitude	GPSCoordinate	内部	GPS タグ 2（0x02）（位置）および 1（0x01）（北緯または南緯）。緯度を示します。
exif:GPSLongitude	GPSCoordinate	内部	GPS タグ 4（0x04）（位置）および 3（0x03）（東経または西経）。経度を示します。
exif:GPSAltitudeRef	Closed Choice of Integer	内部	GPS タグ 5（0x5）。標高が正の海抜か負の海抜かを示します。 0 = 正の海抜 1 = 負の海抜
exif:GPSAltitude	Rational	内部	GPS タグ 6（0x06）。海抜（メートル）を示します。

プロパティ	値の型	カテゴリー	説明
exif:GPSTimeStamp	Date	内部	GPS タグ 29（0x1D）（日付）、および GPS タグ 7（0x07）（時刻）。GPS データのタイムスタンプ（協定世界時（CUT））。 **注意：** `GPSDateStamp` タグは、EXIF 2.2 で新しく追加されたタグです。EXIF 2.1 の GPS タイムスタンプには、日付が含まれていません。この場合でも、フォーマットされた XMP 値は、ISO 8601 の「`HH:MM:SS.sss<tz>`」の形式に従う必要があります。`SS.sss` の 1 秒未満は、必要な桁数だけ指定できます。`<tz>` ではタイムゾーンを指定します。1 秒未満の部分が、分母が 10 の階乗でない分数から計算されている場合は、1 秒未満の部分を 6 桁にして、マイクロ秒の精度にすることをお勧めします。
exif:GPSSatellites	Text	内部	GPS タグ 8（0x08）。衛星情報。形式は規定されていません。
exif:GPSStatus	Closed Choice of Text	内部	GPS タグ 9（0x09）。画像作成時の GPS 受信機のステータス。 A = 測位中 V = 未測位
exif:GPSMeasureMode	Closed Choice of Integer	内部	GPS タグ 10（0x0A）。GPS 測位モード。 2 = 2 次元測位 3 = 3 次元測位
exif:GPSDOP	Rational	内部	GPS タグ 11（0x0B）。GPS データの精度。
exif:GPSSpeedRef	Closed Choice of Text	内部	GPS タグ 12（0x0C）。速度の測定に使用する単位。 K = キロメートル / 時 M = マイル / 時 N = ノット
exif:GPSSpeed	Rational	内部	GPS タグ 13（0x0D）。GPS 受信機の移動速度。
exif:GPSTrackRef	Closed Choice of Text	内部	GPS タグ 14（0x0E）。移動方向の基準 T = 真方向 M = 磁気方向

プロパティ	値の型	カテゴリー	説明
exif:GPSTrack	Rational	内部	GPS タグ 15（0x0F）。GPS の移動方向。値の範囲は 0 から 359.99 です。
exif:GPSImgDirectionRef	Closed Choice of Text	内部	GPS タグ 16（0x10）。移動方向の基準 T = 真方向 M = 磁気方向
exif:GPSImgDirection	Rational	内部	GPS タグ 17（0x11）。撮影時の画像の方向。値の範囲は 0 から 359.99 です。
exif:GPSMapDatum	Text	内部	GPS タグ 18（0x12）。測地測量データ。
exif:GPSDestLatitude	GPSCoordinate	内部	GPS タグ 20（0x14）（位置）および 19（0x13）（北緯または南緯）。目的地の緯度を示します。
exif:GPSDestLongitude	GPSCoordinate	内部	GPS タグ 22（0x16）（位置）および 21（0x15）（東経または西経）。目的地の経度を示します。
exif:GPSDestBearingRef	Closed Choice of Text	内部	GPS タグ 23（0x17）。移動方向の基準 T = 真方向 M = 磁気方向
exif:GPSDestBearing	Rational	内部	GPS タグ 24（0x18）。目的地の方角。値の範囲は 0 ～ 359.99 です。
exif:GPSDestDistanceRef	Closed Choice of Text	内部	GPS タグ 25（0x19）。速度の測定に使用する単位。 K = キロメートル M = マイル N = ノット
exif:GPSDestDistance	Rational	内部	GPS タグ 26（0x1A）。目的地までの距離。
exif:GPSProcessingMethod	Text	内部	GPS タグ 27（0x1B）。場所の検索に使用した方法を記録した文字列。
exif:GPSAreaInformation	Text	内部	GPS タグ 28（0x1C）。GPS のエリア名を記録した文字列。
exif:GPSDifferential	Closed Choice of Integer	内部	GPS タグ 30（0x1E）。GPS 受信機に誤差補正が適用されているかどうかを示します。 0 = 補正なし 1 = 補正適用

データの表現および変換

この節では、EXIF 2.2 のネイティブメタデータ形式から XMP 形式へのマッピングについて説明します。重要なデータを失わずに変換を行う方法を解説し、変換後の XMP の例を示します。

注意： EXIF ファイルで特定のタグが削除されている場合、対応する XMP プロパティも削除してください。存在しない EXIF タグのデフォルト値を使用して対応する XMP プロパティを作成することはしないでください。

EXIF から XMP への型マッピングは、ほとんどの場合、損失なく行えます。主に問題となるのは、EXIF のテキスト値です。XMP から変換する場合、EXIF で `short` または `long` としてオプション指定されている整数は、値が -32768 ～ +32767 の範囲にある場合は `short` に、それ以外の場合は `long` にする必要があります。

EXIF のテキスト

EXIF のテキスト値は、null で終わる一連の ASCII 文字です。XMP のテキスト値は、UTF-8 の Unicode 文字で、最後に null はありません。EXIF を XMP に変換すると、最後の null は削除され、残りの ASCII コードは有効な UTF-8 になります。XMP を EXIF に変換すると、ASCII 以外の文字はエスケープされます（http://www.w3.org/Addressing/URL/4_Recommentations.html で指定されている URL エスケープが使用されます）。0 ～ 127 の範囲の ASCII 文字（空白など）はエスケープされません。そして最後に null が追加されます。

XMP のテキスト値はローカライズが可能です。Lang Alt 型のプロパティには、ローカライズした一連のテキスト値を指定することができます。EXIF から XMP の Lang Alt 型プロパティへの変換では、EXIF メタデータの値をそのプロパティのデフォルトエントリ（`[@xml:lang='x-default']`）に書き出します。XMP から EXIF への変換では、このデフォルトエントリを使用して EXIF メタデータを作成します。

EXIF の日付

XMP では、日付／時刻の値はすべて ISO 8601 形式で保存されます。これは、日付と時刻を結合したもので、1 秒未満の値を持っており、タイムゾーンが指定されています。バイナリの EXIF 値では、1 秒未満の値は分離されるのが普通です。EXIF 2.1 にはタイムゾーン情報がありませんが、EXIF 2.2 では部分的に追加されています。XMP に変換する際には、1 秒未満の値も含める必要があります。EXIF にタイムゾーンが含まれていない場合は、ローカルタイムを想定して XMP に変換します。

例

EXIF 2.2 のメタデータの例と、その EXIF データを XMP に変換した場合の XMP メタデータを示します。

EXIF データ：

```
IFD 0 [1]
Make = "Canon"
Model = "Canon PowerShot S300"
Orientation = "1"
XResolution = "180/1" (180.00)
YResolution = "180/1" (180.00)
ResolutionUnit = "2"
DateTime = "2001:07:25 20:18:27"
YCbCrPositioning = "1"
ExposureTime = "1/60" (0.0167)
FNumber = "27/10" (2.70)
ExifVersion = "30 32 31 30"
DateTimeOriginal = "2001:07:25 20:18:27"
DateTimeDigitized = "2001:07:25 20:18:27"
ComponentsConfiguration = "1 2 3 0"
CompressedBitsPerPixel = "3/1" (3.00)
ShutterSpeedValue = "189/32" (5.91)
ApertureValue = "93/32" (2.91)
ExposureBiasValue = "0/3" (0.00)
MaxApertureValue = "187820/65536" (2.8659)
SubjectDistance = "913/1000" (0.9130)
MeteringMode = "5"
Flash = "0x01"
FocalLength = "173/32" (5.41)
```

XMP メタデータ：

注意： この例では、単純なプロパティを XML 要素ではなく XML 属性として表現する RDF の省略表記を使用しています。

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <rdf:Description about='' xmlns:tiff='http://ns.adobe.com/tiff/1.0'
     tiff:Make='Canon'
     tiff:Model='Canon PowerShot S300'
     tiff:Orientation='1'
     tiff:XResolution='180/1'
     tiff:YResolution='180/1'
     tiff:ResolutionUnit='2'
     tiff:DateTime='2001-07-25T20:18:27-07:00'
     tiff:YCbCrPositioning='1'>
  </rdf:Description>

  <rdf:Description about='' xmlns:exif='http://ns.adobe.com/exif/1.0'
     exif:ExposureTime='1/60'
     exif:FNumber='27/10'
     exif:ExifVersion='0210'
     exif:DateTimeOriginal='2001-07-25T20:18:27-07:00'
     exif:DateTimeDigitized='2001-07-25T20:18:27-07:00'
     exif:CompressedBitsPerPixel='3/1'
     exif:ShutterSpeedValue='189/32'
     exif:ApertureValue='93/32'
     exif:ExposureBiasValue='0/3'
     exif:MaxApertureValue='187820/65536'
     exif:SubjectDistance='913/1000'
     exif:MeteringMode='5'
     exif:Flash='1'
     exif:FocalLength='173/32'>
     <exif:ComponentsConfiguration>
        <rdf:Seq>
           <rdf:li>1</rdf:li>
           <rdf:li>2</rdf:li>
           <rdf:li>3</rdf:li>
           <rdf:li>0</rdf:li>
        </rdf:Seq>
     </exif:ComponentsConfiguration>
  </rdf:Description>
</rdf:RDF>
```

プロパティ値の型

次の表に、XMP スキーマで使用される値の型を示します。

基本的な型

Boolean

使用できる値は `True` または `False` です（この文字列は示されているとおりの綴りで入力する必要があります）。

Choice

ボキャブラリの複数の値から選択された、1 つの値です。文字列で表現されます。ボキャブラリを使用すると、限定された値のセット、または拡張可能な値のセットをプロパティに指定することができます。メタデータのスキーマでは、有効な値のセットが固定であるか、拡張可能なのかを指定できます。

choice には、オープン（open）choice とクローズ（closed）choice があります。

- オープン choice では、1 つまたは複数の候補値がリストされますが、他の値も自由に使用できます。
- クローズ choice の場合は、そこに定義されている値のみが使用できます。

プロパティに明確な値を指定する必要があり、すべてのユーザにそのプロパティの意味を正確に理解してもらう必要がある場合は、クローズ choice ボキャブラリを使用します。理解しやすい明確な値のセットが決定しているが、別の値が必要になる可能性があり、値が追加されても問題がない場合には、オープン choice を使用します。

Date

ISO 8601 形式で表現された日付。http://www.w3.org/TR/NOTE-datetime に説明があります。

Dimensions

描画オブジェクトの寸法を保持する構造体。

- このフィールドの名前空間の URI は、`http://ns.adobe.com/xap/1.0/sType/Dimensions#` です。
- このフィールドの推奨される名前空間接頭辞は、`stDim` です。

フィールド名	値の型	説明
w	Real	幅
h	Real	高さ
unit	open Choice	単位。例えば、`inch`、`mm`、`pixel`、`pica`、`point` など。

Integer

整数表現として使用される、符号付きまたは符号なしの数値の文字列。この文字列は、任意の長さの 10 進数の文字列で構成されます。最初に「+」または「-」の符号を付けることもできます。

Lang Alt

言語別の値（14 ページの「言語別の値」を参照してください）。言語プロパティ修飾子を持つ「alt Text」型の配列です。

Locale

言語を指定する クローズ choice。RFC1766 の値を使用します。http://www.ietf.org/rfc/rfc1766.txt を参照してください。

Real

任意の精度の数値。10 進数の文字列で構成されます。小数点を 1 個使用することもできます。また、最初に「+」または「-」の符号を付けることもできます。

また、オプションで Text 型の vQual:binRep 修飾子も設定できます。これは、正確な値が必要な場合に、代わりとなるバイナリ表現を提供するために使用されます。このテキストは次のように解釈されます。

std size,endian,hexadecimal_value

- *std* は標準名（「IEEE754」）です。
- *size* は、32 ビットの場合は S、64 ビットの場合は D です。
- *endian* は、リトルエンディアンの場合は L、ビッグエンディアンの場合は B です。

例えば、「IEEE754D,L,3A4901F387D31108」のように指定します。

MIMEType

ファイル形式を識別するための オープン Choice。MIME タイプは、http://www.ietf.org/rfc/rfc2046.txt にある RFC 2046『Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types』で定義されています。

ProperName

Unicode テキスト文字列で表現された、人または組織の名前。

Text

Unicode 文字列。

Thumbnail

ファイルのサムネール画像。

- このフィールドの名前空間の URI は、`http://ns.adobe.com/xap/1.0/g/img/` です。
- このフィールドの推奨される名前空間接頭辞は、`xapGImg` です。

フィールド名	値の型	説明
height	Integer	高さ（ピクセル）
width	Integer	幅（ピクセル）
format	Closed Choice	画像のエンコーディング。定義値： `JPEG`
image	Text	base 64 に変換されたサムネール画像（ピクセルデータのみ。RFC 2045 の第 6.8 節に準拠）。

URI

インターネットの Uniform Resource Identifier。抽象的または物理的なリソースを特定するための簡潔な文字列。http://www.w3.org/Addressing/ を参照してください。

URL

インターネットの Uniform Resource Locator。http://www.w3.org/Addressing/ を参照してください。`http`、`ftp`、`mailto` などのよく知られた URI 方式に関連する非公式の用語です（技術仕様では使用されなくなっています）。

XPath

XML 文書の一部分を参照するために使用する XML Path 言語（XPath）。http://www.w3.org/TR/xpath を参照してください。

Media Management の値の型

AgentName

プログラムの名前。「ベンダ アプリケーション バージョン」（「vendor app version」）という表記を推奨します。例えば、「Adobe Acrobat Distiller 5.0」のようにします。

RenditionClass

レンディションのタイプ。規定されている標準名のボキャブラリ（オープン Choice）の中から選択します。一連のトークンおよびパラメータをコロンで区切って指定します。最初のトークンは、レンディションの基本的な概念を表します。オプションで追加のトークンを指定して、レンディションの具体的な特性を指定することもできます。定義値は、次のとおりです。

`default`	マスター文書。トークンの追加はできません。
`thumbnail`	特定のバージョンを簡略化または縮小したプレビュー。トークンを追加して特性を指定できます。`thumbnail`：形式：サイズ：カラースペースの順に指定することを推奨します。例えば、`thumbnail:jpeg`、`thumbnail:16x16`、`thumbnail:gif:8x8:bw` のように指定します。
`screen`	画面解像度または Web 用のレンディション。
`proof`	レビュー用校正出力。
`draft`	レビュー用のレンディション。
`low-res`	低解像度でフルサイズの代替物。

ResourceEvent

文書の処理で発生した高レベルのイベント。

- このフィールドの名前空間の URI は、`http://ns.adobe.com/xap/1.0/sType/ResourceEvent#` です。
- このフィールドの推奨される名前空間接頭辞は、`stEvt` です。

フィールド名	値の型	説明
action	open Choice	発生したアクション。定義値は、`converted`、`copied`、`created`、`cropped`、`edited`、`filtered`、`formatted`、`version_updated`、`printed`、`published`、`managed`、`produced`、`resized` です。新しい値を追加する場合は、英語の動詞の過去形にしてください。
instanceID	URI	変更されたリソースのインスタンス ID。
parameters	Text	アクションの追加説明。
softwareAgent	AgentName	アクションを実行したソフトウェアエージェント。
when	Date	アクションが実行されたときのタイムスタンプ（省略可能）。

ResourceRef

複数の部分から成る、リソースへの参照。前のバージョン、レンディションの元の文書、派生文書の元の文書などを示すのに使用します。使用方法および参照リソースが管理されているかどうかによって、参照で使用されるフィールドは異なります。instanceID 以外のすべてのフィールドは、参照しているリソースの xmpMM スキーマのプロパティです。

- このフィールドの名前空間の URI は、`http://ns.adobe.com/xap/1.0/sType/ResourceRef#` です。
- このフィールドの推奨される名前空間接頭辞は、`stRef` です。

フィールド名	値の型	説明
instanceID	URI	参照しているリソースのインスタンス ID。
documentID	URI	参照しているリソースの xmpMM:DocumentID。
versionID	Text	参照しているリソースの xmpMM: VersionID。
renditionClass	RenditionClass	参照しているリソースの xmpMM: RenditionClass。
renditionParams	Text	参照しているリソースの xmpMM: RenditionParams。
manager	AgentName	参照しているリソースの xmpMM:Manager。
managerVariant	Text	参照しているリソースの xmpMM: ManagerVariant。
manageTo	URI	参照しているリソースの xmpMM:ManageTo。
manageUI	URI	参照しているリソースの xmpMM:ManageUI。

Version

文書の 1 つのバージョンに関する説明です。

- このフィールドの名前空間の URI は、`http://ns.adobe.com/xap/1.0/sType/Version#` です。
- このフィールドの推奨される名前空間接頭辞は、`stVer` です。

フィールド名	値の型	説明
comments	Text	変更内容に関するコメント。
event	ResourceEvent	ユーザが実行した操作に関する、高レベルで正式な説明。
modifyDate	Date	このバージョンがチェックインされた日付。
modifier	ProperName	このバージョンを変更した人。
version	Text	新しいバージョン番号。

Basic Job ／ Workflow の値の型

Basic Job ／ Workflow スキーマでは、次の値の型が使用されています。

Job

ジョブ管理システムのジョブに関する説明です。

- このフィールドの名前空間の URI は、`http://ns.adobe.com/xap/1.0/sType/Job#` です。
- このフィールドの名前空間接頭辞は、`stJob` です。

フィールド名	値の型	説明
name	Text	ジョブの非公式な名前。この名前はユーザに表示するためのもので、非公式なシステムで使用されます。
id	Text	ジョブの一意の ID。このフィールドは、外部のジョブ管理システムへの参照です。
url	URL	外部のジョブ管理ファイルを参照する URL。

EXIF スキーマの値の型

次の型は、EXIF スキーマでのみ使用します。

Rational

EXIF の有理数の値を XMP で表現するには、テキストに変換する必要があります。分子 + スラッシュ（「/」）+ 分母の形の Text 型の値を使用することをお勧めします。例えば、2/3 という値は、XMP に変換すると "`2/3`" というテキスト値になります。

GPSCoordinate

「`DD,MM,SSk`」または「`DD,MM.mmk`」の形のテキスト値です。ここで、

- `DD` は度を表す数値です。
- `MM` は分を表す数値です。
- `SS` は秒を表す数値です。
- `mm` は 1 分未満の分を表す値です。
- `k` は、`N`、`S`、`E`、`W` のいずれか 1 文字で、方位（北、南、東、西）を表します。

Flash

フラッシュの状態を表す構造体。

フィールド	値の型	説明
Fired	Boolean	フラッシュが発光した場合は `True`。
Return	Closed Choice	ストロボのリターン検出機能がサポートされているかどうか。サポートされている場合は、検出されたかどうか。次のいずれかの値になります。 0 = ストロボのリターン検出機能なし 2 = ストロボのリターン光未検出 3 = ストロボのリターン光検出
Mode	Closed Choice	フラッシュのモード。次のいずれかの値になります。 0 = 不明 1 = 強制フラッシュ発光 2 = 強制フラッシュ抑制 3 = 自動モード
Function	Boolean	フラッシュ機能がない場合は True。
RedEyeMode	Boolean	赤目軽減機能がサポートされている場合は `True`。

OECF/SFR

OECF/SFR を表す構造体。

フィールド	値の型	説明
Columns	Integer	列数、n
Rows	Integer	行数、m
Names	seq Text	列項目の名前、n エントリ。

Values	seq Rational	OECF/SFR の値。次の順序で指定します。
		`value [0,0]`
		…
		`value [n - 1, 0]`
		`value [0, m - 1]`
		…
		`value [n - 1, m - 1]`

CFAPattern

CFA パターンを示す構造体。

フィールド	値の型	説明
Columns	Integer	列数、n
Rows	Integer	行数、m

Values	seq Integer	CFA の値。次の順序で指定します。
		`value [0,0]`
		…
		`value [n - 1, 0]`
		`value [0, m - 1]` …
		`value [n - 1, m - 1]`

DeviceSettings

デバイス設定を示す構造体。

フィールド	値の型	説明
Columns	Integer	列を表示します。
Rows	Integer	行を表示します。
Settings	seq Text	カメラ設定を規定の順に指定します。

スキーマの拡張

この節では、新規スキーマの作成方法、および既存のスキーマの拡張方法を説明します。

カスタムスキーマの作成

このマニュアルで定義されているスキーマは、多種多様なニーズに適用できると考えられているコアスキーマです。既存のスキーマのプロパティが使用できるのであれば、できる限りそうすべきです。しかし、XMP は、カスタムスキーマを追加することで簡単に拡張できるようになっていますので、使用したいメタデータが既存のコアスキーマに用意されていない場合は、独自のスキーマを定義して使用することができます。

新しい名前空間の作成を検討する際は、次の指針に従ってください。

- 既存の名前空間に用意されているプロパティと同じ意味を持つプロパティを含めないようにしてください。
- 公開するのが有益であると思われるプロパティについては、協力しあって共通の名前空間を作成し、互換性のない名前空間が乱立しないようにしてください。

新しいスキーマを定義したら、それを使用する人に読んでもらうための仕様書を作成する必要があります。そして、そのメタデータの処理コードを書くことになるすべての開発者のために、その仕様書を入手できるようにする必要があります。

注意： XMP の将来のバージョンでは、プログラムで読み取る形式のスキーマ仕様書がサポートされる可能性もありますが、これはあくまでも人間が読む形式の仕様書を補助するためのものです。

仕様書には次の項目を含める必要があります。

- 一意のスキーマ名。URI と、推奨する接頭辞を記述します。
- 各プロパティの名前、値の型、プロパティの説明を示した表。

構造化された型を使用したプロパティを定義する場合は、構造化されたプロパティ値の構成要素を識別するための URI 名を追加することもできます。

その後、必要に応じてさらにプロパティを追加できます。その際は、このマニュアルで説明している RDF および XMP の構文要件に従って、互換性のある RDF メタデータを作成してください。

スキーマの拡張

スキーマを拡張する際は、次の点に留意してください。

- 新しいプロパティを既存の名前空間に追加できます。この場合、アプリケーションの動作に影響を与えることはありません。
- 既存の名前空間のプロパティの定義は変更しないでください。変更すると、アプリケーションが正しく動作しなくなる場合があります。プロパティの意味を変更する必要がある場合は、新しいプロパティを作成してください（古いプロパティは非推奨にしてください）。
- スキーマの名前空間の「新しいバージョン」を作成することもできます。ただし、古いバージョンと新しいバージョンの間に論理的な関係はありません。2 つの名前空間にそれぞれ同じ名前のプロパティがあっても、これらは異なるプロパティになります。

5 アプリケーションファイルへの XMP メタデータの埋め込み

この章では、XMP パケット内の XMP メタデータが、さまざまな形式のファイルにどのように埋め込まれているかについて説明します。最も円滑に文書を処理できるアプリケーションは、XMP がどのように埋め込まれているかを認識しているアプリケーションです。

以下の各節では、対象となっているファイル形式について実用的な知識があることを前提として説明を行っています。

- TIFF
- JPEG
- JPEG 2000
- GIF
- PNG
- HTML
- PDF
- AI（Adobe Illustrator）
- SVG/XML
- PSD （Adobe Photoshop）
- PostScript および EPS

TIFF

TIFF ファイルでは、Image File Directory（IFD）内のエントリが XMP パケットをポイントします。そのエントリは、700（10 進数）というタグ値を持っています。

バイトオフセット	フィールド値	フィールド名	コメント
0、1	700	`TAG`	フィールドを識別するためのタグ（10 進数）。
2、3	1	`Field type`	フィールドの型は `BYTE` で、1 という値で表現されています。
4 ～ 7		`Count`	XMP パケットの合計バイト数。
8 ～ 11		`Value` または `Offset`	XMP パケットのバイトオフセット。

関連資料

TIFF 6.0 仕様：

http://partners.adobe.com/asn/developer/pdfs/tn/TIFF6.pdf

JPEG

JPEG ファイルでは、APP1 マーカが XMP パケットの位置を指します。次の表に、エントリのフォーマットを示します。

バイトオフセット	フィールド値	フィールド名	長さ（バイト）	コメント
0	0xFFE1	APP1	2	APP1 マーカ
2	2 + 名前空間の長さ（29）+ XMP パケットの長さ	Lp	2	この数に次の 2 つの部分を加えたサイズ（バイト）。
4	null で終わり引用符が付かない ASCII 文字列	namespace	29	一意の ID として使用される、XMP の名前空間の URI ：http://ns.adobe.com/xap/1.0/
33	< XMP パケット >			

ヘッダとその後のデータを合わせて、64 KB 未満である必要があります。XMP パケットは複数のマーカで分割することはできないので、XMP パケットのサイズは最大でも 65502 バイトにする必要があります。

関連資料

- 『JPEG File Interchange Format Version 1.02』
- ISO/IEC 10918-1『Information technology - Digital Compression and Coding of continuous-tone still images: requirements and guidelines』
- ISO/IEC 10918-4『Information technology - Digital compression and coding of continuous-tone still images: Registration of JPEG profiles, SPIFF profiles, SPIFF tags, SPIFF color spaces, APP*n* markers, SPIFF compression types and Registration Authorities (REGAUT)』

  この資料では、APP*n* マーカのフォーマットおよびファイル交換フォーマットが規定されています。

JPEG 2000

JP2 形式は、一連の「ボックス」で構成されています。次の表に示すように、XMP パケットは UUID ボックスに保持されています。

フィールド値	フィールド名	長さ（バイト）	コメント
全体の長さ（バイト。このフィールドの 4 バイトも含む）	Length	4	ビッグエンディアンの符号なし整数。
0x75756964 ('uuid')	Type	4	ビッグエンディアンの符号なし整数。
BE 7A CF CB 97 A9 42 E8 9C 71 99 94 91 E3 AF AC	UUID	16	ISO/IEC 11578:1996 で定義されている 16 バイトのバイナリ UUID。
< XMP パケット >	DATA		

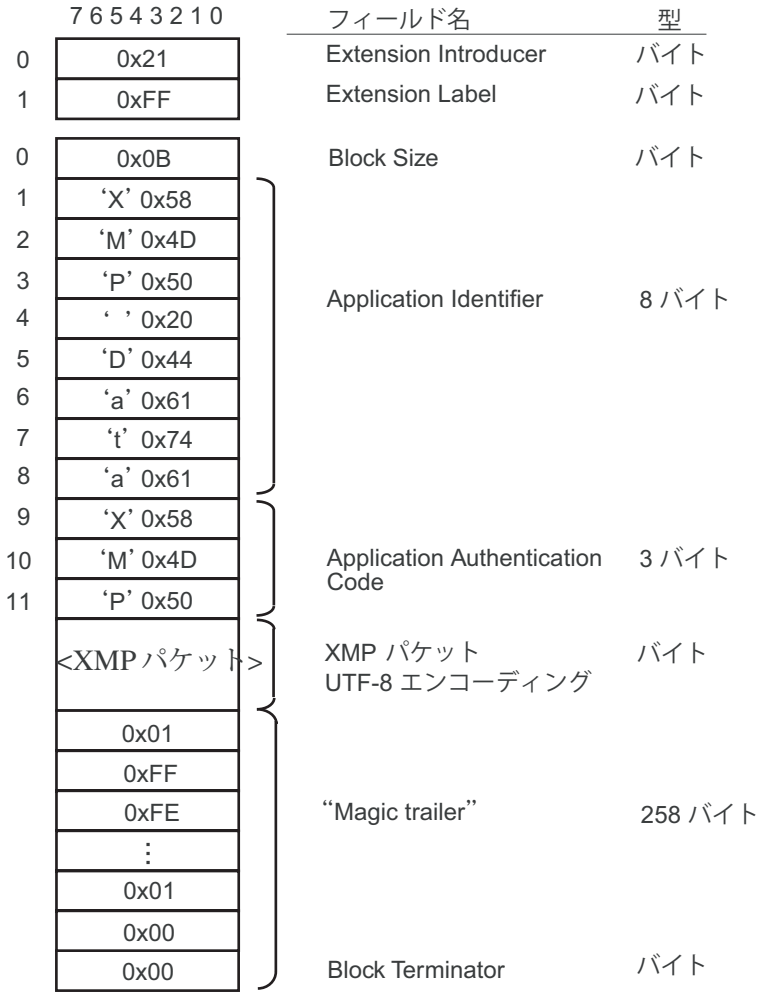
関連資料

JPEG 2000 標準に関する情報は、http://www.jpeg.org/JPEG2000.html で入手できます。

GIF

GIF89a ファイルでは、Application Extension に XMP パケットが保存されます（次の図を参照してください）。その Application Identifier は「XMP Data」で、Application Authenticator は「XMP」です。Application Data は、XMP パケット（UTF-8 エンコードである必要があります）と、それに続く 258 バイトの「マジック」トレーラ（値は 0x01、0xFF、0xFE、0xFD ....0x03、0x02、0x01、0x00、0x00）で構成されています。最後のバイトは、Block Terminator です。

GIF ファイル形式での XMP：



XMP が UTF-8 でなければならないのには理由があります。GIF の Application Data は、実際には GIF データの一連のサブブロックとして扱われます。各サブブロックの最初のバイトは、サブブロックの内容の長さを表します（ただし最初のバイト自身は含まれません）。

Application Data が読み取られるとき、まず長さを表すバイトが読み取られます。これがゼロでなかった場合は、そのバイト数だけデータが読み取られ、再び長さを表すバイトが読み取られます。この処理は、長さを表すバイトの値がゼロになるまで繰り返されます。

XMP が UTF-8 でエンコードされている場合、XMP パケット内に値がゼロのバイトは存在しないことになります。そのため、XMP に対応していないソフトウェアは、パケットデータのバイトの値をサブブロックの長さと解釈しながらパケットを読み進んでいき、最終的にマジックトレーラのどこかに到達します。このトレーラのどのバイトが読み込まれても、最後の Block Terminator にジャンプするようになっています。

関連資料

GIF 89a の仕様は、http://members.aol.com/royalef/gif89a.txt で入手できます。

PNG

PNG グラフィックファイルでは、iTXt 型のチャンクの追加によって XMP パケットが埋め込まれます。このチャンクは、意味的には tEXt のチャンクや zTXt のチャンクと同じですが、テキストデータのエンコードが Latin-1 ではなく、Unicode 文字セットの UTF-8 になっています。

Chunk Data の部分が XMP パケットです。このパケットには、読み取り専用のマークを付ける必要があります。

注意： チャンクデータの後には CRC チェックサムがあるので、このファイル形式に対応していない XMP ソフトウェアが XMP パケットの内容を変更しないようにする必要があります。

XMP を含むチャンクを 1 つの PNG ファイルに複数含めないでください。エンコードする際には、ファイルの最初にチャンクを配置することをお勧めしますが、これは必須ではありません。

PNG のデータフォーマット

フィールド	長さ	コメント
Length	4	チャンクのデータフィールドのバイト数を表す符号なし整数（チャンクタイプコードまたは CRC は含みません）。
Chunk Type	4	「iTXt」
Chunk Data：標準の iTXt チャンクヘッダと XMP パケット		
Keyword	17	「XML:com.adobe.xmp」
Null separator	1	値 = 0x00
Compression flag	1	値 = 0x00、非圧縮データを指定。
Compression method	1	値 = 0x00
Language tag	0	XMP メタデータでは使用しません。
Null separator	1	値 = 0x00
Translated keyword	0	XMP メタデータでは使用しません。
Null separator	1	値 = 0x00
Text	パケット長	<XMP パケット >
CRC	4	巡回冗長検査（Cyclic Redundancy Check）。チャンク内の転送前のバイトに基づいて計算されます。Chunk Type コードと Chunk Data のフィールドは含まれますが、Length フィールドは含まれません。

関連資料

http://www.w3.org/TR/REC-png.html

HTML

HTML に埋め込む XMP は、HTML への XML の埋め込みに関する W3C 勧告のいずれかに準拠している必要があります。関連資料の項を参照してください。

XML は、SCRIPT または XML 要素として埋め込むことができます。これは任意の有効な場所に配置できますが、HEAD 要素の終わりに配置することをお勧めします。SCRIPT または XML 要素の内容が XMP パケットになります。

ブラウザで RDF プロパティの値を表すテキストがページのコンテンツとして表示されないためには、ブラウザが SCRIPT または XML 要素を認識できる必要があります。古いソフトウェアとの間に互換性がない場合以外は、XML 要素を使用することをお勧めします。互換性がない場合は、SCRIPT 要素の方が認識される可能性が高くなります。

HTML に XML を埋め込む方法は、次の例に示すように 3 種類あります。最初の 2 つの方法では SCRIPT 要素を使用しており、3 番目の方法では XML 要素を使用しています。

SCRIPT 要素と LANGUAGE 属性を使用

```
<html>
 <head>
  <SCRIPT LANGUAGE="XML">
  <?xpacket begin='' id='W5M0MpCehiHzreSzNTczkc9d'?>
  <!-- シリアル化した RDF をここに挿入します。簡潔にするために省略してあります。-->
  <?xpacket end='w'?>
  </SCRIPT>
 </head>
 <body>
 </body>
</html>
```

注意： Microsoft Windows XP 上の Microsoft Word 2000 で SCRIPT 要素と LANGUAGE 属性を使用すると、本文（body 部分）が表示されないという問題が発生します。

SCRIPT 要素と TYPE 属性を使用

```
<html>
 <head>
  <SCRIPT TYPE="text/xml">
  <?xpacket begin='' id='W5M0MpCehiHzreSzNTczkc9d'?>
  <!-- シリアル化した RDF をここに挿入します。簡潔にするために省略してあります。-->
  <?xpacket end='w'?>
  </SCRIPT>
 </head>
 <body>
 </body>
</html>
```

XML 要素を使用

```
<html>
 <head>
  <XML>
  <?xpacket begin='' id='W5M0MpCehiHzreSzNTczkc9d'?>
  <!-- シリアル化した RDF をここに挿入します。簡潔にするために省略してあります。 -->
  <?xpacket end='w'?>
  </XML>
 </head>
 <body>
 </body>
</html>
```

関連資料

1998 年 5 月の W3C ミーティングレポート：http://www.w3.org/TR/NOTE-xh

PDF

PDF ファイルでは、PDF オブジェクトの中のメタデータストリームに XMP パケットが埋め込まれます（PDF 1.4 よりサポート）。

次の例は、XMP メタデータが XMP パケットとして埋め込まれ、メタデータストリームとして保存されているところを示しています。

```
1152 0 obj
<< /Type /Metadata /Subtype /XML /Length 1706 >>
stream
<?xpacket begin='' id='W5M0MpCehiHzreSzNTczkc9d'?>
<!-- シリアル化した RDF をここに挿入します。簡潔にするために省略してあります。-->
<?xpacket end='w'?>
endstream
endobj
```

増分保存が行われた PDF ファイルには、パケットが複数含まれていて、どれも「メイン」の XMP メタデータのように見える場合があります。増分保存では、古いデータは削除されず、ファイルの最後に新しいデータ（XMP パケットを含む）が書き込まれます。トップレベルの PDF 辞書も書き直されるので、PDF を扱うアプリケーションは、辞書を調べることで新しいパケットのみを見つけることができます。

関連資料

PDF ファイル内のメタデータストリームについて詳しくは、http://partners.adobe.com/asn/tech/pdf/specifications.jsp にある 『PDF Reference, Version 1.5』を参照してください。

AI（Adobe Illustrator）

Adobe Illustrator® で生成される .ai ファイルは、Portable Document Format（PDF）と互換性があります。したがって、XMP メタデータを埋め込むフォーマットは、PDF ファイルのフォーマットと同じです。

SVG/XML

XMP メタデータは有効な XML なので、XML 文書には直接埋め込むことができます。XMP パケットは完全に独立した XML 文書として使用されることを意図したものではないので、XMP パケットに `XML` 宣言は含まれていません。要素または処理命令が有効になる場所であれば、XML 文書内のどの場所にも XMP パケットを配置できます。

ファイルは、UTF-8 または UTF-16 を使用して、Unicode でエンコードすることをお勧めします。そうすれば、XMP パケットをスキャンしてその内容のみを解析するソフトウェアにも対応できるようになります。

関連資料

XML の仕様は、http://www.w3.org/TR/REC-xml で入手できます。

PSD （Adobe Photoshop）

Adobe Photoshop® の .psd ファイルには、画像リソースブロックが含まれています。これは、画像に関連する非ピクセルデータを保持しておくために使用されます。次の表に、画像リソースブロックのフォーマットを示します。

フィールド	型	説明
Type	OSType	Photoshop では、必ず 8BIM という署名が使用されます。
ID	2 バイト	ID = 1060（XMP メタデータの場合）。
Name	PString	Pascal 文字列。パディングによってサイズは常に偶数に保たれます（つまり、ゼロのバイトが必要に応じて最後に追加されます）。null は 2 バイトの 0 で表現されます。 Photoshop 7 の場合、XMP メタデータの Name 値は「XMP」になります。
Size	4 バイト	リソースデータの実際のサイズ。これには、Type、ID、Name、Size の各フィールドは含まれません。
Data	Variable	リソースデータ。パディングによってサイズは常に偶数に保たれます。これが XMP パケットです。

上の表の Name フィールドと Data フィールドにある「パディングによってサイズは常に偶数に保たれます」とは、元のフィールド値に必要に応じてゼロのバイトが追加されるという意味です。

PostScript および EPS

XMP メタデータを PostScript® ファイルや EPS ファイルに埋め込んで、PostScript や PDF のワークフローで活用することができます。この節では、PostScript や EPS に XMP を埋め込む方法を、文書レベルのメタデータ（メイン XMP）の場合と、画像などの内部オブジェクトに対するメタデータ（オブジェクト XMP）の場合に分けて説明します。また、Distiller を使用して PostScript や EPS から PDF を生成する場合が多いので、Acrobat Distiller に関連する問題についても説明しています。

注意： これは、Distiller を使用する必要があるという意味ではありません。また、他のアプリケーションには問題がないという意味でもありません。

PostScript ファイルには主に 3 つの種類があり、XMP の書き出し、検索および使用方法に影響する場合があります。その種類は、次のとおりです。

- DSC PostScript（または単に「PostScript」）：『PostScript Language Reference』の「Appendix G」に定義されている DSC 表記に準拠した PostScript。
- 生の PostScript：特定の構造規則に従っていない PostScript。生の PostScript は使用しないことをお勧めします。84 ページの「コンテンツの順序」で説明するように、メイン XMP を迅速かつ確実に見つけられるようにするには、特別な DSC コメントが必要です。
- EPS：『PostScript Language Reference』の「Appendix H」に定義されている EPS 表記に準拠した PostScript。EPS は、DSC PostScript のサブセットです。

使用上の問題を避けるため、文書レベルの XMP は PostScript と EPS では別の書き方をする必要があります。オブジェクトレベルの XMP は、PostScript と EPS で同じ書き方をします。

文書レベルのメタデータ

PostScript や EPS に含まれている XMP を最も確実に見つける方法は、他のファイル形式の場合と同じく、そのファイル形式に完全に対応することです。つまり、PostScript の場合は PostScript インタプリタを実行することになります。パケットのスキャンによる方法は、ファイルに複数の XMP パケットがある場合や、メイン XMP がなくオブジェクト XMP だけがある場合には、信頼性が低くなります。

多くの場合、オブジェクト XMP を（一時的であれ）無視して、メイン XMP を探すのが良い方法です。メイン XMP を見つけるために PostScript ファイル全体を解釈しようとすると、非常に効率が悪くなることがあります。XMP パケットのスキャンと PostScript コメントのスキャンによって確実にメイン XMP を見つけられるようにするには、ヒントを埋め込み、適切な順序で XMP を配置しておく必要があります。

PostScript に文書レベルのメタデータを書く場合、アプリケーションは次のことを行う必要があります。

- `%ADO_ContainsXMP` コメントを書き込みます。84 ページの「コンテンツの順序」を参照してください。
- XMP パケットを書き込みます。85 ページの「PostScript の文書レベルの XMP」を参照してください。

EPS に文書レベルのメタデータを書く場合、アプリケーションは次のことを行う必要があります。

- %ADO_ContainsXMP コメントを書き込みます。84 ページの「コンテンツの順序」を参照してください。
- XMP パケットを書き込みます。86 ページの「EPS の文書レベルの XMP」を参照してください。

生の PostScript は %ADO_ContainsXMP コメントがないので、使用しないことをお勧めします。生の PostScript を使用する必要がある場合は、85 ページの「PostScript の文書レベルの XMP」の説明に従って、XMP を埋め込む必要があります。

コンテンツの順序

大規模な出版物の多くは、PostScript を広範囲に使用しています。数百または数千の EPS ファイルが配置された、非常に大規模なレイアウトも珍しくありません。PostScript はテキストなので、PostScript に埋め込まれている XMP を見つけるには、PostScript プログラム全体を解析するか、少なくともそのテキストをすべてスキャンする必要があります。配置されている PostScript ファイルは、非常に大きい場合があります。これらは複合文書を表すものであったり、複数の XMP パケットを含んでいたりする場合もあります。PostScript ファイルに XMP が含まれているかどうかを判断するだけでも、ファイル全体を検索する必要があります。

こうしたことは、PostScript に埋め込まれている XMP を処理するレイアウトプログラムにとって、パフォーマンスに影響を与える問題となります。プログラムでこの問題を部分的に解決する方法として、PostScript のヘッダコメントに特別なマーカコメントを配置して、メイン XMP の場所に関するヒントを提供することができます。このマーカは、%%EndComments 行よりも前に配置する必要があります。

このマーカでは、PostScript を読み取るアプリケーションに対して、メイン XMP が存在するかどうか、およびどのようにメイン XMP を探したらよいかという情報を提供するようにします。XMP マーカの形式は次のとおりです。

```
%ADO_ContainsXMP: <オプション> ...
```

このオプションは、スペース、タブ、LF、または CR を含まない一連の文字列です。大文字と小文字は区別されます。コロンの前に空白文字は入れないでください。アプリケーションでは、サポートしていないオプションは無視するようにしてください。

現在定義されているオプションは 3 つあります。これらのオプションは、メイン XMP を探すためのヒントとなるもので、互いに排他的です。メイン XMP は文書レベルの XMP とは限らないことに注意してください。

- MainFirst：ファイルの最初の XMP パケットがメイン XMP で、ファイルの最初の方にあります。XMP は、PostScript のコンテンツのできるだけ最初の方に配置しておく必要があります。
- MainLast：ファイルの最後の XMP パケットがメイン XMP で、ファイルの最後の方にあります。XMP は、PostScript のコンテンツのできるだけ最後の方に配置しておく必要があります。
- NoMain：この PostScript ファイルに、メイン XMP パケットはありません。ただし、このファイルに XMP パケットが含まれていないとは限りません。例えば、埋め込まれた EPS セクションの中にあったり、内部オブジェクトに添付されていることもあります。

注意： XMP の位置オプションは、ファイル内でのメイン XMP の位置を表すとともに、他のオブジェクトレベルの XMP に対する相対位置も表しています。MainFirst を使用する場

合、メイン XMP パケットは他のすべての XMP よりも前に配置する必要があります。MainLast を使用する場合は、他のすべての XMP よりも後ろに配置する必要があります。他の XMP パケットを、メインパケットに隣接して配置する必要はありません。

注意： EPS ファイルを連結する場合は、連結後のファイルに対して、一連の PostScript ヘッダコメントを新たに作成する必要があります。また、必要に応じてメイン XMP パケットを新たに作成します。そうしないと、最初の EPS に含まれている XMP マーカコメントが、連結後のファイルに関する情報として誤って解釈される危険があります。

PostScript の文書レベルの XMP

この節では、PostScript デバイスがレベル 2 またはそれ以降であり、Distiller のバージョン 6.0 またはそれ以降が使用されることを前提としています。互換性の問題については、88 ページの「PostScript の Distiller 5 との互換性」および 88 ページの「PostScript および EPS の LanguageLevel 1」を参照してください。

文書レベルの XMP を設定する手順は、大きく 3 つの段階からなります。

1. XMP を保持するための PostScript ストリームオブジェクトの作成。
2. ストリームオブジェクトへの XMP の配置。
3. XMP ストリームオブジェクトと文書の関連付け。

XMP メタデータは、ファイルをスキャンしてメタデータを探すソフトウェアから認識できるような形で、PostScript ファイルに埋め込む必要があります。つまり、完全な XMP パケットとして埋め込む必要があります。しかし、そのファイルが PostScript 出力デバイスに送られると、そのパケットデータが原因となって PostScript エラーが発生し、ジョブが失敗します。任意のデータを扱えるようにするには、対象のファイルから XMP データを読み取って、解析が不要な場合はそのデータを廃棄するプロシージャが必要になります。

注意： この後の例では、次のようなプライベート辞書のプロシージャをいくつか定義しています。

```
privatedict /metafile_pdfmark {flushfile cleartomark} bind put
```

この privatedict は、説明の都合上このような名前にしているに過ぎません。実際の製品のコードでは、これらのプロシージャを一意の辞書に定義して、1 つの文書内で様々な EPS ファイルを使用できるようにし、これらのプロシージャとわずかに異なるバージョンが共存できるようにしてください。

PostScript に文書レベルの XMP を埋め込む例を、次に示します。この例には、必要なマーカコメントは含まれていません。

```
% ==========================================================================
% まず Postscript のプロローグを書きます。ここでは、XMP メタデータを処理する際に
% 使用する演算子やプロシージャを定義します。

% pdfmark から cleartomark までを定義して、PostScript プリンタや Distiller
% の 4.0 以前によって読み込まれた場合にはデータが廃棄されるようにします。この後の
% 「privatedict」へのすべての参照は、競合が起こらないように、一意の名前に変更する必要が
% あります。詳しくは、87 ページの「名前の競合の回避」の節で説明します。

/currentdistillerparams where
{pop currentdistillerparams /CoreDistVersion get 5000 lt} {true} ifelse

{privatedict /pdfmark /cleartomark load put
```

```
   privatedict   /metafile_pdfmark   {flushfile   cleartomark}   bind   put}
{privatedict /metafile_pdfmark {/PUT pdfmark} bind put} ifelse

% ========================================================================
% 次に、XMP メタデータを保持するストリームを作成します。これは、上のプロローグの
% 後に記述する必要がありますが、直後に記述する必要はありません。

% データを保持するための、名前の付いた pdfmark ストリームオブジェクトを作成します。
% 前述の privatedict と同様に、my_metadata_stream_123 という名前をそのまま
% 使用せず、一意の名前を使用することをお勧めします。このストリームの名前は、
% この Postscript プログラム内でのみ有効です。外部に対しては有効ではありません。
%
% 最初にストリームオブジェクトを定義し、次にそのストリームに XMP パケットを読み込み、
% 最後にメイン XMP としてそのストリームを追加します。
%
% 「&&end XMP packet marker&&」というコメントは重要です。このコメントによって
% XMP パケットの読み込みが終了します。

% 手順 1： XMP メタデータストリームオブジェクトを作成し、XMP として宣言します。
[/_objdef   {my_metadata_stream_123}   /type   /stream   /OBJ   pdfmark
[{my_metadata_stream_123} 2 dict begin
   /Type /Metadata def /Subtype /XML def currentdict end /PUT pdfmark

% 手順 2： ストリームに XMP パケットを読み込みます。
[{my_metadata_stream_123}
   currentfile 0 (% &&end XMP packet marker&&)
   /SubFileDecode filter metafile_pdfmark

... ここに XMP パケットを記述します ...

% &&end XMP packet marker&&

% 手順 3： メイン XMP のメタデータストリームとしてストリームを追加します。
[{Catalog} {my_metadata_stream_123} /Metadata pdfmark
```

EPS の文書レベルの XMP

EPS への XMP の埋め込みは PostScript の場合と非常によく似ていますが、EPS は他の EPS や PostScript に埋め込まれることがよくあるので、それに起因する問題が存在します。EPS の文書レベルの XMP とは、実際には EPS の最も外側の XMP のことを意味しています。これは、EPS が単独で PDF に変換された場合には、PDF の文書レベルの XMP になりますが、他の EPS や PostScript に埋め込まれている状態で変換された場合には、適切なマーク付きコンテンツになります。

EPS の処理では次のことが必要になります。

- XMP は、すべての EPS コンテンツ（PostScript 描画コマンド）の前に配置する必要があります。
- EPS のコンテンツは、/BDC および /EMC **pdfmark** で囲む必要があります。

- XMP セットアップの手順 3 では、異なる PostScript コードを使用します。

前述の例に変更を加えた例（一部省略）を次に示します。

```
%%EndPageSetup
[/NamespacePush pdfmark

... 前述の XMP セットアップの手順 3 までの処理をすべて行います ...

% 手順 3： Marked Content 辞書にストリームを添付します。
% 描画コマンドは、すべて /BDC 演算子と /EMC 演算子の間に配置する必要があります。
[/Document 1 dict begin
   /Metadata {my_metadata_stream_123} def currentdict end /BDC pdfmark
[/NamespacePop pdfmark

... すべての描画コマンドをここに配置します ...

%%PageTrailer
[/EMC pdfmark
```

名前の競合の回避

これらのサンプルでは、*{my_metadata_stream_123}* という名前を使用していますが、この名前には一意のものを使用するようにしてください。たとえば、一般的な UUID を生成し、そこから主要な英数字以外を取り除き、それを名前の後に結合するという方法をお勧めします。

または、NamespacePush および NamespacePop **pdfmark** を使用することもできます。この方法は、『Pdfmark Reference Manual』（Distiller のヘルプメニューからアクセス可能）でも推奨されています。可能であればこちらの方が望ましい方法ですが、プッシュとポップが大きく分離して現実的でない場合もあります。

名前付きオブジェクトを使用しているすべての pdfmark を、NamespacePush と NamespacePop で囲まれた 1 つのブロック内に配置することが重要です。例えば、次の PostScript コードは悪い例です。

```
[/NamespacePush pdfmark
[/_objdef {my_metadata_stream_123} /type /stream /OBJ pdfmark
[{my_metadata_stream_123} 2 dict begin
   /Type /Metadata def /Subtype /XML def currentdict end /PUT pdfmark
[{my_metadata_stream_123}
   currentfile 0 (% &&end XML Packet marker&&)
/SubFileDecode filter metafile_pdfmark
... ここに XML パケットを記述します ...
% &&end XML Packet marker&&
[/NamespacePop pdfmark
% この時点で、{my_metadata_stream_123} という名前は使用できなくなっています。
% 次の行で「undefined」エラーが発生します。
[{Catalog} {my_metadata_stream_123} /Metadata pdfmark
```

PostScript の Distiller 5 との互換性

XMP は Acrobat Distiller バージョン 5 で初めてサポートされましたが、/Metadata **pdfmark** はサポートされていません。Distiller 5 を使用する場合、文書レベルの XMP を簡単に追加する方法はありません。/Metadata **pdfmark** は無視され、PostScript エラーも発生しません。

PostScript および EPS の LanguageLevel 1

`SubFileDecode` フィルタが使用可能なのは、PostScript LanguageLevel 2 からです。XMP を含む PostScript または EPS を PostScript LanguageLevel 1 デバイス（古いプリンタなど）で処理する必要がある場合は、別の方法を使用してストリームオブジェクトに XMP を読み込む必要があります。

PostScript LanguageLevel 1 では、少なくとも 2 つの方法が考えられます。1 つは、`readstring` を使用して XMP パケット全体を読み込む方法です。もう 1 つは、`readline` を使用して、エンドマーカが見つかるまで XMP パケットデータを 1 行ずつ読み込む方法です。

ここでは、`readline` を使用する方法を説明します。`readline` を使用すれば、`readstring` を使用した場合に発生する 2 つの問題を解決することができます。

- パケット全体の正確なサイズを知っておく必要はなく、1 行の最大長を知っていれば十分です。
- テキストエディタを使用して、PostScript ／ EPS ファイルを異なる改行文字（CR、LF、または CRLF）で再保存すると、XMP パケットの正確な長さが変わる可能性があります。

PostScript に対して `readline` を使用する方法の例を次に示します。全体的には前の例とよく似ていますが、手順 2 とそれに関連するプロローグが異なっています。

```
% ==========================================================================
% まず Postscript のプロローグを書きます。ここでは、XMP メタデータを処理する際に
% 使用する演算子やプロシージャ手順を定義します。

% pdfmark から cleartomark までを定義して、PostScript プリンタや Distiller
% の 4.0 以前によって読み込まれた場合にはデータが廃棄されるようにします。この後の
% 「privatedict」へのすべての参照は、競合が起こらないように、一意の名前に変更する必要が
% あります。詳しくは、「名前の競合の回避」の節で説明しています。


/currentdistillerparams where
{pop currentdistillerparams /CoreDistVersion get 5000 lt} {true} ifelse
{privatedict /pdfmark /cleartomark load put} if

% マーカ行が見つかるまで現在のファイルから 1 行ずつ読み込む別のプロシージャを定義
% します。行の最大長は、XMP の行を読み込む一時バッファの作成に使用
% されます。
% スタック上： [ {name} maxLineLength MarkerString

privatedict /metastring_pdfmark
{ 2 dict begin
/markerString exch def string /tmpString exch def
{ currentfile tmpString readline pop
markerString anchorsearch
{pop pop cleartomark exit}
```

```
{3 copy /PUT pdfmark pop 2 copy (¥n) /PUT pdfmark} ifelse
} loop
end
}bind put

% ========================================================================
% 次に、XMP メタデータを保持するストリームを作成します。これは、上のプロローグの
% 後に記述する必要がありますが、直後に記述する必要はありません。

% データを保持するための、名前の付いた pdfmark ストリームオブジェクトを PDF に作成します。
% 前述の privatedict と同様に、my_metadata_stream_123 という名前をそのまま使用せず、
% 一意の名前を使用することをお勧めします。このストリームの名前は、この Postscript
% プログラム内でのみ有効です。外部に対しては有効ではありません。
%
% 最初にストリームオブジェクトを定義し、次にそのストリームに XMP パケットを読み込み、
% 最後にメイン XMP としてそのストリームを追加します。

% 下の <LineLength> は、XMP パケットの最も長い行の長さよりも大きな値で置き換える
% 必要があります。この値を正確に決定するための安全で一般的な方法はありません。
% XMP は、Postscript が書かれた後にインプレースで変更される可能性があります。
% また、すべてを 1 行で記述することも可能です。
%
% しかし、パケット全体の長さは変更できません。<LineLength> は、パケットサイズまたは
% 65500 のどちらか小さいほうの値に設定します。上限値を設けているのは、PostScript 文字
% 列の上限である 64KB を超えないようにするためです。
%
% 「&&end XMP packet marker&&」というコメントは重要です。このコメントによって
% XMP パケットの読み込みが終了します。

% 手順 1： XMP メタデータストリームオブジェクトを作成し、XMP として宣言します。
[/_objdef {my_metadata_stream_123} /type /stream /OBJ pdfmark
[{my_metadata_stream_123} 2 dict begin
   /Type /Metadata def /Subtype /XML def currentdict end /PUT pdfmark

% 手順 2： ストリームに XMP パケットを読み込みます。
[{my_metadata_stream_123} <LineLength>
   (% &&end XMP Packet marker&&) metastring_pdfmark

... ここに XMP パケットを記述します ...

% &&end XMP Packet marker&&

% 手順 3： メイン XMP のメタデータストリームとしてストリームを追加します。
[{Catalog} {my_metadata_stream_123} /Metadata pdfmark
```

従来の PDF メタデータと XMP

この章では、主に、PostScript および EPS に明示的に XMP を埋め込み、メタデータを追加する方法について説明を行ってきました。しかし、Distiller を使用すれば、XMP を PostScript に埋め込まなくても、別のソースによる情報を PDF ファイルの文書レベルのメタデータに含めることができます。このメタデータは、従来は PDF 文書の情報辞書に格納されていましたが、XMP の登場によって、PDF の文書レベルの XMP に複製されるようになりました。

PostScript ファイルにメタデータを埋め込み、Distiller によってそのメタデータを PDF 文書情報辞書に追加し、さらに XMP パケットを作成して PDF 文書に埋め込むには、他に 2 つの方法があります。次の方法を使用できます。

- DSC（Document Structuring Conventions）コメント。DSC コメントは、DSC 解析を有効にしている場合にのみ処理されます。つまり、ジョブファイルに次の行が含まれている場合にのみ処理されます。

  ```
  /ParseDSCCommentsForDocInfo true
  ```

- `DOCINFO` **pdfmark** コマンド。**pdfmark** について詳しくは、Distiller アプリケーションのヘルプメニューにある「pdfmark ガイド」を参照してください。

PDF に埋め込むために、上記 3 つのメタデータのソースのうち 2 つ以上が PostScript で使用されている場合、その最初のソースが文書レベルの XMP のプロパティ値として使用され、PDF が作成されます。この PDF には、次のプロパティが含まれています。

- 明示的な文書レベルの XMP。
- 明示的な文書情報辞書。
- DSC コメント。

pdfmark コマンドは DSC コメントより信頼性が高いので、多くのアプリケーションは PDF 文書の DocInfo プロパティの設定に pdfmark コマンドを使用しています。FrameMaker によって作成された PostScript コードを次に示します。これは、`DOCINFO` **pdfmark** 演算子の使用例を示しています。

```
/Creator (FrameMaker 6.0)
/CreationDate (D:20020214144924)
/ModDate (D:20020215142701)
/Author(John Doe)
/Title (Processing XMP Data in EPS Files)
/Subject (XMP)
/Keywords (XMP, pdfmark)
/DOCINFO pdfmark
```

Distiller は、これら 7 つのプロパティ（と「Producer」）を、2 つの場所に配置して PDF を生成します。1 つは文書情報辞書であり、もう 1 つは文書レベルのメタデータ（XML パケット）です。Producer は、「Acrobat Distiller 5.0（Windows）」などの製品名です。その他のキーと値のペアを PDF DocInfo に追加することもできますが、Distiller 5.0 の場合、文書レベルのメタデータにそれらは追加されません。

このファイルが、Distiller ではなく PostScript インタプリタに送られる可能性がある場合は、注意が必要です。古いプリンタなどで使用されている一部の PostScript インタプリタでは、**pdfmark** コマンドが認識されない場合があります。問題を回避する方法としては、条件付きで **pdfmark** 演算子を「`cleartomark`」演算子として定義する方法があります。これについては、前述の例で説明しています。

オブジェクトレベルのメタデータ

オブジェクトレベルの XMP は、PostScript と EPS で同じ書き方をします。

メタデータストリームは、PostScript ファイル内の特定のオブジェクトに、**pdfmark** 演算子を使用して添付できます。この方法は、文書レベルの PostScript の方法とほぼ同じです（85 ページの「PostScript の文書レベルの XMP」を参照してください）。ただし、手順 3 で、XMP メタデータを含むストリームをオブジェクトに添付するところが異なります。次の例では、画像に対してこの方法を使用しています。

```
% ===========================================================================
% XMP ストリームは、前の例と同じように定義されているものとします。手順 3 以外では
% すべて、ストリームをメタデータストリームとして定義します。画像も {myImage_123} と
% 定義されているものとします。ここでもやはり、一意の名前を使用してください。
%
% 手順 3 は、XMP メタデータを画像と関連付ける処理で置き換えられています。これは、
% 画像と XMP ストリームよりも後に記述する必要があるので、他の XMP の部分
% と隣接していないこともあります。「コンテンツの順序」に記載されている、順
% 序に関する問題を参照してください。

% 手順 3： XMP メタデータストリームを画像に添付します。
[{myImage_123} <</Metadata {my_metadata_stream_123}>> /PUT pdfmark
```

注意： ここに示した方法は、すべての PostScript デバイスと互換性があります。つまり、追加の変更を加えなくても、レベル 1 デバイスでは認識不能な XMP（前述）は正しく無視され、Distiller の 5 以降では XMP が PDF ファイル内の関連するオブジェクトに添付されます。

注意： Distiller 5 は PDF ファイル内の関連するオブジェクトに XMP を添付しますが、PDF 内の XMP ストリームは Flate 圧縮されます。したがって、PDF 内のオブジェクト XMP パケットは、外部のパケットスキャナからは見えません。この XMP を見るには、PDF 形式を認識してストリームの圧縮を解除できるソフトウェアが必要です。Distiller 6 以降では、XMP パケットストリームは圧縮されません。