SAS® Viya™ 3.1 XML LIBNAME Engine: ユーザーガイド

SAS® ドキュメント

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2016. *SAS® Viya™ 3.1 XML LIBNAME Engine: ユーザーガイド*. Cary, NC: SAS Institute Inc.

SAS® Viya™ 3.1 XML LIBNAME Engine: ユーザーガイド

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

September 2016

3.1-P1:engxml

目次

1 部

使用

1 章
入門ガイド: XML Engine

XML LIBNAME Engine の機能

XML LIBNAME Engine は *XML ドキュメント*を処理します。このエンジンは次のことを行います。

- SAS 独自のファイル形式を XML マークアップに変換することにより、DATA タイプの SAS データセットから XML ドキュメントをエクスポートします(出力場所に書き出します)。出力された XML ドキュメントは次のように扱われます。
  - XML ドキュメントを処理する製品により使用されます。
  - 別のホストに移動し、XML Engine が XML マークアップを SAS データセットに変換して処理します。
- 外部 XML ドキュメントをインポートします(入力場所から読み込みます)。入力された XML ドキュメントは、SAS データセットに変換されます。

XML LIBNAME Engine の機能について

ライブラリ参照名の割り当て

XML LIBNAME Engine は、他の SAS エンジンと同様に動作します。すなわち、ライブラリ参照名を割り当ててエンジンを指定するには、LIBNAME ステートメントを実行します。いったん割り当てたライブラリ参照名は、その SAS セッション全体を通じて使用できます。

XML LIBNAME Engine は特定の XML ドキュメントに関連付けることができます。または、ディレクトリベースの環境における SAS ライブラリの物理的な場所に関連付けることもできます。このようなライブラリ参照名を使用すると、SAS システムにより、SAS データセット内のデータが XML マークアップに変換されるか、または XML マークアップが SAS 形式のデータに変換されます。

XML ドキュメントのインポート

XML ドキュメントを SAS データセットとしてインポートするには、次の LIBNAME ステートメントで、ライブラリ参照名を該当する XML ドキュメントに割り当て、XML Engine を指定します。

```
libname myxml xmlv2 '/u/sasxxx/XML/Students.xml';
```

DATASETS プロシジャを実行すると、XML ドキュメントが Students という SAS データセットに変換されることが示されます。

```
proc datasets library=myxml;
```

アウトプット 1.1　MYXML ライブラリの DATASETS プロシジャ出力

Directory	
Libref	MYXML
Engine	XMLV2
Physical Name	/u/ /XML/Students.xml
XMLType	SAS XML Generic
Version	.

#	Name	Member Type
1	STUDENTS	DATA

PRINT プロシジャを実行すると、次の出力が表示されます。

```
proc print data=myxml.students;
run;
```

アウトプット 1.2 MYXML.STUDENTS の PRINT プロシジャ出力

Obs	ID	NAME	ADDRESS	CITY	STATE
1	755	Brad Martin	1611 Glengreen	Huntsville	Texas
2	1522	Zac Harvell	11900 Glenda	Houston	Texas

XML ドキュメントのエクスポート

XML ドキュメントを SAS データセットからエクスポートするには、XML Engine の LIBNAME ステートメントで、作成する XML ドキュメントにライブラリ参照名を割り当てます。

次のプログラムの最初の LIBNAME ステートメントは、ライブラリ参照名 MYFILES を、SAS データセット Singers を含んでいる SAS ライブラリに割り当てます。2 番目の LIBNAME ステートメントは、ライブラリ参照名 MYXML を、データセット Myfiles.Singers からエクスポートされる XML ドキュメントの物理的な場所に割り当てます。

```
libname myfiles '/u/MyFiles/';

libname myxml xmlv2 '/u/MyFiles/XML/Singers.xml';
```

次のステートメントを実行すると、Singers.XML という名前の XML ドキュメントが作成されます。

```
data myxml.Singers;
  set myfiles.Singers;
run;
```

アウトプット 1.3 XML ドキュメント Singers.XML の内容

```
<?xml version="1.0" encoding="windows-1252" ?> <TABLE> <SINGERS> <FirstName>
Tom </FirstName> <Age> 62 </Age> </SINGERS> <SINGERS> <FirstName> Willie </
FirstName> <Age> 70 </Age> </SINGERS> <SINGERS> <FirstName> Randy </FirstName>
<Age> 43 </Age> </SINGERS> </TABLE>
```

XML Engine でサポートされる SAS 処理

XML Engine は次のような処理をサポートします。

- XML Engine は、入力(読み込み)処理と出力(作成)処理をサポートします。XML Engine は更新処理をサポートしません。
- XML Engine は順次アクセスエンジンであり、データを 1 つずつ順番に処理します。このエンジンは、ファイルの先頭で処理を開始し、ファイルの終わりまで処理を続行します。XML Engine は、一部の SAS アプリケーションと機能で必要となるランダム(ダイレクト)アクセスは提供しません。たとえば、XML Engine を使用する場合、SORT プロシジャや、SQL プロシジャでの ORDER BY ステートメントは使用できません。ランダムアクセスを必要とする処理を要求した場合、そのような処理は順次アクセスでは無効であることを知らせるメッセージが SAS ログに表示されます。このメッセージが表示

された場合、続行する前に、XML データを一時 SAS データセット内に配置します。

環境間での XML ドキュメントの移送

XML ドキュメントを(FTP などを使用して)環境間で転送する場合、ドキュメントの内容に基づいて適切な転送モードを決定する必要があります。ドキュメントに XML 宣言によるエンコーディング属性が含まれている場合、または XML 宣言の前にバイトオーダーマークが記述されている場合、そのファイルをバイナリモードで転送します。ドキュメントがどちらの条件も満たしていない場合、類似したホスト間でドキュメントを転送するには、そのファイルをテキストモードで転送します。

XML Engine を使用して XML ドキュメントをエクスポートした場合、デフォルトでは、その XML ドキュメントには、SAS データセットのエンコーディングに基づいて生成された XML 宣言によるエンコーディング属性が含まれています (例:**<?xml version="1.0" encoding="windows-1252" ?>**)。XML ドキュメントをエクスポートする際に LIBNAME ステートメントで XMLENCODING=オプションを指定すると、SAS データセットのエンコーディングをオーバーライドできます。

FAQ

DOM または SAX アプリケーションの XML Engine

XML Engine は「Document Object Model (DOM)」ではなく「Simple API for XML (SAX)」モデルを使用します。SAX は、ドキュメントの内容に関するランダムアクセス検索を提供しません。SAX はドキュメントを順次スキャンし、各項目を 1 つずつアプリケーションに提供します。

XML Engine の XML ドキュメントの検証について

XML Engine は入力 XML ドキュメントを検証しません。XML Engine は、渡されたデータが有効な XML マークアップ形式で記述されていると仮定します。XML Engine は DTD (Document Type Definition)や SCHEMA を使用しないため、検証の基準となるものが存在しません。

XML Engine と ODS MARKUP 出力先の使用時の違いについて

XML Engine は、XML ドキュメントの作成と読み込みが可能です。ODS MARKUP は XML ドキュメントの作成は行いますが、読み込みは行いません。通常、データを転送する場合には XML Engine を使用し、SAS 出力から XML を作成する場合には ODS MARKUP 出力先を使用します。

SAS 以外で作成した XML ドキュメントのインポート時に発生するエラーの原因

XML Engine が読み込むファイルは、「XMLTYPE=LIBNAME」ステートメントオプションでサポートされているマークアップタイプに従っているものに限りま

す。サポートされているマークアップタイプが必要とする仕様に従っていないフリーフォームの XML ドキュメントをインポートしようとすると、エラーが発生します。XMLTYPE=マークアップタイプに従っていないファイルを正常にインポートするには、XMLMap と呼ばれる独立した XML ドキュメントを作成する必要があります。XMLMap の構文は、XML マークアップを SAS データセット、変数(列)、オブザベーション(行)に変換する方法を XML Engine に伝えます。5 章, “XMLMap を使用し、XML ドキュメントをインポートする” (29 ページ)を参照してください。

XML LIBNAME Engine のユーザー補助機能

XML LIBNAME Engine はコマンドベースの製品です。今回のリリースでは、ユーザー補助機能は追加されていませんが、XML LIBNAME Engine はグラフィカルユーザーインターフェイスを使用しておらず、文字をタイプしてコマンドを発行できるユーザーなら誰でも同機能を使用できるため、同製品はユーザー補助の標準に準拠しています。SAS 製品のユーザー補助機能に関するご質問は、accessibility@sas.com または SAS テクニカルサポートまでお問い合わせください。

2 章
XML ドキュメントのエクスポート

XML ドキュメントのエクスポートの方法について

XML ドキュメントのエクスポートとは、DATA タイプの SAS データセットを、出力先の XML ドキュメントに書き出す処理のことです。XML Engine は、SAS に固有のフォーマットを XML マークアップに変換することにより、XML ドキュメントをエクスポートします。

XML ドキュメントをエクスポートするには、XML Engine 用の LIBNAME ステートメントを使用して、XML ドキュメントが作成される物理的な場所にライブラリ参照名を割り当てます。その後、このライブラリ参照名を使用して出力を生成する SAS プログラム(DATA ステップや COPY プロシジャなど)を実行します。

SAS 日付値、SAS 時刻値、SAS 日時値を含む XML ドキュメントのエクスポート

この例では、日時値、日付値、時刻値を含む SAS データから、XML ドキュメントをエクスポートします。この XML ドキュメントは、GENERIC マークアップタイプのドキュメントとして生成されます。

まず、次の SAS プログラムにより、単純な SAS データセットを作成し、そのデータセットの内容を出力します。変数 DateTime には日時値、変数 Date には日付値、変数 Time には時刻値がそれぞれ含まれます。

```
data test;
  DateTime=14686;
  format DateTime datetime.;
  Date=14686;
  format Date date9.;
  Time=14686;
```

```
  format Time timeampm.;

proc print data=test;
run;
```

アウトプット 2.1　SAS 日付値、時刻値、日時値を含むデータセット WORK.TEST の PRINT プロシジャ出力

The SAS System

Obs	DateTime	Date	Time
1	01JAN60:04:04:46	17MAR2000	4:04:46 AM

次のプログラムは、SAS 日付値、時刻値、日時値を含む GENERIC マークアップタイプの XML ドキュメントをエクスポートします。

```
libname trans xmlv2 'XML-document' xmltype=generic; 1

data trans.test; 2
  set work.test;
run;
```

1　LIBNAME ステートメントは、エンジンとして XML Engine を指定した上で、ライブラリ参照名 TRANS を、エクスポートされた XML ドキュメントが格納されるファイルの物理的な場所(完全なパス名、ファイル名、ファイル拡張子を含むもの)に割り当てます。XMLTYPE=オプションではデフォルト値の GENERIC を指定しています。

2　続く DATA ステップでは、SAS データセット WORK.TEST を読み込み、同データセットの内容を指定の XML ドキュメントに出力します。

結果として生成される XML ドキュメントの内容は次のようになります。

アウトプット 2.2　GENERIC マークアップを使用した XML ドキュメントの内容

```
<?xml version="1.0" encoding="windows-1252" ?> <TABLE> <TEST> <DateTime>
1960-01-01T04:04:46.000000 </DateTime> <Date> 2000-03-17 </Date> <Time>
04:04:46 </Time> </TEST> </TABLE>
```

数値のエクスポート

次の例では、高精度の値を含んでいる数値変数を持つ小さな SAS データセットを使用します。次の SAS プログラムは、割り当て済みのユーザー定義の出力形式を使ってデータセットを作成し、そのデータセットから XML ドキュメントをエクスポートします。その後、PRINT プロシジャで出力の違いを示します。

```
libname format xmlv2 '/u/mydocuments/format.xml'; 1
```

```
libname prec xmlv2 '/u/mydocuments/precision.xml' xmldouble=internal; 2

data npi; 3
  do n=1 to 10;
    n_pi = n*3.141592653589793;
    output;
  end;
format n_pi f14.2;
run;

data format.dbltest; 4
  set npi;
run;

data prec.rawtest; 5
  set npi;
run;

title 'Drops the Precision'; 6
proc print data=format.dbltest;
  format n_pi f14.10;
run;

title 'Keeps the Precision'; 7
proc print data=prec.rawtest;
  format n_pi f14.10;
run;
```

1 最初の LIBNAME ステートメントは、ライブラリ参照名 FORMAT を、生成される XML ドキュメント FORMAT.XML が格納されるファイルに割り当てます。エンジンのデフォルトの動作は、割り当てられた SAS フォーマットによる数値制御になります。

2 2 番目の LIBNAME ステートメントは、ライブラリ参照名 PREC を、生成される XML ドキュメント PRECISION.XML が格納されるファイルに割り当てます。XMLDOUBLE=オプションに INTERNAL が指定されているため、エンジンは格納されている生の値を取り出します。

3 続く DATA ステップでは、一時データセット NPI を作成します。このデータセットには、高精度の値を含む数値変数が含まれています。この変数には、小数点以下 2 桁までを有効とするユーザー定義の出力形式が割り当てられます。

4 次の DATA ステップでは、WORK.NPI からデータセット FORMAT.DBLTEST を作成します。

5 次の DATA ステップでは、WORK.NPI からデータセット PREC.RAWTEST を作成します。

6 続く PRINT プロシジャでは、データセット FORMAT.DBLTEST から XML ドキュメント FORMAT.XML を作成します。FORMAT.XML には、SAS 出力形式により制御された数値が含められます。アウトプット 2.3 (12 ページ)を参照してください。

7 この PRINT プロシジャには、精度の損失を示す出力形式が指定されています。この出力では、小数点以下 2 桁よりも小さい桁はすべてゼロになります。アウトプット 2.4 (12 ページ)を参照してください。

8 続く PRINT プロシジャでは、データセット PREC.RAWTEST から XML ドキュメント PRECISION.XML を作成します。PRECISION.XML には、データセット

PREC.RAWTEST に格納されていた数値が含められます。アウトプット 2.5 (13 ページ)を参照してください。

9 この PRINT プロシジャ出力では、精度が保持されていることを示す出力形式が指定されています。アウトプット 2.6 (13 ページ)を参照してください。

アウトプット 2.3 XML ドキュメント FORMAT.XML の内容

```
<?xml version="1.0" encoding="iso-8859-1" ?> <TABLE> <DBLTEST> <n>1</n>
<n_pi>3.14</n_pi> </DBLTEST> <DBLTEST> <n>2</n> <n_pi>6.28</n_pi> </DBLTEST>
<DBLTEST> <n>3</n> <n_pi>9.42</n_pi> </DBLTEST> <DBLTEST> <n>4</n> <n_pi>12.57</
n_pi> </DBLTEST> <DBLTEST> <n>5</n> <n_pi>15.71</n_pi> </DBLTEST> <DBLTEST>
<n>6</n> <n_pi>18.85</n_pi> </DBLTEST> <DBLTEST> <n>7</n> <n_pi>21.99</n_pi> </
DBLTEST> <DBLTEST> <n>8</n> <n_pi>25.13</n_pi> </DBLTEST> <DBLTEST> <n>9</n>
<n_pi>28.27</n_pi> </DBLTEST> <DBLTEST> <n>10</n> <n_pi>31.42</n_pi> </DBLTEST>
</TABLE>
```

アウトプット 2.4 FORMAT.DBLTEST の PRINT プロシジャ出力

Drops the Precision

Obs	N_PI	N
1	3.1400000000	1
2	6.2800000000	2
3	9.4200000000	3
4	12.5700000000	4
5	15.7100000000	5
6	18.8500000000	6
7	21.9900000000	7
8	25.1300000000	8
9	28.2700000000	9
10	31.4200000000	10

アウトプット 2.5 XML ドキュメント PRECISION.XML の内容

```
<?xml version="1.0" encoding="iso-8859-1" ?> <TABLE> <RAWTEST> <n
rawvalue="QRAAAAAAAAA=">1</n> <n_pi rawvalue="QTJD9qiIWjA=">3.14</n_pi> </
RAWTEST> <RAWTEST> <n rawvalue="QSAAAAAAAAA=">2</n> <n_pi
rawvalue="QWSH7VEQtGA=">6.28</n_pi> </RAWTEST> <RAWTEST> <n
rawvalue="QTAAAAAAAAA=">3</n> <n_pi rawvalue="QZbL4/mZDpA=">9.42</n_pi> </
RAWTEST> <RAWTEST> <n rawvalue="QUAAAAAAAAA=">4</n> <n_pi
rawvalue="QckP2qIhaMA=">12.57</n_pi> </RAWTEST> <RAWTEST> <n
rawvalue="QVAAAAAAAAA=">5</n> <n_pi rawvalue="QftT0UqpwvA=">15.71</n_pi> </
RAWTEST> <RAWTEST> <n rawvalue="QWAAAAAAAAA=">6</n> <n_pi
rawvalue="QhLZfH8zIdI=">18.85</n_pi> </RAWTEST> <RAWTEST> <n
rawvalue="QXAAAAAAAAA=">7</n> <n_pi rawvalue="QhX9u+m7p3U=">21.99</n_pi> </
RAWTEST> <RAWTEST> <n rawvalue="QYAAAAAAAAA=">8</n> <n_pi rawvalue="Qhkh
+1RELRg=">25.13</n_pi> </RAWTEST> <RAWTEST> <n rawvalue="QZAAAAAAAAA=">9</n>
<n_pi rawvalue="QhxGOr7Msrs=">28.27</n_pi> </RAWTEST> <RAWTEST> <n
rawvalue="QaAAAAAAAAA=">10</n> <n_pi rawvalue="Qh9qeilVOF4=">31.42</n_pi> </
RAWTEST> </TABLE>
```

アウトプット 2.6 PREC.RAWTEST の PRINT プロシジャ出力

Keeps the Precision

Obs	N_PI	N
1	3.1415926536	1
2	6.2831853072	2
3	9.4247779608	3
4	12.5663706144	4
5	15.7079632679	5
6	18.8495559215	6
7	21.9911485751	7
8	25.1327412287	8
9	28.2743338823	9
10	31.4159265359	10

3 章
XML ドキュメントのインポート

XML ドキュメントのインポートの方法について

XML ドキュメントのインポートとは、外部 XML ドキュメントを SAS データセットとして読み込む処理のことです。XML Engine は、入力された XML ドキュメントを SAS システムに独自のファイル形式に変換します。

XML ドキュメントをインポートするには、XML Engine 用の LIBNAME ステートメントを使用して、既存の XML ドキュメントが配置されている物理的な場所にライブラリ参照名を割り当てます。その後、同ライブラリ参照名を使用する SAS プログラムを実行することにより、当該 XML ドキュメントに SAS データセットとしてアクセスできます。

GENERIC マークアップタイプを使用した XML ドキュメントのインポート

この例では、GENERIC マークアップタイプの物理構造に従っている、次の XML ドキュメントをインポートします。必要とされる物理構造に関する詳細は、“GENERIC マークアップタイプを使用した XML ドキュメントのインポートに必要な物理構造の条件について” (30 ページ)を参照してください。

```
<?xml version="1.0" encoding="windows-1252" ?>
<TABLE>
  <CLASS>
    <Name> Alfred </Name>
    <Gender> M </Gender>
    <Age> 14 </Age>
```

```
    <Height> 69 </Height>
    <Weight> 112.5 </Weight>
  </CLASS>
  <CLASS>
    <Name> Alice </Name>
    <Gender> F </Gender>
    <Age> 13 </Age>
    <Height> 56.5 </Height>
    <Weight> 84 </Weight>
  </CLASS>
.
.
.
  <CLASS>
    <Name> William </Name>
    <Gender> M </Gender>
    <Age> 15 </Age>
    <Height> 66.5 </Height>
    <Weight> 112 </Weight>
  </CLASS>
</TABLE>
```

次の SAS プログラムは、XML マークアップを SAS システムに独自の形式に変換します。

```
libname trans xmlv2 'XML-document'; 1

libname myfiles 'SAS-library'; 2

data myfiles.class; 3
  set trans.class;
run;
```

1 最初の LIBNAME ステートメントは、エンジンとして XML Engine を指定した上で、ライブラリ参照名 TRANS を、XML ドキュメントの物理的な場所(完全なパス名、ファイル名、ファイル拡張子を含むもの)に割り当てます。デフォルトでは、XML Engine は GENERIC マークアップの使用を仮定します。

2 2 番目の LIBNAME ステートメントは、ライブラリ参照名 MYFILES を、結果として生成される SAS データセットが格納される SAS ライブラリの物理的な場所に割り当てます。この場合、デフォルトのエンジンとして V9 Engine が使用されます。

3 続く DATA ステップでは、XML ドキュメントを読み込み、その内容を SAS システム独自の形式で出力します。

変換後の XML ドキュメントの内容を含むデータセット出力を生成するには、次のような PRINT プロシジャを実行します。

```
proc print data=myfiles.class;
run;
```

アウトプット 3.1 MYFILES.CLASS の PRINT プロシジャ出力

The SAS System

Obs	Name	Gender	Age	Height	Weight
1	Alfred	M	14	69.0	112.5
2	Alice	F	13	56.5	84.0
3	Barbara	F	13	65.3	98.0
4	Carol	F	14	62.8	102.5
5	Henry	M	14	63.5	102.5
6	James	M	12	57.3	83.0
7	Jane	F	12	59.8	84.5
8	Janet	F	15	62.5	112.5
9	Jeffrey	M	13	62.5	84.0
10	John	M	12	59.0	99.5
11	Joyce	F	11	51.3	50.5
12	Judy	F	14	64.3	90.0
13	Louise	F	12	56.3	77.0
14	Mary	F	15	66.5	112.0
15	Philip	M	16	72.0	150.0
16	Robert	M	12	64.8	128.0
17	Ronald	M	15	67.0	133.0
18	Thomas	M	11	57.5	85.0
19	William	M	15	66.5	112.0

数値を含む XML ドキュメントのインポート

この例では、“数値のエクスポート” (10 ページ)でエクスポートした、Precision.XML という XML ドキュメントをインポートします。この例では、数値を含んでいる XML ドキュメントをインポートする場合に、デフォルトの動作を変更する方法を示します。

最初の SAS プログラムでは、デフォルトの動作を使用して XML ドキュメントをインポートします。デフォルトの動作では、パーシングされた文字データ (PCDATA)を要素から取り出します。

```
libname default xmlv2 '/u/mydocuments/precision.xml';
```

```
title 'Default Method';
proc print data=default.rawtest;
  format n_pi f14.10;
run;
```

このインポートの結果、DEFAULT.RAWTEST という名前の SAS データセットが作成されます。

アウトプット 3.2 DEFAULT.RAWTEST の PRINT プロシジャ出力

Default Method

Obs	N_PI	N
1	3.1400000000	1
2	6.2800000000	2
3	9.4200000000	3
4	12.5700000000	4
5	15.7100000000	5
6	18.8500000000	6
7	21.9900000000	7
8	25.1300000000	8
9	28.2700000000	9
10	31.4200000000	10

2 番目の SAS プログラムでは、XMLDOUBLE=を使用して XML ドキュメントをインポートすることにより、デフォルトの動作を変更しています。このプログラムは、要素内の rawdata=属性から値を取り出します。

```
libname new xmlv2 '/u/mydocuments/precision.xml' xmldouble=internal;

title 'Precision Method';
proc print data=new.rawtest;
  format n_pi f14.10;
run;
```

このインポートの結果、NEW.RAWTEST という名前の SAS データセットが作成されます。

アウトプット 3.3 NEW.RAWTEST の PRINT プロシジャ出力

Precision Method

Obs	N_PI	N
1	3.1415926536	1
2	6.2831853072	2
3	9.4247779608	3
4	12.5663706144	4
5	15.7079632679	5
6	18.8495559215	6
7	21.9911485751	7
8	25.1327412287	8
9	28.2743338823	9
10	31.4159265359	10

非エスケープ文字データを含む XML ドキュメントのインポート

W3C 規格 (第 4.6 節: 定義済みエンティティ) では、文字データの場合、左山かっこ(<)、アンパサンド(&)、アポストロフィー(')などの特定の文字は、**<**、**&**、**'**のような文字参照または文字列を使用してエスケープする必要があると規定されています。たとえば、属性値に一重引用符や二重引用符を含める場合、一重引用符(')は**'**として、二重引用符(")は**"**として表されます。

非エスケープ文字を含んでいる XML ドキュメントをインポートするには、LIBNAME ステートメントでオプション XMLPROCESS=PERMIT を指定して、W3C 規格に従っていない文字データを XML Engine が受け付けるようにします。このようにすると、アポストロフィー、二重引用符、アンパサンドのような非エスケープ文字が文字データとして受け付けられます。

注: XMLPROCESS=PERMIT を使用する場合には注意が必要です。XML ドキュメントが非エスケープ文字を含んでいる場合、そのドキュメントの内容は標準的な XML 構成ではありません。このオプションは、利便性のために提供されているものであり、無効な XML マークアップを奨励するものではありません。

次の例では、非エスケープ文字データを含んでいる Permit.XML という XML ドキュメントをインポートしています。

```
<?xml version="1.0" ?>
<PERMIT>
  <CHARS>
```

```
    <accept>OK</accept>
    <status>proper escape sequence</status>
    <ampersand>&amp;</ampersand>
    <squote>&apos;</squote>
    <dquote>&quot;</dquote>
    <less>&lt;</less>
    <greater>&gt;</greater>
  </CHARS>
  <CHARS>
     <accept>OK</accept>
     <status>unescaped character in CDATA</status>
     <ampersand>Abbott & Costello</ampersand>
     <squote>Logan's Run</squote>
     <dquote>This is "realworld" stuff</dquote>
     <less> e < pi </less>
     <greater> pen > sword </greater>
  </CHARS>
  <CHARS>
    <accept>NO</accept>
    <status>single unescaped character</status>
    <ampersand>&</ampersand>
    <squote>'</squote>
    <dquote>"</dquote>
    <less></less>
    <greater></greater>
  </CHARS>
  <CHARS>
    <accept>NO</accept>
    <status>unescaped character in string</status>
    <ampersand>Dunn & Bradstreet</ampersand>
    <squote>Isn't this silly?</squote>
    <dquote>Quoth the raven, "Nevermore!"</dquote>
    <less></less>
    <greater></greater>
  </CHARS>
</PERMIT>
```

まず、デフォルトの XML Engine の動作を使用する例を示します。この場合、XML Engine は、XML マークアップが W3C 規格に従っていると仮定します。次の SAS プログラムを実行すると、先頭の 2 つのオブザベーション(有効な XML マークアップを含んでいるもの)だけがインポートされ、残りの 2 つのレコード(非エスケープ文字を含んでいるもの)に関してはエラーが生成されます。

```
libname permit xmlv2 '/u/mydocuments/XML/permit.xml';

proc print data=permit.chars;
run;
```

ログ 3.1 SAS ログの出力

```
ERROR: There is an illegal character in the entity name. encountered during XMLInput parsing occurred
at or near line 24, column 22 NOTE: There were 2 observations read from the data set PERMIT.CHARS.
```

LIBNAME ステートメントのオプション XMLPROCESS=PERMIT を指定すると、XML Engine がこの XML ドキュメントをインポートできるようになります。

```
libname permit xmlv2 '/u/mydocuments/XML/permit.xml' xmlprocess=permit;
```

```
proc print data=permit.chars;
run;
```

アウトプット 3.4 PERMIT.CHARS の PRINT プロシジャ出力

The SAS System

Obs	accept	status	ampersand	squote	dquote	less	greater
1	OK	proper escape sequence	&	'	"	<	>
2	OK	unescaped character in CDATA	Abbott & Costello	Logan's Run	This is "realworld" stuff	e < pi	pen > sword
3	NO	single unescaped character	&	'	"		
4	NO	unescaped character in string	Dunn & Bradstreet	Isn't this silly?	Quoth the raven, "Nevermore!"		

連結された XML ドキュメントのインポート

複数の XML ドキュメントを連結したファイルも、XML Engine でインポート可能です。これを行うには、LIBNAME ステートメントで XMLCONCATENATE=YES オプションを指定します。

注: XMLCONCATENATE=YES オプションを使用する場合には注意が必要です。XML ドキュメントが連結された複数の XML ドキュメントから構成されている場合、そのドキュメントの内容は標準的な XML 構成ではありません。このオプションは、利便性のために提供されているものであり、無効な XML マークアップを奨励するものではありません。

次の例では、2 つの XML ドキュメントから構成されるファイル ConcatStudents.XML をインポートします。

```
<?xml version="1.0" ?>
<LIBRARY>
  <STUDENTS>
    <ID>1345</ID>
    <NAME>Linda Kay</NAME>
    <SCHOOL>Bellaire</SCHOOL>
    <CITY>Houston</CITY>
  </STUDENTS>
  <STUDENTS>
    <ID>2456</ID>
    <NAME>Chas Wofford</NAME>
    <SCHOOL>Sam Houston</SCHOOL>
```

```
    <CITY>Houston</CITY>
  </STUDENTS>
  <STUDENTS>
    <ID>3567</ID>
    <NAME>Jerry Kolar</NAME>
    <SCHOOL>Sharpstown</SCHOOL>
    <CITY>Houston</CITY>
  </STUDENTS>
</LIBRARY>

<?xml version="1.0" ?>
<LIBRARY>
  <STUDENTS>
    <ID>1234</ID>
    <NAME>Brad Martin</NAME>
    <SCHOOL>Reagan</SCHOOL>
    <CITY>Austin</CITY>
  </STUDENTS>
  <STUDENTS>
    <ID>2345</ID>
    <NAME>Zac Harvell</NAME>
    <SCHOOL>Westwood</SCHOOL>
    <CITY>Austin</CITY>
  </STUDENTS>
  <STUDENTS>
    <ID>3456</ID>
    <NAME>Walter Smith</NAME>
    <SCHOOL>Bowie</SCHOOL>
    <CITY>Austin</CITY>
  </STUDENTS>
</LIBRARY>
```

まず、デフォルトの XML Engine の動作を使用する例を示します。この場合、XMLCONCATENATE=NO となるため、XML Engine は連結されている XML ドキュメントをサポートしません。次の SAS プログラムを実行すると、先頭の XML ドキュメント(3 つのオブザベーションを含んでいるもの)のみがインポートされ、2 番目の XML ドキュメントに関してはエラーが生成されます。

```
libname concat xmlv2 '/u/mydocuments/XML/ConcatStudents.xml';

proc datasets library=concat;
```

ログ 3.2 SAS ログの出力

```
NOTE: Libref CONCAT was successfully assigned as follows: Engine:        XML Physical Name: /u/mydocuments/XML/
ConcatStudents.xml 20   proc datasets library=concat; ERROR: "xml" is illegal as a processing-instruction target name. encountered
during XMLMap parsing occurred at or near line 23, column 7 Directory Libref         CONCAT Engine         XML Physical Name  /u/
mydocuments/XML/ConcatStudents.xml XMLType        GENERIC XMLMap        NO XMLMAP IN EFFECT Member #  Name      Type 1
STUDENTS  DATA
```

LIBNAME ステートメントでオプション XMLCONCATENATE=YES を指定すると、XML Engine が、連結された複数の XML ドキュメントを 1 つの SAS データセットとしてインポートできるようになります。

```
libname concat xmlv2 '/u/mydocuments/XML/ConcatStudents.xml' xmlconcatenate=yes;

proc print data=concat.students;
run;
```

アウトプット 3.5 CONCAT.STUDENTS の PRINT プロシジャ出力

The SAS System

Obs	CITY	SCHOOL	NAME	ID
1	Houston	Bellaire	Linda Kay	1345
2	Houston	Sam Houston	Chas Wofford	2456
3	Houston	Sharpstown	Jerry Kolar	3567
4	Austin	Reagan	Brad Martin	1234
5	Austin	Westwood	Zac Harvell	2345
6	Austin	Bowie	Walter Smith	3456

4 章

XMLMap を使用し、XML ドキュメントをエクスポートする

エクスポート時に XMLMap を使用する理由

XMLMap でインポートした XML ドキュメントのエクスポートには XMLMap を使います。XMLMap の構文は、SAS データセットを特定の XML ドキュメントの構造に対応付ける方法を XML Engine に伝えます。

XMLMap を使用して XML ドキュメントをエクスポートするには、XML Engine のニックネームである XMLV2 を LIBNAME ステートメントで指定し、さらに XMLMAP=オプションを使ってファイルを指定します。

XMLMap を使用し、階層構造を持つ XML ドキュメントをエクスポートする

この例では、既存の XMLMap を使って、SAS データセットを特定の XML ドキュメントの構造に対応付ける方法を、XML Engine に伝える方法を説明します。この XMLMap は、セクション“XMLMap を使用し、XML ドキュメントを 1 つの SAS データセットとしてインポートする” (32 ページ)で、データセット NHL.TEAMS をインポートする際に使用されたものです。

エクスポートする SAS データセット NHL.TEAMS の内容は次のとおりです。

アウトプット 4.1 データセット NHL.TEAMS の PRINT プロシジャ出力

The SAS System

Obs	NAME	ABBREV	CONFERENCE	DIVISION
1	Thrashers	ATL	Eastern	Southeast
2	Hurricanes	CAR	Eastern	Southeast
3	Panthers	FLA	Eastern	Southeast
4	Lightning	TB	Eastern	Southeast
5	Capitals	WSH	Eastern	Southeast
6	Stars	DAL	Western	Pacific
7	Kings	LA	Western	Pacific
8	Ducks	ANA	Western	Pacific
9	Coyotes	PHX	Western	Pacific
10	Sharks	SJ	Western	Pacific

XMLMap を使用せずにデータをエクスポートした場合、結果として作成される XML ドキュメントの構造は矩形となり、SAS データセット内のオブザベーションごとに 1 つの TEAMS 要素が含まれることになります。例:

```
<?xml version="1.0" encoding="windows-1252" ?>
<TABLE>
 <TEAMS>
    <NAME>Thrashers</NAME>
    <ABBREV>ATL</ABBREV>
    <CONFERENCE>Eastern</CONFERENCE>
    <DIVISION>Southeast</DIVISION>
 </TEAMS>
 <TEAMS>
    <NAME>Hurricanes</NAME>
    <ABBREV>CAR</ABBREV>
    <CONFERENCE>Eastern</CONFERENCE>
    <DIVISION>Southeast</DIVISION>
 </TEAMS>
.
.
.
</TABLE>
```

各会議における部門別のデータ階層構造を持つ XML ドキュメントとして SAS データセットをエクスポートするには、XMLMap が必要となります。既存の XMLMap に対する唯一の変更は、OUTPUT 要素を含めることです。次の例では、XMLMap 構文の表記について説明します。

```
<?xml version="1.0" ?>
<SXLEMAP version="2.1"> 1
```

```
 <OUTPUT> 2
  <HEADING> 2
   <ATTRIBUTE  name="description" 3
          value="Teams of the National Hockey League" />
  </HEADING>
  <TABLEREF name="TEAMS" /> 4
 </OUTPUT>

<TABLE name="TEAMS">

   <TABLE-PATH syntax="XPath">/NHL/CONFERENCE/DIVISION/TEAM</TABLE-PATH>

   <COLUMN name="NAME">
    <PATH syntax="XPath">/NHL/CONFERENCE/DIVISION/TEAM/@name</PATH>
     <TYPE>character</TYPE>
     <DATATYPE>STRING</DATATYPE>
     <LENGTH>30</LENGTH>
   </COLUMN>

   <COLUMN name="ABBREV">
    <PATH syntax="XPath">/NHL/CONFERENCE/DIVISION/TEAM/@abbrev</PATH>
     <TYPE>character</TYPE>
     <DATATYPE>STRING</DATATYPE>
     <LENGTH>3</LENGTH>
   </COLUMN>

   <COLUMN name="CONFERENCE" retain="YES">
    <PATH syntax="XPath">/NHL/CONFERENCE</PATH>
     <TYPE>character</TYPE>
     <DATATYPE>STRING</DATATYPE>
     <LENGTH>10</LENGTH>
   </COLUMN>

   <COLUMN name="DIVISION" retain="YES">
     <PATH syntax="XPath">/NHL/CONFERENCE/DIVISION</PATH>
     <TYPE>character</TYPE>
     <DATATYPE>STRING</DATATYPE>
     <LENGTH>10</LENGTH>
   </COLUMN>
 </TABLE>
</SXLEMAP>
```

1 XMLMap を使用して SAS データセットを XML ドキュメントとしてエクスポートするには、XMLMap のバージョン番号として 1.9 または 2.1 を指定する必要があります。

2 XMLMap を使用して SAS データセットを XML ドキュメントとしてエクスポートするには、XMLMap に OUTPUT 要素を含める必要があります。
OUTPUT 要素には、1 つ以上の HEADING 要素と、1 つの TABLEREF 要素が含まれています。

3 ATTRIBUTE 要素は、追加となるファイル属性情報を定義するものであり、ここではエクスポートされる XML ドキュメントの名前と説明を指定します。

4 TABLEREF 要素は、エクスポートされるテーブルの名前を参照するものであり、ここではテーブル TEAMS を指定します。

次の SAS ステートメントは、NHLEXPORT.MAP という名前の XMLMap を使用して、SAS データセット NHL.TEAMS を XML ドキュメント NHLOUT.XML にエクスポートします。

```
libname nhl '/u/mydocuments/myfiles';

filename out '/u/mydocuments/XML/nhlout.xml';

libname out xmlv2 xmltype=xmlmap xmlmap='/u/mydocuments/XML/nhlexport.map';

data out.TEAMS;
  set nhl.teams;
run;
```

結果として生成される XML ドキュメントの内容は次のようになります。

```
<?xml version="1.0" encoding="windows-1252" ?>
<NHL description="Teams of the National Hockey League">
  <CONFERENCE>Eastern
    <DIVISION>Southeast
      <TEAM name="Thrashers" abbrev="ATL" />
      <TEAM name="Hurricanes" abbrev="CAR" />
      <TEAM name="Panthers" abbrev="FLA" />
      <TEAM name="Lightning" abbrev="TB" />
      <TEAM name="Capitals" abbrev="WSH" />
    </DIVISION>
  </CONFERENCE>
  <CONFERENCE>Western
    <DIVISION>Pacific
      <TEAM name="Stars" abbrev="DAL" />
      <TEAM name="Kings" abbrev="LA" />
      <TEAM name="Ducks" abbrev="ANA" />
      <TEAM name="Coyotes" abbrev="PHX" />
      <TEAM name="Sharks" abbrev="SJ" />
    </DIVISION>
  </CONFERENCE>
</NHL>
```

5 章

XMLMap を使用し、XML ドキュメントをインポートする

インポート時に XMLMap を使用する理由

XML Engine は、LIBNAME ステートメントの XMLTYPE=オプションでサポートされているマークアップタイプに従っている XML ドキュメントのみをインポートします。サポートされているマークアップタイプが必要とする仕様に従っていないフリーフォームの XML ドキュメントをインポートしようとすると、エラーが発生します。XMLTYPE=マークアップタイプに従っていないファイルを正常にインポートするには、*XMLMap* と呼ばれる XML ドキュメントを別途作成する必要があります。

ある XML ドキュメントを正常にインポートできない場合、そのドキュメントを直接変換するのではなく、その XML マークアップを変換する方法を XML Engine に伝えることにより、同 XML ドキュメントを正常にインポートできるようになります。特定の XMLMap 構文(すなわち XML マークアップ)を含む XMLMap を作成できます。XMLMap の構文は、XML マークアップを SAS データセット、変数(列)、オブザベーション(行)に変換する方法を XML Engine に伝えます。

XMLMap を作成した後、LIBNAME ステートメントの XMLMAP=オプションの値としてその XMLMap ファイルを指定します。

GENERIC マークアップタイプを使用した XML ドキュメントのインポートに必要な物理構造の条件について

必要とされる物理構造について

XML ドキュメントを正常にインポートするには、正しく構成されている XML ドキュメントに関する必要条件を次のように変換する必要があります。

- XML ドキュメントのルート包含要素(トップレベルノード)とは、ドキュメントコンテナのことです。SAS システムでは、これは SAS ライブラリに該当します。
- コンテナ内におけるネストされる要素(反復要素インスタンス)は、第 2 レベルのインスタンスタグにより開始されます。
- このような反復要素インスタンスは、1 つの矩形構造を表す必要があります。SAS データセットの場合、このようなインスタンスにより、一定の個数の*列*を伴う任意の個数の*行*の集合(すなわち表)を生成するオブザベーション境界が決定されます。

必要とされる物理構造を表す XML ドキュメントの例を次に示します。

```
<?xml version="1.0" encoding="windows-1252" ?>
<LIBRARY> 1
  <STUDENTS> 2
    <ID> 0755 </ID>
    <NAME> Brad Martin </NAME>
    <ADDRESS> 1611 Glengreen </ADDRESS>
    <CITY> Huntsville </CITY>
    <STATE> Texas </STATE>
  </STUDENTS>

  <STUDENTS> 3
    <ID> 1522 </ID>
    <NAME> Zac Harvell </NAME>
    <ADDRESS> 11900 Glenda </ADDRESS>
    <CITY> Houston </CITY>
    <STATE> Texas </STATE>
  </STUDENTS>
.
. more instances of <STUDENTS>
.
</LIBRARY>
```

前述の XML ドキュメントをインポートする場合、次の処理が行われます。

1 XML Engine は、<LIBRARY>をルート包含要素として認識します。

2 XML Engine は第 2 レベルのインスタンスタグ(<STUDENTS>)に移動し、それをデータセット名に変換し、開始タグ<STUDENTS>と終了タグ</STUDENTS>の間にネストされている(含まれている)要素のスキャンを行って変数を見つけようとします。

3 インスタンスタグ<ID>、<NAME>、<ADDRESS>、<CITY>、<STATE>は、開始タグ<STUDENTS>と終了タグ</STUDENTS>に囲まれているため、XML Engine はこれらのインスタンスタグを変数として変換します。個々のインスタンスタグ名がデータセット変数名となります。反復要素インスタンスは、一定の個数の列を伴う任意の個数の行の集合(すなわち表)に変換されます。

次のステートメントを実行すると、次の SAS 出力が生成されます。

```
libname test xmlv2 '/u/mydocuments/students.xml';

proc print data=test.students;
run;
```

アウトプット 5.1 TEST.STUDENTS の PRINT プロシジャ出力

The SAS System

Obs	STATE	CITY	ADDRESS	NAME	ID
1	Texas	Huntsville	1611 Glengreen	Brad Martin	755
2	Texas	Houston	11900 Glenda	Zac Harvell	1522

特定の物理構造が必要な理由

XML ドキュメントが正しく構成されているかどうかは、その内容ではなく、構造により判定されます。このため、XML Engine は XML ドキュメントが正しく構成されているとは仮定できますが、ルート要素が単一ノード要素のインスタンスのみ(すなわち単一のデータセットのみ)を包含しているとは仮定できません。したがって、XML Engine は、複数のノード(すなわち複数の SAS データセット)の可能性を考慮に入れる必要があります。

たとえば、次のような正しく構成されている XML ドキュメントをインポートする場合、同ドキュメントは、HighTemp および LowTemp という 2 つの SAS データセットから成ると認識されます。

```
<?xml version="1.0" encoding="windows-1252" ?>
<CLIMATE> 1
  <HIGHTEMP> 2
    <PLACE> Libya </PLACE>
    <DATE> 1922-09-13 </DATE>
    <DEGREE-F> 136 </DEGREE-F>
    <DEGREE-C> 58 </DEGREE-C>
  </HIGHTEMP>
```

```
.
. more instances of <HIGHTEMP>
.
  <LOWTEMP> 3
    <PLACE> Antarctica </PLACE>
    <DATE> 1983-07-21 </DATE>
    <DEGREE-F> -129 </DEGREE-F>
    <DEGREE-C> -89 </DEGREE-C>
  </LOWTEMP>
.
. more instances of <LOWTEMP>
.
</CLIMATE>
```

前述の XML ドキュメントをインポートする場合、次の処理が行われます。

1 XML Engine は、最初のインスタンスタグ<CLIMATE>をルート包含要素(すなわちドキュメントコンテナ)として認識します。

2 第 2 レベルのインスタンスタグ<HIGHTEMP>以降では、XML Engine は、反復要素インスタンスを、一定の個数の列を伴う任意の個数の行の集合(すなわち表)として使用します。

3 第 2 レベルのインスタンスタグが変化すると、XML Engine は、その変化を異なる SAS データセットとして解釈します。

結果として、HighTemp および LowTemp という 2 つの SAS データセットが得られます。これらのデータセットは同じ変数を持ちますが、含まれているデータは異なります。

必要な物理構造を備えていない XML ドキュメントの処理

XML ドキュメントが必要な物理構造を備えていない場合、そのドキュメントを正しくインポートするには、同ドキュメントに含まれている XML マークアップを解釈する方法を XML Engine に伝える必要があります。“インポート時に XMLMap を使用する理由” (29 ページ)を参照してください。

XMLMap を使用し、XML ドキュメントを 1 つの SAS データセットとしてインポートする

次の例では、XMLMap の作成および使用方法を示します。XMLMap を使用すると、XML マークアップを SAS データセット、変数、オブザベーションに対応付ける方法を XML Engine に伝えることができます。

インポートする XML ドキュメント NHL.XML の内容は次のとおりです。この XML ドキュメントは、単純で比較的読みやすい構成を持っていますが、その XML マークアップは必要な物理構造を備えていないため、同ドキュメントは正しくインポートできません。

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<NHL>
 <CONFERENCE> Eastern
  <DIVISION> Southeast
   <TEAM name="Thrashers"  abbrev="ATL" />
```

```
    <TEAM name="Hurricanes" abbrev="CAR" />
    <TEAM name="Panthers"   abbrev="FLA" />
    <TEAM name="Lightning"  abbrev="TB" />
    <TEAM name="Capitals"   abbrev="WSH" />
  </DIVISION>
 </CONFERENCE>

 <CONFERENCE> Western
  <DIVISION> Pacific
   <TEAM name="Stars"   abbrev="DAL" />
   <TEAM name="Kings"   abbrev="LA" />
   <TEAM name="Ducks"   abbrev="ANA" />
   <TEAM name="Coyotes" abbrev="PHX" />
   <TEAM name="Sharks"  abbrev="SJ" />
  </DIVISION>
 </CONFERENCE>
</NHL>
```

この XML ドキュメントを正しくインポートするには、XMLMap が必要となります。インポートするデータを十分に把握したら、そのデータを正しくインポートするために、XMLMap 構文をコーディングします。この XML ドキュメントをインポートするのに使用される XMLMap の内容は次のようになります。

```
<?xml version="1.0" ?>
<SXLEMAP version="2.1">
 <TABLE name="TEAMS"> 1
    <TABLE-PATH syntax="XPath"> 2
      /NHL/CONFERENCE/DIVISION/TEAM
     </TABLE-PATH>

    <COLUMN name="NAME"> 3
     <PATH> 5
       /NHL/CONFERENCE/DIVISION/TEAM@name
      </PATH>
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>30</LENGTH>
    </COLUMN>

    <COLUMN name="ABBREV"> 3
     <PATH> 5
      /NHL/CONFERENCE/DIVISION/TEAM/@abbrev
      </PATH>
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>3</LENGTH>
    </COLUMN>

   <COLUMN name="CONFERENCE" retain="YES"> 4
     <PATH>/NHL/CONFERENCE</PATH> 5
      <TYPE>character</TYPE>
      <DATATYPE>STRING</DATATYPE>
      <LENGTH>10</LENGTH>
    </COLUMN>

    <COLUMN name="DIVISION" retain="YES"> 4
```

```
        <PATH> 5
          /NHL/CONFERENCE/DIVISION
        </PATH>
        <TYPE>character</TYPE>
        <DATATYPE>STRING</DATATYPE>
        <LENGTH>10</LENGTH>
      </COLUMN>
  </TABLE>
</SXLEMAP>
```

前述の XMLMap 構文は、次のようなデータ調査手順を使用して、XML マークアップを変換する方法を定義します。

1 情報を含んでいる表を特定します。

ナショナルホッケーリーグの複数のチームを含む SAS データセット(表)が必要であるとします。それが XML ドキュメントに含まれている唯一の情報であるため、XMLMap 内で TEAMS という名前の単一データセットを定義します(これ以外の XML ドキュメントでは、関連情報の表を複数含んでいる場合もあることに注意してください。複数の表のインポートをサポートする XMLMap 構文については、“XMLMap を使用し、XML ドキュメントを複数の SAS データセットとしてインポートする” (36 ページ)を参照してください)。

2 SAS データセットのオブザベーション境界を指定します。オブザベーション境界は、一定の個数の列を伴う任意の個数の行の集合(すなわち表)に変換されます。

この XML ドキュメント内では、各チームに関する情報は、<CONFERENCE>タグおよび<DIVISION>タグにより囲まれた領域内にある<TEAM>タグ内に記述されています。TEAM 要素を読み込むたびに、新しいオブザベーションを生成する必要があります。

3 各表の列定義を収集します。

この XML ドキュメントでは、データのコンテンツ形式が混合されています。XML PCDATA として記述されているデータもあれば(例: CONFERENCE)、属性と値のペアとして記述されているデータ(例: NAME)もあります。データ型はすべて文字列値です。構成されるオブザベーションは、チームの名前(NAME)と略称(ABBREV)を含むことになります。NAME の長さは 30 文字で十分であり、ABBREV フィールドの内容を表すには 3 文字で十分です。

4 外部キーまたは必要な外部コンテキストを追加します。

チームの所属リーグに関する情報を含めるとします。また、CONFERENCE データと DIVISION データの抽出も行うとします。

注: 列定義内の retain=属性は、オブザベーションを出力データセットに書き出した後、処理されたデータが保持されるようにします。外部キーフィールドはオブザベーション境界の外側で発生する(すなわち、階層的な XML データ内では、外部キーフィールドが、SAS オブザベーション内に比べて、よりまばらに出現する)ため、追加行での外部キーフィールドの値は、当該フィールドを検出する時点まで保持する必要があります。

5 各変数定義の場所パスを定義します。

PATH 要素は、各列の値が取り出される XML ドキュメント内の位置を指定します。要素-PCDATA (構文解析対象文字データ)は、属性値とは異なる方法で扱われます。関連する条件付きの選択基準は存在しません。

次の SAS ステートメントは、XML ドキュメント NHL.XML をインポートします。

```
filename nhl '/u/mydocuments/XML/Nhl.xml'; 1
filename map '/u/mydocuments/XML/Nhl.map'; 2

libname nhl xmlv2 xmlmap=map; 3

proc print data=nhl.teams; 4
run;
```

1 最初の FILENAME ステートメントは、ファイル参照名 NHL を、NHL.XML という名前の XML ドキュメントの物理的な場所(完全なパス名、ファイル名、ファイル拡張子を含むもの)に割り当てます。

2 2 番目の FILENAME ステートメントは、ファイル参照名 MAP を、NHL.MAP という名前の XMLMap ファイルの物理的な場所に割り当てます。

3 この LIBNAME ステートメントは、ファイル参照名 NHL を使用して XML ドキュメントを参照します。同ステートメントでは、エンジンとして XML Engine を指定した上で、ファイル参照名 MAP を使用して XMLMap を参照しています。

4 PRINT プロシジャにより出力を生成することで、インポートが正常に行われたことを確認します。

アウトプット 5.2 NHL.TEAMS の PRINT プロシジャ出力

The SAS System

Obs	NAME	ABBREV	CONFERENCE	DIVISION
1	Thrashers	ATL	Eastern	Southeast
2	Hurricanes	CAR	Eastern	Southeast
3	Panthers	FLA	Eastern	Southeast
4	Lightning	TB	Eastern	Southeast
5	Capitals	WSH	Eastern	Southeast
6	Stars	DAL	Western	Pacific
7	Kings	LA	Western	Pacific
8	Ducks	ANA	Western	Pacific
9	Coyotes	PHX	Western	Pacific
10	Sharks	SJ	Western	Pacific

XMLMap を使用し、XML ドキュメントを複数の SAS データセットとしてインポートする

次の例では、XMLMap を使用して、XML マークアップを 2 つの SAS データセットに対応付ける方法を定義します。この例で使用する XML ドキュメント RSS.XML は、正しく構成されていないため XML Engine により正しく変換されず、その結果として正しくインポートできません。

注: XML ドキュメント RSS.XML は、RSS (Rich Site Summary)という XML フォーマットを使用します。RSS は、元々は My Netscape Network (MNN)コミュニティ内でコンテンツを交換するために、Netscape により設計されたものです。RSS フォーマットは、見出しやその他の Web コンテンツを共有するために広く採用されており、伝送フォーマットとしての XML の優れた事例となっています。

インポートする XML ドキュメント RSS.XML の内容は次のとおりです。

```
 <?xml version="1.0" encoding="ISO-8859-1" ?>
- <rss version="0.91">
- <channel>
 <title>WriteTheWeb</title>
 <link>http://writetheweb.com</link>
 <description>News for web users that write back</description>
 <language>en-us</language>
 <copyright>Copyright 2000, WriteTheWeb team.</copyright>
 <managingEditor>editor@writetheweb.com</managingEditor>
 <webMaster>webmaster@writetheweb.com</webMaster>
- <image>
 <title>WriteTheWeb</title>
 <url>http://writetheweb.com/images/mynetscape88.gif</url>
 <link>http://writetheweb.com</link>
 <width>88</width>
 <height>31</height>
 <description>News for web users that write back</description>
 </image>
- <item>
 <title>Giving the world a pluggable Gnutella</title>
 <link>http://writetheweb.com/read.php?item=24</link>
 <description>WorldOS is a framework on which to build programs that work like
   Freenet or Gnutella -allowing distributed applications using peer-to-peer
   routing.
   </description>
 </item>
- <item>
 <title>Syndication discussions hot up</title>
 <link>http://writetheweb.com/read.php?item=23</link>
 <description>After a period of dormancy, the Syndication mailing list has become
   active again, with contributions from leaders in traditional media and Web
   syndication.
   </description>
 </item>
- <item>
```

```
  <title>Personal web server integrates file sharing and messaging</title>
  <link>http://writetheweb.com/read.php?item=22</link>
  <description>The Magi Project is an innovative project to create a combined personal
    web server and messaging system that enables the sharing and synchronization of
    information across desktop, laptop and palmtop devices.</description>
  </item>
- <item>
  <title>Syndication and Metadata</title>
  <link>http://writetheweb.com/read.php?item=21</link>
  <description>RSS is probably the best known metadata format around. RDF is probably
    one of the least understood. In this essay, published on my O'Reilly Network
    weblog, I argue that the next generation of RSS should be based on RDF.
  </description>
  </item>
- <item>
  <title>UK bloggers get organised</title>
  <link>http://writetheweb.com/read.php?item=20</link>
  <description>Looks like the weblogs scene is gathering pace beyond the shores of the
    US. There's now a UK-specific page on weblogs.com, and a mailing list at egroups.
    </description>
  </item>
- <item>
  <title>Yournamehere.com more important than anything</title>
  <link>http://writetheweb.com/read.php?item=19</link>
  <description>Whatever you're publishing on the web, your site name is the most
    valuable asset you have, according to Carl Steadman.</description>
  </item>
  </channel>
  </rss>
```

この XML ドキュメントを正しくインポートするには、同ドキュメントに含まれている XML マークアップを対応付ける方法を定義した XMLMap を作成します。RSS.XML を正しくインポートするために必要となる構文を含む XMLMap ファイル RSS.MAP の内容は次のようになります。この XMLMap 構文は、XML マークアップの変換方法を XML Engine に伝えます。変換方法の詳細については、後続の説明を参照してください。RSS.XML をインポートすることで、2 つの SAS データセット CHANNEL および ITEMS が生成されます。

```
<?xml version="1.0" encoding="UTF-8"?>

<SXLEMAP name="SXLEMap" version="2.1"> 1

    <TABLE name="CHANNEL"> 2
    <TABLE-PATH syntax="XPath">/rss/channel</TABLE-PATH> 3

    <COLUMN name="title"> 4
      <PATH syntax="XPath">/rss/channel/title</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>200</LENGTH>
    </COLUMN>

    <COLUMN name="link"> 5
      <PATH syntax="XPath">/rss/channel/link</PATH>
      <DESCRIPTION>Story link</DESCRIPTION>
      <TYPE>character</TYPE>
```

```
      <DATATYPE>string</DATATYPE>
      <LENGTH>200</LENGTH>
    </COLUMN>

    <COLUMN name="description">
      <PATH syntax="XPath">/rss/channel/description</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>1024</LENGTH>
    </COLUMN>

    <COLUMN name="language">
      <PATH syntax="XPath">/rss/channel/language</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>8</LENGTH>
    </COLUMN>

    <COLUMN name="version"> 6
      <PATH syntax="XPath">/rss@version</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>8</LENGTH>
    </COLUMN>

  </TABLE>

  <TABLE description="Individual news stories" name="ITEMS"> 7
    <TABLE-PATH syntax="XPath">/rss/channel/item</TABLE-PATH>

    <COLUMN name="title"> 8
      <PATH syntax="XPath">/rss/channel/item/title</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>200</LENGTH>
    </COLUMN>

    <COLUMN name="URL"> 9
      <PATH syntax="XPath">/rss/channel/item/link</PATH>
      <DESCRIPTION>Story link</DESCRIPTION>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>200</LENGTH>
    </COLUMN>

    <COLUMN name="description"> 9
      <PATH syntax="XPath">/rss/channel/item/description</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>1024</LENGTH>
    </COLUMN>

  </TABLE>

</SXLEMAP>
```

前述の XMLMap が XML マークアップをどのように変換するかについて、次に説明します。

1 SAS データセットを定義するルート包含要素です。

2 CHANNEL データセットを定義する要素です。

3 この要素は、CHANNEL データセットの変数を収集する XML ドキュメント内の位置を定義する場所パスを指定します。

4 この要素は、CHANNEL データセット内の変数 TITLE の属性を指定します。XPath 構成は、現在のタグが見つかる場所、および指定の要素からデータにアクセスできる場所を指定します。

5 続く COLUMN 要素は、CHANNEL データセットの変数 LINK、DESCRIPTION、LANGUAGE を定義します。

6 この要素は、CHANNEL データセット内にある最後の変数(VERSION)の属性を指定します。この XPath 構成は、現在のタグが見つかる場所を指定し、属性フォームを使用して指定の要素からデータにアクセスします。

7 ITEMS データセットを定義する要素です。

8 この要素は、ITEMS データセットの変数 TITLE の属性を指定します。

9 後続の COLUMN 要素は、ITEMS データセット内の他の変数(URL および DESCRIPTION)を定義します。

次の SAS ステートメントでは、RSS.MAP という名前の XMLMap ファイルを指定した上で、XML ドキュメント RSS.XML をインポートします。

```
filename rss '/u/mydocuments/rss.xml';
filename map '/u/mydocuments/rss.map';

libname rss xmlv2 xmlmap=map access=readonly;
```

階層データを関連データセットとしてインポートする

多くの場合、XML ドキュメントには階層データが含まれています。階層データでは、会社の組織図のように、データが各種のレベルに構造化されています。階層構造は 1 対多の関係を表します。最上位の項目は、1 つまたは複数の下位項目(顧客や注文など)を持ちます。

この例では、XML ドキュメントを、関連情報を含んでいる 2 つのデータセットとしてインポートするためには、どのように XMLMap を定義すればよいかを示します。

XML ドキュメント Pharmacy.XML の内容を次に示します。このファイルには、個々の顧客とその処方箋という形式で複数の関連エンティティを 持つ階層データが含まれています。各顧客は、1 つまたは複数の処方箋を持つことができます。PRESCRIPTION 要素が、開始タグ<PERSON>と終了タグ</PERSON>に囲まれた各領域内でネストされていることに注意してください。

```
<?xml version="1.0" ?>
<PHARMACY>
 <PERSON>
  <NAME>Brad Martin</NAME>
```

```
  <STREET>11900 Glenda Court</STREET>
  <CITY>Austin</CITY>
 <PRESCRIPTION>
  <NUMBER>1234</NUMBER>
  <DRUG>Tetracycline</DRUG>
 </PRESCRIPTION>
 <PRESCRIPTION>
  <NUMBER>1245</NUMBER>
  <DRUG>Lomotil</DRUG>
 </PRESCRIPTION>
 </PERSON>
<PERSON>
  <NAME>Jim Spano</NAME>
  <STREET>1611 Glengreen</STREET>
  <CITY>Austin</CITY>
 <PRESCRIPTION>
  <NUMBER>1268</NUMBER>
  <DRUG>Nexium</DRUG>
 </PRESCRIPTION>
</PERSON>
</PHARMACY>
```

顧客情報を含むものと処方箋情報を含むものという 2 つの独立したデータセットをインポートするには、各顧客とそれに関連付けられている処方箋との間の関係を指定することにより、個々の顧客に帰属する処方箋を明確化する必要があります。

XMLMap は、XML マークアップを 2 つの SAS データセットに変換する方法を定義します。データセット Person は、各顧客の名前と住所をインポートします。一方、データセット Prescription は、顧客名、処方箋番号、処方薬をインポートします。XMLMap 構文の表記については次の説明を参照してください。

```
<?xml version="1.0" encoding="UTF-8"?>
<SXLEMAP name="AUTO_GEN" version="2.1"> 1

  <NAMESPACES count="0"/>

  <!-- ############################################################ -->
  <TABLE description="PERSON" name="PERSON"> 2
    <TABLE-PATH syntax="XPath">/PHARMACY/PERSON</TABLE-PATH>

    <COLUMN name="NAME"> 3
      <PATH syntax="XPath">/PHARMACY/PERSON/NAME</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>11</LENGTH>
    </COLUMN>

    <COLUMN name="STREET"> 3
      <PATH syntax="XPath">/PHARMACY/PERSON/STREET</PATH>
      <TYPE>character</TYPE>
      <DATATYPE>string</DATATYPE>
      <LENGTH>18</LENGTH>
    </COLUMN>

    <COLUMN name="CITY"> 3
```

```
    <PATH syntax="XPath">/PHARMACY/PERSON/CITY</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>string</DATATYPE>
    <LENGTH>6</LENGTH>
  </COLUMN>

</TABLE>

<!-- ############################################################ -->
<TABLE description="PRESCRIPTION" name="PRESCRIPTION"> 4
  <TABLE-PATH syntax="XPath">/PHARMACY/PERSON/PRESCRIPTION</TABLE-PATH>

  <COLUMN name="NAME" retain="YES"> 5
    <PATH syntax="XPath">/PHARMACY/PERSON/NAME</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>string</DATATYPE>
    <LENGTH>11</LENGTH>
  </COLUMN>

  <COLUMN name="NUMBER"> 6
    <PATH syntax="XPath">/PHARMACY/PERSON/PRESCRIPTION/NUMBER</PATH>
    <TYPE>numeric</TYPE>
    <DATATYPE>integer</DATATYPE>
  </COLUMN>

  <COLUMN name="DRUG"> 6
    <PATH syntax="XPath">/PHARMACY/PERSON/PRESCRIPTION/DRUG</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>string</DATATYPE>
    <LENGTH>12</LENGTH>
  </COLUMN>

</TABLE>

</SXLEMAP>
```

1 SXLEMAP は、2 つの SAS データセットを定義するルート包含要素です。

2 最初の TABLE 要素は、Person データセットを定義します。

3 続く COLUMN 要素は、Person データセット内の変数 Name、Street、City の属性を指定します。

4 2 番目の TABLE 要素は、Prescription データセットを定義します。

5 続く COLUMN 要素は、Prescription データセット内の変数 Name の属性を指定します。変数 Name の値が異なる値に置き換えられるまで各オブザベーションで保持されるようにするために、**retain="YES"**属性を指定します (retain=属性は SAS DATA ステップの RETAIN ステートメントに類似した機能であり、これを使うことにより、DATA ステップの反復間で特定の変数の値が保持されるように設定できます)。

6 続く COLUMN 要素は、Prescription データセット内の変数 Number および Drug の属性を指定します。

次の SAS ステートメントは、XML ドキュメントをインポートして、XMLMap を指定します。

```
filename pharm '/u/mydocuments/Pharmacy.xml';
filename map '/u/mydocuments/Pharmacy.map';
```

```
libname pharm xmlv2 xmlmap=map;
```

インポートされた 2 つの SAS データセットの PRINT プロシジャ出力は次のようになります。

アウトプット 5.3 PHARM.PERSON の PRINT プロシジャ出力

The SAS System

Obs	NAME	STREET	CITY
1	Brad Martin	11900 Glenda Court	Austin
2	Jim Spano	1611 Glengreen	Austin

アウトプット 5.4 PHARM.PRESCRIPTION の PRINT プロシジャ出力

The SAS System

Obs	NAME	NUMBER	DRUG
1	Brad Martin	1234	Tetracycline
2	Brad Martin	1245	Lomotil
3	Jim Spano	1268	Nexium

生成された数値キーを含むキーフィールドを挿入する

次の例では、階層データを含んでいる XML ドキュメント Pharmacy.XML をインポートします。同ドキュメントは、“階層データを関連データセットとしてインポートする” (39 ページ)で使われているものと同じです。この例では、前と同じ XMLMap ファイルを引き続き使用しますが、生成された数値をキー値として含むキーフィールドを追加することにより、2 つのデータセット間の関係を提供しています(キーフィールドは、レコードを識別するための一意のデータを保持します。典型的なキーフィールドの例としては、アカウント番号、製品コード、顧客名などが挙げられます)。

キーフィールド値を生成するには、COLUMN 要素内で class="ORDINAL"属性を使用することにより、カウンタ変数を作成します。カウンタ変数は、INCREMENT-PATH 要素により指定される場所パスが検出される回数を記録します。カウンタ変数の値は、この場所パスが一致するたびに 1 ずつインクリメントされます(このカウンタ変数は、DATA ステップの自動変数_N_に似ています。

DATA ステップにおいて、自動変数_N_は、SAS データセットに読み込まれるオブザベーションの数をカウントします)。

注: カウンタ変数を使用して関連する 2 つのデータセット用のキーフィールドを作成する場合、両方の TABLE 要素に対して同じ場所パスを指定する必要があります。これを行わない場合、結果が一致しなくなります。各表は、同じ名前のデータ要素に関して、同一の生成されたキーを持つ必要があります。

次の XMLMap は、ドキュメント Pharmacy.XML を、関連した情報を含む 2 つの SAS データとしてインポートします。また、生成された数値をキー値として含むキーフィールドを作成します。

```
<?xml version="1.0" encoding="UTF-8"?>
<SXLEMAP name="AUTO_GEN" version="2.1">

  <NAMESPACES count="0"/>

  <!-- ############################################################ -->
  <TABLE description="PERSON" name="PERSON">
    <TABLE-PATH syntax="XPath">/PHARMACY/PERSON</TABLE-PATH> 1

   <COLUMN name="KEY" retain="YES" class="ORDINAL"> 2
     <INCREMENT-PATH
syntax="XPath">/PHARMACY/PERSON</INCREMENT-PATH>
    <TYPE>numeric</TYPE>
    <DATATYPE>integer</DATATYPE>
    <FORMAT width="3">Z</FORMAT>
    </COLUMN>

     <COLUMN name="NAME">
       <PATH syntax="XPath">/PHARMACY/PERSON/NAME</PATH>
       <TYPE>character</TYPE>
       <DATATYPE>string</DATATYPE>
       <LENGTH>11</LENGTH>
     </COLUMN>

     <COLUMN name="STREET">
       <PATH syntax="XPath">/PHARMACY/PERSON/STREET</PATH>
       <TYPE>character</TYPE>
       <DATATYPE>string</DATATYPE>
       <LENGTH>18</LENGTH>
     </COLUMN>

     <COLUMN name="CITY">
       <PATH syntax="XPath">/PHARMACY/PERSON/CITY</PATH>
       <TYPE>character</TYPE>
       <DATATYPE>string</DATATYPE>
       <LENGTH>6</LENGTH>
     </COLUMN>

  </TABLE>

  <!-- ############################################################ -->
  <TABLE description="PRESCRIPTION" name="PRESCRIPTION">
    <TABLE-PATH syntax="XPath">/PHARMACY/PERSON/PRESCRIPTION</TABLE-PATH> 3
```

```
<COLUMN name="KEY" retain="YES" class="ORDINAL"> 4
  <INCREMENT-PATH syntax="XPath">/PHARMACY/PERSON</INCREMENT-PATH>
  <TYPE>numeric</TYPE>
  <DATATYPE>integer</DATATYPE>
  <FORMAT width="3">Z</FORMAT>
</COLUMN>

<COLUMN name="NUMBER">
   <PATH syntax="XPath">/PHARMACY/PERSON/PRESCRIPTION/NUMBER</PATH>
   <TYPE>numeric</TYPE>
   <DATATYPE>integer</DATATYPE>
 </COLUMN>

 <COLUMN name="DRUG">
   <PATH syntax="XPath">/PHARMACY/PERSON/PRESCRIPTION/DRUG</PATH>
   <TYPE>character</TYPE>
   <DATATYPE>string</DATATYPE>
   <LENGTH>12</LENGTH>
 </COLUMN>

</TABLE>

</SXLEMAP>
```

キーフィールドを生成する XMLMap 構文の説明を次に示します。

1 Person データセットを定義する TABLE 要素内で、TABLE-PATH 要素は、同データセットのオブザベーション境界を指定します。この場所パスは、PERSON 要素が読み込まれるたびに新しいオブザベーションを生成します。

2 Person データセットの場合、変数 Key の COLUMN 要素は、class="ORDINAL"属性と、INCREMENT-PATH 要素を含んでいます。XML Engine は次の手順で、Person データセットのキーフィールド値を生成します。

   1. XML Engine は、開始タグ<PERSON>を検出すると、その値を入力バッファに読み込み、Key 変数の値を 1 増やします。
   2. XML Engine は、終了タグ</PERSON>を検出するまで、値を入力バッファに読み込み続けます。同終了タグを検出した時点で、XML Engine は、完了した入力バッファを 1 つのオブザベーションとして SAS データセットに書き出します。
   3. 開始タグ<PERSON>(INCREMENT-PATH 内にあるもの)と終了タグ</PERSON>(TABLE-PATH 内にあるもの)のシーケンスを検出するたびに、この処理が繰り返されます。
   4. 結果として、4 つの変数と 2 つのオブザベーションが生成されます。

3 Prescription データセットを定義する TABLE 要素内で、TABLE-PATH 要素は同データセットのオブザベーション境界を指定します。この場所パスは、PRESCRIPTION 要素が読み込まれるたびに新しいオブザベーションを生成します。

4 Prescription データセットの場合、変数 Key の COLUMN 要素は、class="ORDINAL"属性と、INCREMENT-PATH 要素を含んでいます。

   XML Engine は次の手順で、Prescription データセットのキーフィールド値を生成します。

1. XML Engine は、開始タグ<PERSON>を検出すると、その値を入力バッファに読み込み、Key 変数の値を 1 増やします。
2. XML Engine は、終了タグ</PRESCRIPTION>を検出するまで、値を入力バッファに読み込み続けます。同終了タグを検出した時点で、XML Engine は、完了した入力バッファを 1 つのオブザベーションとして SAS データセットに書き出します。カウンタ変数の場所パスは両 TABLE 要素で同一でなければならないため、Prescription データセットの変数 Key に関する XML Engine の動作は、Person データセットの変数 Key の場合と同じになります。XML Engine は PERSON タグのオカレンスを両方のカウンタ変数のキー値として記録しますが、オブザベーションは異なる TABLE-PATH の場所パスにより生成されます。
3. 開始タグ<PERSON> (INCREMENT-PATH 内にあるもの)と終了タグ</PRESCRIPTION> (TABLE-PATH 内にあるもの)のシーケンスを検出するたびに、この処理が繰り返されます。
4. 結果として、3 つの変数と 3 つのオブザベーションが生成されます。

次の SAS ステートメントは、XML ドキュメントをインポートします。

```
filename pharm '/u/mydocuments/XML/Pharmacy.xml';
filename map '/u/mydocuments/XML/PharmacyOrdinal.map';

libname pharm xmlv2 xmlmap=map;
```

数値キーを含むインポートされた 2 つの SAS データセットの PRINT プロシジャ出力は次のようになります。

アウトプット 5.5 PHARM.PERSON の PRINT プロシジャ出力

The SAS System

Obs	KEY	NAME	STREET	CITY
1	001	Brad Martin	11900 Glenda Court	Austin
2	002	Jim Spano	1611 Glengreen	Austin

アウトプット 5.6 PHARM.PRESCRIPTION の PRINT プロシジャ出力

The SAS System

Obs	KEY	NUMBER	DRUG
1	001	1234	Tetracycline
2	001	1245	Lomotil
3	002	1268	Nexium

データの連結を避けるために、オブザベーションの境界を決定する

次の例では、XML ドキュメントをインポートする際に、連結されたデータではなく独立したオブザベーションが生成されるようにオブザベーション境界を決定する方法を示します。

オブザベーション境界は、一定の個数の列を伴う任意の個数の行の集合(すなわち表)に変換されます。XMLMap を使用する場合、オブザベーション境界を決定するには、TABLE-PATH 要素で場所パスを指定します。この場所パスの終了タグにより、どの時点でデータが 1 つのオブザベーションとして SAS データセットに書き出されるかが決定されます。

開始タグと終了タグのペアリングシーケンスが原因で、オブザベーション境界の特定が困難になる場合があります。適切なオブザベーション境界を特定しない場合、結果として独立したオブザベーションではなく、連結されたデータ文字列が生成されることがあります。次の例では、望ましくない結果を引き起こすペアリング状況を示します。

次の XML ドキュメントを正しくインポートするには、XMLMap が必要となります。XMLMap を使用しない場合、XML Engine は同 XML ドキュメントをインポートした結果、Row0、Model0、Year0、Row1、Model1、Year1 のような変数を含む、FORD という名前のデータセットを 1 つ生成します。

```
<?xml version="1.0" ?>
<VEHICLES>
 <FORD>
  <ROW>
   <Model>Mustang</Model>
   <Year>1965</Year>
  </ROW>
  <ROW>
   <Model>Explorer</Model>
   <Year>1982</Year>
  </ROW>
  <ROW>
   <Model>Taurus</Model>
   <Year>1998</Year>
  </ROW>
  <ROW>
   <Model>F150</Model>
   <Year>2000</Year>
  </ROW>
 </FORD>
</VEHICLES>
```

前述の XML ドキュメントを調べると、VEHICLES、FORD、ROW という開始タグと終了タグからなる要素のシーケンスが 3 つ存在することが分かります。表の場所パスと列の場所パスを次のように指定した場合、XML Engine はこの XML ドキュメントを次のように処理します。

```
<TABLE-PATH syntax="XPath"> /VEHICLES/FORD </TABLE-PATH>
 <PATH syntax="XPath"> /VEHICLES/FORD/ROW/Model </PATH>
 <PATH syntax="XPath"> /VEHICLES/FORD/ROW/Year </PATH>
```

1. XML Engine は、開始タグ<FORD>を検出するまで、XML マークアップを読み込みます。これは、FORD が表の場所パスの最後の要素として指定されているためです。
2. XML Engine は入力バッファをクリアし、列の場所パスに基づいて、後続の要素を変数用にスキャンします。各変数の値を検出すると、その値が入力バッファに読み込まれます。たとえば、最初の ROW 要素を読み込んだ後、入力バッファには値 **Mustang** および **1965** が含まれています。
3. XML Engine は、終了タグ</FORD>を検出するまで、入力バッファへの値の読み込みを続行します。同終了タグを検出した時点で、XML Engine は、完了した入力バッファを 1 つのオブザベーションとして SAS データセットに書き出します。
4. この結果、1 つのオブザベーションのみが書き出されます。これは望ましい結果ではありません。

連結されていない、独立したオブザベーションを生成するには、XML Engine が独立したオブザベーションを SAS データセットに書き出すように表の場所パスを変更する必要があります。正しい場所パスと、それに従って XML Engine が実施する処理を次に示します。

```
<TABLE-PATH syntax="XPath"> /VEHICLES/FORD/ROW </TABLE-PATH>
 <PATH syntax="XPath"> /VEHICLES/FORD/ROW/Model </PATH>
 <PATH syntax="XPath"> /VEHICLES/FORD/ROW/Year </PATH>
```

1. XML Engine は、開始タグ<ROW>を検出するまで、XML マークアップを読み込みます。これは、ROW が表の場所パスの最後の要素として指定されているためです。
2. XML Engine は入力バッファをクリアし、列の場所パスに基づいて、後続の要素を変数用にスキャンします。各変数の値を検出すると、その値が入力バッファに読み込まれます。
3. XML Engine は、終了タグ</ROW>を検出するまで、入力バッファへの値の読み込みを続行します。同終了タグを検出した時点で、XML Engine は、完了した入力バッファを 1 つのオブザベーションとして SAS データセットに書き出します。すなわち、値 **Mustang** および **1965** を含む 1 つのオブザベーションが SAS データセットに書き出されます。
4. 開始タグ<ROW>と終了タグ</ROW>のシーケンスを検出するたびに、前述の処理が繰り返されます。
5. 結果として、4 つのオブザベーションが生成されます。

完全な XMLMap 構文の内容は次のとおりです。

```
<?xml version="1.0" ?>
<SXLEMAP version="2.1" name="path" description="XMLMap for path">
 <TABLE name="FORD">
  <TABLE-PATH syntax="XPath"> /VEHICLES/FORD/ROW </TABLE-PATH>
  <COLUMN name="Model">
   <DATATYPE> string </DATATYPE>
   <LENGTH> 20 </LENGTH>
   <TYPE> character </TYPE>
   <PATH syntax="XPath"> /VEHICLES/FORD/ROW/Model </PATH>
  </COLUMN>
  <COLUMN name="Year">
   <DATATYPE> string </DATATYPE>
   <LENGTH> 4 </LENGTH>
```

```
    <TYPE> character </TYPE>
    <PATH syntax="XPath"> /VEHICLES/FORD/ROW/Year </PATH>
   </COLUMN>
  </TABLE>
 </SXLEMAP>
```

次の SAS ステートメントは、XML ドキュメントをインポートして、XMLMap を指定します。その後、PRINT プロシジャによりインポート結果を検証します。

```
filename path '/u/mydocuments/XML/path.xml';
filename map '/u/mydocuments/XML/path.map';

libname path xmlv2 xmlmap=map;

proc print data=path.ford noobs;
run;
```

アウトプット 5.7 PATH.FORD データセットを表示する PRINT プロシジャ出力

The SAS System

Model	Year
Mustang	1965
Explorer	1982
Taurus	1998
F150	2000

最適な列を選択するために、オブザベーションの境界を決定する

次の例では、XML ドキュメントをインポートする際に、最適な列の集合が生成されるようにオブザベーション境界を決定する方法を示します。

オブザベーション境界は、一定の個数の列を伴う任意の個数の行の集合(すなわち表)に変換されます。XMLMap を使用する場合、オブザベーション境界を決定するには、TABLE-PATH 要素で場所パスを指定します。

次の XML ドキュメントには、オブザベーション境界として使用できる可能性のある要素として PUBLICATION が含まれています。PUBLICATION をオブザベーション境界として使用すると、結果として 3 つの列 TITLE、ACQUIRED、TOPIC が生成されます。ただし、要素 TOPIC は単一の PUBLICATION コンテナ内で任意に出現するため、結果として TOPIC が複数回出現する一連の列が生成されることになります。このため、要素 PUBLICATION ではなく、要素 TOPIC をオブザベーション境界として使用する方がより適切な選択となります。この結果、4 つの列 TITLE、ACQUIRED、TOPIC、MAJOR が生成されます。

```
<?xml version="1.0" encoding="iso-8859-1" ?>
```

```
<Library>
 <Publication>
  <Title>Developer's Almanac</Title>
  <Acquired>12-11-2000</Acquired>
  <Topic Major="Y">JAVA</Topic>
 </Publication>
 <Publication>
  <Title>Inside Visual C++</Title>
  <Acquired>06-19-1998</Acquired>
  <Topic Major="Y">C</Topic>
  <Topic>Reference</Topic>
 </Publication>
 <Publication>
  <Title>Core Servlets</Title>
  <Acquired>05-30-2001</Acquired>
  <Topic Major="Y">JAVA</Topic>
  <Topic>Servlets</Topic>
  <Topic>Reference</Topic>
 </Publication>
</Library>
```

前述の XML ドキュメントをインポートする場合に使用する XMLMap 構文を次に示します。

```
<?xml version="1.0" ?>
<SXLEMAP version="1.2">
 <TABLE name="Publication">
  <TABLE-PATH syntax="XPath">
   /Library/Publication/Topic 1
  </TABLE-PATH>

  <COLUMN name="Title" retain="YES">
   <PATH>
    /Library/Publication/Title
   </PATH>
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>19</LENGTH>
  </COLUMN>

  <COLUMN name="Acquired" retain="YES">
   <PATH>
    /Library/Publication/Acquired
   </PATH>
   <TYPE>numeric</TYPE>
   <DATATYPE>FLOAT</DATATYPE>
   <LENGTH>10</LENGTH>
   <FORMAT width="10" >mmddyy</FORMAT> 2
   <INFORMAT width="10" >mmddyy</INFORMAT>
  </COLUMN>

  <COLUMN name="Topic">
   <PATH>
    /Library/Publication/Topic</PATH>
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>9</LENGTH>
```

```
  </COLUMN>

  <COLUMN name="Major">
   <PATH>
     /Library/Publication/Topic/@Major
   </PATH>
   <TYPE>character</TYPE>
   <DATATYPE>STRING</DATATYPE>
   <LENGTH>1</LENGTH>
   <ENUM> 3
    <VALUE>Y</VALUE>
    <VALUE>N</VALUE>
   </ENUM>
   <DEFAULT>N</DEFAULT> 4
  </COLUMN>
 </TABLE>
</SXLEMAP>
```

前述の XMLMap は、XML マークアップを次のように変換するよう XML Engine に伝えます。

1 要素 TOPIC は、SAS データセットの変数を XML ドキュメント内のどこで収集するかを定義する場所パスを指定します。この XML ドキュメント内で終了タグ</TOPIC>を検出するたびに、1 つのオブザベーションが出力されます。

2 列 ACQUIRED の場合、XMLMap 構文の FORMAT 要素を使用してデータが作成されます。FORMAT および INFORMAT のような要素は、SAS システムで使用するためにデータを変換しなければならない場合に便利です。XML Engine は、互いに独立して使用できるユーザー定義の出力形式と入力形式もサポートしています。

3 XMLMap 構文は列挙もサポートしています。この ENUM 要素は、列 MAJOR の値が Y または N のどちらかでなければならないことを指定します。入力された値が ENUM リスト内に含まれていない場合、その値は MISSING に設定されます。

4 デフォルトでは、欠損値は MISSING に設定されます。DEFAULT 要素は、欠損値のデフォルト値を指定します。この例では、欠損値のデフォルト値は N に指定されています。ENUM 要素を使用する場合、DEFAULT 要素に指定する値が有効となるためには、その値が ENUM 値の 1 つに指定されている必要があります。

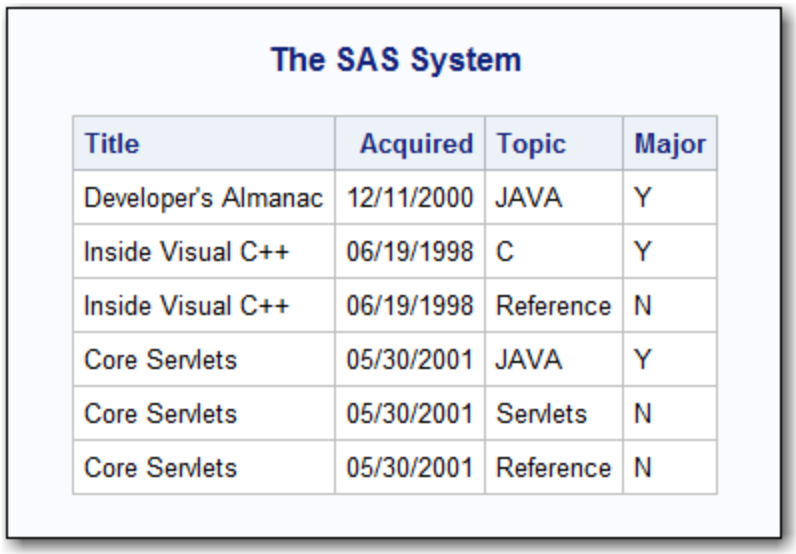
次の SAS ステートメントは、XML ドキュメントをインポートして、XMLMap を指定します。その後、PRINT プロシジャによりインポート結果を検証します。

```
filename rep '/u/mydocuments/XML/Rep.xml';
filename map '/u/mydocuments/XML/Rep.map';

libname rep xmlv2 xmlmap=map;

proc print data=rep.publication noobs;
run;
```

アウトプット 5.8 REP.PUBLICATION データセットを表示する PRINT プロシジャ出力



The SAS System

Title	Acquired	Topic	Major
Developer's Almanac	12/11/2000	JAVA	Y
Inside Visual C++	06/19/1998	C	Y
Inside Visual C++	06/19/1998	Reference	N
Core Servlets	05/30/2001	JAVA	Y
Core Servlets	05/30/2001	Servlets	N
Core Servlets	05/30/2001	Reference	N

ISO 8601 規格の SAS 入力形式と出力形式を使用し、日付をインポートする

次の例では、基本出力形式および拡張出力形式の両方で表した日付値を含んでいる XML ドキュメントをインポートします。この XMLMap では、FORMAT および INFORMAT の各要素を使用して、適切な SAS 出力形式および SAS 入力形式を指定することにより、ISO 8601 規格に従って日付を表示しています。

XML ドキュメントの内容は次のとおりです。

```
<?xml version="1.0" ?>
<Root>
 <ISODATE>
  <BASIC>20010911</BASIC>
  <EXTENDED>2001-09-11</EXTENDED>
 </ISODATE>
</Root>
```

次の XMLMap は、SAS 入力形式および SAS 出力形式を使用して XML ドキュメントをインポートし、日付値の読み込みと書き出しを行います。

```
<?xml version="1.0" encoding="UTF-8"?>
<SXLEMAP description="Reading a Basic and Extended format ISO date field"
  name="ISOdate" version="2.1">

  <NAMESPACES count="0"/>

  <!-- ############################################################ -->
  <TABLE name="ISODATE">
    <TABLE-PATH syntax="XPath">/Root/ISODATE</TABLE-PATH>

    <COLUMN name="BASIC">
```

```
    <PATH syntax="XPath">/Root/ISODATE/BASIC</PATH>
    <TYPE>numeric</TYPE>
    <DATATYPE>date</DATATYPE>
    <FORMAT width="10">E8601DA</FORMAT> 1
    <INFORMAT width="8">B8601DA</INFORMAT> 2
  </COLUMN>

  <COLUMN name="EXTENDED">
    <PATH syntax="XPath">/Root/ISODATE/EXTENDED</PATH>
    <TYPE>numeric</TYPE>
    <DATATYPE>date</DATATYPE>
    <FORMAT>E8601DA</FORMAT> 3
    <INFORMAT width="10">E8601DA</INFORMAT> 4
  </COLUMN>

 </TABLE>

</SXLEMAP>
```

日付値をインポートする XMLMap 構文についての説明を次に示します。

1 変数 Basic に対して、FORMAT 要素で SAS 出力形式 E8601DA を指定します。この出力形式は、日付値を拡張形式 *yyyy-mm-dd* で出力します。

2 変数 Basic に対して、INFORMAT 要素で SAS 入力形式 B8601DA を指定します。この入力形式は、基本形式 *yyyymmdd* を使って日付値を変数に読み込みます。

注: この例では、推奨に従って、基本 SAS 入力形式を使って変数に値を読み込んだ後、対応する拡張 SAS 出力形式を使って同変数の値を出力しています。

3 変数 Extended に対して、FORMAT 要素で SAS 出力形式 E8601DA を指定します。この出力形式は、日付値を拡張形式 *yyyy-mm-dd* で出力します。

4 変数 Extended に対して、INFORMAT 要素で SAS 入力形式 E8601DA を指定します。この入力形式は、基本形式 *yyyy-mm-dd* を使って日付値を変数に読み込みます。

次の SAS ステートメントは、XML ドキュメントをインポートした後、PRINT プロシジャ出力を表示します。

```
filename dates '/u/mydocuments/XML/isodate.xml';
filename map '/u/mydocuments/XML/isodate.map';

libname dates xmlv2 xmlmap=map;

proc print data=dates.isodate;
run;
```

アウトプット 5.9 インポートされたデータセット DATES.ISODATE の PRINT プロシジャ出力

The SAS System

Obs	BASIC	EXTENDED
1	2001-09-11	2001-09-11

ISO 8601 規格の SAS 入力形式と出力形式を使用し、タイムゾーン付きの時刻値をインポートする

次の例では、様々な形式の時刻値を含んでいる XML ドキュメントをインポートします。XMLMap では、FORMAT 要素および INFORMAT 要素を使用して、それぞれ対応する SAS 出力形式および SAS 入力形式を指定することにより、時刻が適切に表示されるようにします。

様々な形式の時刻値を含む XML ドキュメントの内容は次のとおりです。

```
<?xml version="1.0" ?>
<Root>
 <TIME>
  <LOCAL>09:00:00</LOCAL>
  <UTC>09:00:00Z</UTC>
  <OFFSET>14:00:00+05:00</OFFSET>
 </TIME>
</Root>
```

次の XMLMap は、SAS 入力形式および SAS 出力形式を使用して XML ドキュメントをインポートし、時刻値の読み込みと書き出しを行います。

```
<?xml version="1.0" encoding="UTF-8"?>
<SXLEMAP name="ISOtime" version="2.1">

  <NAMESPACES count="0"/>

  <!-- ############################################################ -->
  <TABLE name="TIME">
    <TABLE-PATH syntax="XPath">/Root/TIME</TABLE-PATH>

    <COLUMN name="LOCAL">
      <PATH syntax="XPath">/Root/TIME/LOCAL</PATH>
      <TYPE>numeric</TYPE>
      <DATATYPE>time</DATATYPE>
      <FORMAT width="8">E8601TM</FORMAT> 1
      <INFORMAT width="8">E8601TM</INFORMAT>
    </COLUMN>

    <COLUMN name="LOCALZONE">
```

```
    <PATH syntax="XPath">/Root/TIME/LOCAL</PATH>
    <TYPE>numeric</TYPE>
    <DATATYPE>time</DATATYPE>
    <FORMAT width="14">E8601LZ</FORMAT> 2
    <INFORMAT width="8">E8601TM</INFORMAT>
  </COLUMN>

  <COLUMN name="UTC">
    <PATH syntax="XPath">/Root/TIME/UTC</PATH>
    <TYPE>numeric</TYPE>
    <DATATYPE>time</DATATYPE>
    <FORMAT width="9">E8601TZ</FORMAT> 3
    <INFORMAT width="9">E8601TZ</INFORMAT>
  </COLUMN>

  <COLUMN name="OFFSET">
    <PATH syntax="XPath">/Root/TIME/OFFSET</PATH>
    <TYPE>numeric</TYPE>
    <DATATYPE>time</DATATYPE>
    <FORMAT width="14">E8601TZ</FORMAT> 4
    <INFORMAT width="14">E8601TZ</INFORMAT>
  </COLUMN>

 </TABLE>

</SXLEMAP>
```

時刻値をインポートする XMLMap 構文についての説明を次に示します。

1 変数 Local に対して、INFORMAT 要素および FORMAT 要素で、SAS 入力形式および SAS 出力形式として E8601TM を指定します。これにより、拡張形式 *hh:mm:ss.ffffff* で時刻値の読み込みと書き出しが行われます。タイムゾーンインジケータが存在しないため、値のコンテキストはローカルタイムとなります。

2 変数 Localzone (変数 Local と同じ値を読み込む変数)に対して、INFORMAT 要素で SAS 入力形式 E8601TM を指定します。これにより、拡張形式 *hh:mm:ss.ffffff* で時刻値が読み込まれます。タイムゾーンインジケータが存在しないため、値のコンテキストはローカルタイムとなります。

一方、FORMAT 要素では、SAS 出力形式 E8601LZ を指定します。これにより、拡張形式 *hh:mm:ss+|-hh:mm* で時刻値が出力されます。出力形式 E8601LZ は、現在のローカル SAS セッションによって決定される時刻値の末尾に UTC オフセットを追加します。出力形式 E8601LZ を使うことで、ローカルタイムの曖昧さをなくすような時刻表記を提供できます。

注: この時刻表記を使用する場合でも、時間に基づく値を混在させないことを推奨します。

3 変数 UTC に対して、INFORMAT 要素および FORMAT 要素で、SAS 入力形式および SAS 出力形式として E8601TZ を指定します。これにより、拡張形式 *hh:mm:ss+|-hh:mm* で時刻値の読み込みと書き出しが行われます。タイムゾーンインジケータが存在するため、値は UTC で表されるものと仮定されます。その値に対する調整または変換は行われません。

4 変数 Offset に対して、INFORMAT 要素および FORMAT 要素で、SAS 入力形式および SAS 出力形式として E8601TZ を指定します。これにより、拡張形式 *hh:mm:ss+|-hh:mm* で時刻値の読み込みと書き出しが行われます。タイムゾーンオフセットが存在するため、タイムゾーンを区別する SAS 入力形式

を使って変数に時刻値を読み込む場合、その値はタイムゾーンインジケータを通じて要求される UTC 時間に調整されます。ただし、タイムゾーンコンテキストはその値と一緒には保存されません。タイムゾーンを区別する SAS 出力形式を使って時刻値を書き出す場合、その値は、ゼロのオフセット値を伴う UTC 時間として表現され、ローカルタイムに対する調整は行われません。

次の SAS ステートメントは、XML ドキュメントをインポートした後、PRINT プロシジャ出力を表示します。

```
filename timzn '/u/mydocuments/XML/Time.xml';
filename map '/u/mydocuments/XML/Time.map';

libname timzn xmlv2 xmlmap=map;

proc print data=timzn.time;
run;
```

アウトプット 5.10 インポートされたデータセット TIMZN.TIME の PRINT プロシジャ出力

The SAS System

Obs	LOCAL	LOCALZONE	UTC	OFFSET
1	09:00:00	09:00:00-05:00	09:00:00Z	09:00:00+00:00

URL アクセス方式を使用してファイル参照名を参照する

XML Engine は、複数の方法を使用して、ファイル参照名により参照される XML ドキュメントにアクセスできます。URL アクセス方式を使用してファイル参照名を参照する場合にも、XMLMap を指定する必要があります。XMLMap を指定することにより、XML Engine が単一パスで XML ドキュメントを処理できるようになります。XMLMap を指定しない場合、XML Engine は二重パスで XML ドキュメントを処理します。

次の例では、URL アクセス方式を使用してファイル参照名を参照することにより、XML ドキュメントにアクセスする方法を示します。

```
filename nhl url 'http://www.a.com/Nhl.xml'; 1
filename map '/u/mydocuments/XML/Nhl.map'; 2

libname nhl xmlv2 xmlmap=map; 3

proc copy indd=nhl outdd=work; 4
  select nhl;
run;
```

1 最初の FILENAME ステートメントは、URL アクセス方式を使用して、ファイル参照名 NHL を XML ドキュメントに割り当てます。

2 2 番目の FILENAME ステートメントは、ファイル参照名 MAP を、NHL.map という名前の XMLMap ファイルの物理的な場所に割り当てます。

3 この LIBNAME ステートメントは、エンジンとして XML Engine を指定した上でファイル参照名 NHL を使用して XML ドキュメントを参照し、さらにファイル参照名 MAP を使用して XMLMap ファイルを参照しています。

4 COPY プロシジャで XML ドキュメントを読み込み、その内容を一時 SAS データセットとして出力します。URL アクセス方式を使用する場合、COPY プロシジャか DATA ステップのどちらかを使用して SAS データセットを作成するステップを含める必要があります。

PATH 要素に場所パスを指定する

XMLMap の PATH 要素は、場所パスを指定するための XPath 形式を複数サポートしています。場所パスは、現在の変数の値を取り出すには、XML ドキュメント内のどの位置を検索し、特定のタグにアクセスする必要があるかを XML Engine に伝えます。また、場所パスは、XPath 形式により指定される機能を実行して変数の値を取り出すよう XML Engine に伝えます。

次の例では、XML ドキュメントをインポートする際に、サポートされている各種の XPath 形式を指定する方法を示します。これには、3 つの要素形式と 2 つの属性形式が含まれます。

インポートする XML ドキュメント NHL.XML の内容は次のとおりです。

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<NHL>
 <CONFERENCE> Eastern
  <DIVISION> Southeast
   <TEAM founded="1999" abbrev="ATL"> Thrashers </TEAM>
   <TEAM founded="1997" abbrev="CAR"> Hurricanes </TEAM>
   <TEAM founded="1993" abbrev="FLA"> Panthers </TEAM>
   <TEAM founded="1992" abbrev="TB" > Lightning </TEAM>
   <TEAM founded="1974" abbrev="WSH"> Capitals </TEAM>
  </DIVISION>
 </CONFERENCE>
</NHL>
```

この XML ドキュメントをインポートするのに使用される XMLMap の内容は次のようになります。また、PATH 要素内の各 XPath 形式に関する説明を次に示します。

```
<?xml version="1.0" ?>
<SXLEMAP version="2.1">
 <TABLE name="TEAMS">
     <TABLE-PATH syntax="XPath">
       /NHL/CONFERENCE/DIVISION/TEAM
      </TABLE-PATH>

     <COLUMN name="ABBREV">
      <PATH syntax="XPath">
       /NHL/CONFERENCE/DIVISION/TEAM/@abbrev 1
       </PATH>
       <TYPE>character</TYPE>
       <DATATYPE>STRING</DATATYPE>
```

```
    <LENGTH>3</LENGTH>
  </COLUMN>

  <COLUMN name="FOUNDED">
    <PATH syntax="XPath">
      /NHL/CONFERENCE/DIVISION/TEAM/@founded[@abbrev="ATL"] 2
    </PATH>
    <TYPE>character</TYPE>
    <DATATYPE>STRING</DATATYPE>
    <LENGTH>10</LENGTH>
  </COLUMN>

 <COLUMN name="CONFERENCE" retain="YES">
    <PATH syntax="XPath">
      /NHL/CONFERENCE 3
    </PATH>
    <TYPE>character</TYPE>
    <DATATYPE>STRING</DATATYPE>
    <LENGTH>10</LENGTH>
  </COLUMN>

  <COLUMN name="TEAM">
    <PATH syntax="XPath">
      /NHL/CONFERENCE/DIVISION/TEAM[@founded="1993"] 4
    </PATH>
    <TYPE>character</TYPE>
    <DATATYPE>STRING</DATATYPE>
    <LENGTH>10</LENGTH>
  </COLUMN>

  <COLUMN name="TEAM5">
    <PATH syntax="XPath">
      /NHL/CONFERENCE/DIVISION/TEAM[position()=5] 5
     </PATH>
    <TYPE>character</TYPE>
    <DATATYPE>STRING</DATATYPE>
    <LENGTH>10</LENGTH>
  </COLUMN>

 </TABLE>
</SXLEMAP>
```

1 変数 Abbrev は、特定の属性から値を選択する属性形式を使用します。XML Engine は、TEAM 要素を検出するまで XML マークアップをスキャンします。続いて、XML Engine は、abbrev=属性から値を取り出します。取り出された結果は各チームの略称となります。

2 変数 Founded は、別の属性の値に基づいて特定の属性から条件付きで選択を行う属性形式を使用します。XML Engine は、TEAM 要素を検出するまで XML マークアップをスキャンします。続いて、XML Engine は、abbrev=属性の値が ATL である場合に founded=属性から値を取り出します。取り出された値は 1999 になります。これら 2 つの属性は、同じ要素に対して指定されたものでなければなりません。

3 変数 Conference は、指定の要素から PCDATA を選択する要素形式を使用します。XML Engine は、CONFERENCE 要素を検出するまで XML マークアップをスキャンします。続いて、XML Engine は、開始タグ<CONFERENCE>と

終了タグ</CONFERENCE>の間に記述されている値を取り出します。取り出された値は Eastern になります。

4 変数 Team は、指定の要素から条件付きで PCDATA を選択する要素形式を使用します。XML Engine は、founded=属性の値が 1993 である TEAM 要素を検出するまで XML マークアップをスキャンします。続いて、XML Engine は、開始タグ<TEAM>と終了タグ</TEAM>の間に記述されている値を取り出します。取り出された値は Panthers になります。

5 変数 Team5 は、指定の要素の特定のオカレンスに基づいて、その要素から条件付きで PCDATA を選択する要素形式を使用します。位置関数は、TEAM 要素の 5 番目のオカレンスを検出するまで XML マークアップをスキャンするよう XML Engine に伝えます。続いて、XML Engine は、開始タグ<TEAM>と終了タグ</TEAM>の間に記述されている値を取り出します。取り出された値は Capitals になります。

次の SAS ステートメントでは、Nhl1.map という名前の XMLMap ファイルを指定した上で、XML ドキュメント NHLShort.xml をインポートします。続いて、PRINT プロシジャにより、結果として生成された変数とその選択された値を表示します。

```
filename nhl '/u/mydocuments/XML/NhlShort.xml';
filename map '/u/mydocuments/XML/Nhl1.map';

libname nhl xmlv2 xmlmap=map;

proc print data=nhl.teams noobs;
run;
```

アウトプット 5.11 結果として生成された変数とその選択された値を表示する PRINT プロシジャ出力

The SAS System

ABBREV	FOUNDED	CONFERENCE	TEAM	TEAM5
ATL	1999	Eastern		
CAR		Eastern		
FLA		Eastern	Panthers	
TB		Eastern		
WSH		Eastern		Capitals

XMLMap に名前空間要素を挿入する

次の例では、XMLMap 名前空間要素の使い方を示します。XMLMap 名前空間要素を使用すると、XML 名前空間により修飾される同じ名前の要素を含む XML ドキュメントをインポートできるようになります。XMLMap 名前空間要素を使う

と、インポートされた XML ドキュメント内に含まれている XML 名前空間を維持することや、SAS データセットから名前空間を含む XML ドキュメントをエクスポートすることが可能となります。

インポートする XML ドキュメント NSSample.xml の内容は次のとおりです。この XML ドキュメントには 3 つの XML 名前空間が含まれています。名前空間は、一意の URI への参照を使用して ADDRESS 要素を修飾することにより、同要素を区別します。ADDRESS 要素は、次に示す最初の PERSON 反復要素内で強調表示されています。

```
<?xml version="1.0" encoding="UTF-8"?>
<PEOPLE xmlns:HOME="http://sample.url.org/home"
  xmlns:IP="http://sample.url.org/ip"
  xmlns:WORK="http://sample.url.org/work">
   <PERSON>
      <NAME>Joe Smith</NAME>
      <HOME:ADDRESS>1234 Elm Street</HOME:ADDRESS>
      <HOME:PHONE>999-555-0011</HOME:PHONE>
      <WORK:ADDRESS>2001 Office Drive, Box 101</WORK:ADDRESS>
      <WORK:PHONE>999-555-0101</WORK:PHONE>
      <IP:ADDRESS>192.168.1.1</IP:ADDRESS>
   </PERSON>
   <PERSON>
      <NAME>Jane Jones</NAME>
      <HOME:ADDRESS>9876 Main Street</HOME:ADDRESS>
      <HOME:PHONE>999-555-0022</HOME:PHONE>
      <WORK:ADDRESS>2001 Office Drive, Box 102</WORK:ADDRESS>
      <WORK:PHONE>999-555-0102</WORK:PHONE>
      <IP:ADDRESS>172.16.1.2</IP:ADDRESS>
   </PERSON>
   <PERSON>
      <NAME>Pat Perkinson</NAME>
      <HOME:ADDRESS>1395 Half Way</HOME:ADDRESS>
      <HOME:PHONE>999-555-0033</HOME:PHONE>
      <WORK:ADDRESS>2001 Office Drive, Box 103</WORK:ADDRESS>
      <WORK:PHONE>999-555-0103</WORK:PHONE>
      <IP:ADDRESS>10.0.1.3</IP:ADDRESS>
   </PERSON>
</PEOPLE>
```

この XML ドキュメントをインポートするのに使用した XMLMap の内容を次に示します。名前空間要素に関する説明も示します。

```
<SXLEMAP name="Namespace" version="2.1">

   <NAMESPACES count="3"> 1
      <NS id="1" prefix="HOME">http://sample.url.org/home</NS> 2
      <NS id="2" prefix="IP">http://sample.url.org/ip</NS>
      <NS id="3" prefix="WORK">http://sample.url.org/work</NS>
   </NAMESPACES>

   <TABLE description="PERSON" name="PERSON"> 3
      <TABLE-PATH syntax="XPath">/PEOPLE/PERSON</TABLE-PATH>

      <COLUMN name="NAME"> 4
         <PATH syntax="XPath">/PEOPLE/PERSON/NAME</PATH>
         <TYPE>character</TYPE>
         <DATATYPE>string</DATATYPE>
```

```
    <LENGTH>13</LENGTH>
  </COLUMN>

  <COLUMN name="ADDRESS"> 4
    <PATH syntax="XPathENR">/PEOPLE/PERSON/{1}ADDRESS</PATH> 5
    <TYPE>character</TYPE>
    <DATATYPE>string</DATATYPE>
    <LENGTH>16</LENGTH>
  </COLUMN>

  <COLUMN name="PHONE"> 4
    <PATH syntax="XPathENR">/PEOPLE/PERSON/{1}PHONE</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>string</DATATYPE>
    <LENGTH>12</LENGTH>
  </COLUMN>

  <COLUMN name="ADDRESS1"> 4
    <PATH syntax="XPathENR">/PEOPLE/PERSON/{3}ADDRESS</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>string</DATATYPE>
    <LENGTH>26</LENGTH>
  </COLUMN>

  <COLUMN name="PHONE1"> 4
    <PATH syntax="XPathENR">/PEOPLE/PERSON/{3}PHONE</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>string</DATATYPE>
    <LENGTH>12</LENGTH>
  </COLUMN>

  <COLUMN name="ADDRESS2"> 4
    <PATH syntax="XPathENR">/PEOPLE/PERSON/{2}ADDRESS</PATH>
    <TYPE>character</TYPE>
    <DATATYPE>string</DATATYPE>
    <LENGTH>11</LENGTH>
  </COLUMN>

 </TABLE>

</SXLEMAP>
```

1. NAMESPACES 要素には、XML 名前空間を定義するための NS 要素が含まれています。count=属性で、3 つの定義済み XML 名前空間が存在することを指定します。
2. 3 つの NS 要素は、一意の URI を参照することにより XML 名前空間を定義します。id=属性で、3 つの XML 名前空間の ID 番号として 1、2、3 を指定します。prefix=属性で、HOME、WORK、IP の各名前を、参照される URI に割り当てます。
3. この XMLMap TABLE 要素は、PERSON 反復要素のデータセット定義を含んでいます。
4. この XMLMap COLUMN 要素は、PERSON 内部にあるネストされている各要素の変数定義を含んでいます。これには、NAME、ADDRESS、PHONE、ADDRESS1、PHONE1、ADDRESS2 が含まれます。

5. PATH 要素内では、各 COLUMN 要素に対して、構文型を XPathENR (埋め込み型の名前空間参照を持つ XPath)として指定します。この構文型は、構文が XPath 規格に準拠していないことを示します。また、定義される要素に先立って、ID 番号を場所パス内に含めます。ID 番号は中かっこで囲みます。たとえば、ADDRESS 要素の PATH 要素は次のようになります。**<PATH syntax="XPathENR">/PEOPLE/PERSON/{1}ADDRESS</PATH>**.

次の SAS ステートメントは、XML ドキュメントをインポートして、NSSample.map という名前の XMLMap を指定します。続く PRINT プロシジャで、結果として生成される SAS データセットを表示します。

```
filename ns '/u/mydocuments/XML/NSSample.xml';
filename nsmap '/u/mydocuments/XML/NSSample.map';

libname ns xmlv2 xmlmap=nsmap;

proc print data=ns.person noobs;
run;
```

アウトプット 5.12 NS.PERSON の PRINT プロシジャ出力

The SAS System

NAME	HOME_ADDRESS	HOME_PHONE	WORK_ADDRESS	WORK_PHONE	IP_ADDRESS
Joe Smith	1234 Elm Street	999-555-0011	2001 Office Drive, Box 101	999-555-0101	192.168.1.1
Jane Jones	9876 Main Street	999-555-0022	2001 Office Drive, Box 102	999-555-0102	172.16.1.1
Pat Perkinson	1395 Half Way	999-555-0033	2001 Office Drive, Box 103	999-555-0103	10.0.1.1

6 章

XML Engine のタグセットの説明と使い方

タグセットについて

タグセットは、SAS データセットからマークアップ言語を生成するための命令を指定します。結果として生成される出力には、レイアウトや一部のコンテンツを定義する組み込み型の命令が含まれています。SAS システムには、XML などの各種マークアップ言語用のタグセットが用意されています。

カスタマイズしたタグセットの作成

SAS システムが提供するタグセットを使用する以外に、ユーザーはそのような SAS タグセットを変更することや、ユーザー独自のタグセットを作成することができます。タグセットを作成するには、TEMPLATE プロシジャを使用してタグセットの定義を作成する必要があります。

注意:

カスタマイズしたタグセットを使用する場合には注意が必要です。 XML 出力に慣れていない場合は、異なるタグセットを指定しないでください。XML ドキュメントをエクスポートする際にタグセットを変更した後、その変更されたタグセットにより生成された XML ドキュメントをインポートした場合、XML Engine はその XML マークアップを SAS システムに独自のフォーマットに戻せないことがあります。

カスタマイズしたタグセットを使用し、XML ドキュメントをエクスポートする

例の概要

次の例では、カスタマイズしたタグセットを定義した後、そのタグセットを XML Engine と共に使用することにより、カスタマイズされたタグを含む XML ドキュメントをエクスポートしています。

TEMPLATE プロシジャによるカスタマイズしたタグセットの定義

次の TEMPLATE プロシジャは、Tagsets.Custom というカスタマイズしたタグセットを定義します。

次のプログラムをテンプレートとして使用することで、ユーザー独自のカスタマイズしたタグセットを定義できます。たとえば、EmitMeta、EmitRow、EmitCol の各イベントを少し変更するだけで、ユーザー独自のカスタマイズしたタグセットを作成できます。

```
proc template;

 /* +-------------------------------------------------+
   |                                                 |
   +-------------------------------------------------+ */

 define tagset tagsets.custom ;
   notes "SAS XML Engine output event model(interface)";

   indent = 3;
   map = '<>&"';
   mapsub = '/&lt;/&gt;/&amp;/&quot;/&apos;/';


/* +-------------------------------------------------+
   |                                                 |
   +-------------------------------------------------+ */

   define event XMLversion;
     put  '<?xml version="1.0"';
     putq ' encoding=' ENCODING;
     put  ' ?>' CR;
     break;
   end;


   define event XMLcomment;
     put  '<!-- ' CR;
     put  '    ' TEXT CR;
     put  '  -->' CR;
     break;
   end;
```

```
  define event initialize;
  set   $LIBRARYNAME        'LIBRARY' ;
  set   $TABLENAME          'DATASET' ;
  set   $COLTAG             'column' ;
  set   $META               'FULL' ;

  eval  $is_engine          1;
  eval  $is_procprint       0;
  eval  $is_OUTBOARD        1;
  end;


/* +-------------------------------------------------+
   |                                                 |
   +-------------------------------------------------+ */

  define event doc;
  start:
    trigger initialize;
    trigger XMLversion;
    break;
  finish:
    break;
  end;


  define event doc_head;
  start:
    break;
  finish:
    break;
  end;


  define event doc_body;
  start:
    break;
  finish:
    break;
  end;


  define event proc;
  start:
    break / if frame_name ;            /* set by ODS statement use  */
    eval $is_OUTBOARD 0 ;              /* default for non-engine    */
    do / if cmp(XMLCONTROL, "OUTBOARD"); /* only the engine sets this */
      eval $is_OUTBOARD 1 ;
    else ;
      eval $is_OUTBOARD 0 ;
    done ;
    break;
  finish:
    break;
  end;
```

```
define event leaf;
start:
  /*
   *  PROC PRINT
   *  data set reference is in the value and label fields
   *  and NOT in the output_label field
   */
  eval $is_engine     0; /* NOT ENGINE */
  break / if ^cmp("Print", name);
  eval $is_procprint  1; /* PROC PRINT */
  eval $regex prxparse("/\.(.+)/");
  eval $match prxmatch($regex, value);
  set  $TABLENAME prxposn($regex, 1, value);
  break;
finish:
  break;
end;


define event output;
start:
  break / if $is_procprint ;
  eval  $is_engine     0;          /* NOT ENGINE   */
  set   $TABLENAME name / if name;   /* TABLE VIEWER */
  break;
finish:
  break;
end;


define event table;
start:
  unset $col_names;
  unset $col_types;
  unset $col_width;
  eval  $index      1;
  eval  $index_max  0;
  set   $TABLENAME name / if name;          /* LIBNAME ENGINE */
  set   $META XMLMETADATA / if XMLMETADATA ; /* LIBNAME ENGINE */
  set   $SCHEMA XMLSCHEMA / if XMLSCHEMA ;   /* LIBNAME ENGINE */
  break;
finish:
  break;
end;


define event colspecs;
start:
  break / if cmp(XMLMETADATA, "NONE");
finish:
  break / if cmp(XMLMETADATA, "NONE");
end;
```

```
  define event colgroup;
  start:
    break / if cmp(XMLMETADATA, "NONE");
  finish:
    break / if cmp(XMLMETADATA, "NONE");
  end;


/* +-------------------------------------------------+
   |                                                 |
   +-------------------------------------------------+ */

  define event colspec_entry;
  start:
    break / if ^$is_engine and $index eq 1 and cmp(name, "Obs");
    eval  $index_max $index_max+1;
    set $col_names[] name;
    set $col_types[] type;
    set $col_width[] width;
    break;
  finish:
    break;
  end;


  define event table_head;
  start:
    break;
  finish:
    break;
  end;


  define event table_body;
  start:
    trigger EmitMeta ;
    break;
  finish:
    trigger EmitMeta ;
    break;
  end;

/* +-------------------------------------------------+
   |                                                 |
   +-------------------------------------------------+ */

  define event row;
  start:
    break / if !cmp(SECTION, "body");
    break / if cmp(XMLMETADATA, "ONLY");
    eval    $index 1;
    unset   $col_values;
    break;
  finish:
    break / if !cmp(SECTION, "body");
    break / if cmp(XMLMETADATA, "ONLY");
```

```
    trigger EmitRow ;
    break;
  end;


  define event data;
  start:
    break / if !cmp(SECTION, "body");
    do / if $is_engine ;
      break / if !cmp(XMLCONTROL, "Data");
    else ;
      break / if !cmp(HTMLCLASS, "Data");
    done ;
    break / if cmp(XMLMETADATA, "ONLY");
    set $name $col_names[$index];
    do / if exists(MISSING);
      eval  $is_MISSING  1;
      eval  $value_MISSING MISSING;
      set   $col_values[$name] " ";
    else ;
      eval  $is_MISSING  0;
      set   $col_values[$name] VALUE;
    done;
    break;
  finish:
    break / if !cmp(SECTION, "body");
    do / if $is_engine ;
      break / if !cmp(XMLCONTROL, "Data");
    else ;
      break / if !cmp(HTMLCLASS, "Data");
    done ;
    break / if cmp(XMLMETADATA, "ONLY");
    set  $name  $col_names[$index];
    eval $index $index+1;
    break;
  end;


/* +-----------------------------------------------+
   |                                               |
   | at this point, we just take over XML output.  |
   | EmitRow() is triggered each time the data is  |
   |          loaded into the $col_values array.   |
   |                                               |
   | we can output anything we desire from here... |
   |                                               |
   +-----------------------------------------------+ */

   define event EmitMeta; 1
   start:
     put '<' $LIBRARYNAME '>' CR ;
     put '   <!-- ' CR ;
     put '       List of available columns' CR  ;

     eval $index  1;
     iterate $col_names ;
```

```
    do /while _value_;
      put '          ' $index ' ' _value_ CR  ;
      next $col_names;
      eval $index $index+1;
      done;
    put '   -->' CR ;
    break;
  finish:
    put '</' $LIBRARYNAME '>' ;
    break;
  end;


  define event EmitRow; 2
    ndent;
    put "<STUDENT>" CR ;
    ndent;

    set $name  "Name";   trigger EmitCol ;
    set $name  "Height"; trigger EmitCol ;
    set $name  "Weight"; trigger EmitCol ;

    xdent;
    put "</STUDENT>" CR ;
    xdent;
    break;
  end;


  define event EmitCol; 3
    unset $value;
    set $value $col_values[$name];
    put '<'  $name '>' ;
    put      $value ;
    put '</' $name '>' CR ;
    break;
  end;

 end; /* custom */
run;
```

1 EmitMeta イベントは、SAS データセット内にある変数のリストを含む XML コメントを生成します。このイベントには、リスト変数を反復して使用することで SAS データセット内のすべての変数を処理する例が含まれています。反復処理に関する詳細は、TEMPLATE プロシジャの DEFINE EVENT ステートメントにおける ITERATE ステートメントの説明を参照してください。

2 EmitRow イベントは、SAS データセットの 3 つのオブザベーションから XML 出力を生成します。EmitRow イベントは、処理対象の具体的な変数(Name、Height、Weight)を指定します。

3 EmitCol イベントは、処理対象となる変数ごとに、汎用的な見かけの XML を生成します。

カスタマイズしたタグセットを使用した XML ドキュメントのエクスポート

次の SAS プログラムは、カスタマイズしたタグセットを使用して、SAS データセットを XML ドキュメントとしてエクスポートします。

```
data work.class; 1
  set sashelp.class (obs=3);
run;

filename xmlout "/u/mydocuments/XML/engine92.xml"; 2

libname xmlout xmlv2 xmltype=GENERIC tagset=tagsets.custom; 3

data xmlout.class; 4
  set work.class;
run;
```

1 この DATA ステップは、3 つのオブザベーションのみを含むデータセット WORK.CLASS を作成します。

2 この FILENAME ステートメントは、ファイル参照名 XMLOUT を、エクスポートされた XML ドキュメントが格納されるファイルの物理的な場所(完全なパス名、ファイル名、ファイル拡張子を含むもの)に割り当てます。

3 この LIBNAME ステートメントは、このファイル参照名を使用して XML ドキュメントを参照し、エンジンとして XML Engine を指定します。TAGSET=オプションでは、カスタマイズしたタグセット Tagsets.Custom を指定します。

4 この DATA ステップは、データセット WORK.CLASS を読み込み、カスタマイズしたタグセットにより定義されている形式で、読み込んだ内容を指定の XML ドキュメントに出力します。

結果として生成される XML ドキュメントの内容は次のようになります。

アウトプット 6.1 カスタマイズしたタグセットを使用した XML ドキュメントのエクスポート

```
<?xml version="1.0" encoding="windows-1252" ?> <LIBRARY> <!-- List of available
columns 1 Name 2 Sex 3 Age 4 Height 5 Weight --> <STUDENT> <Name>Alfred</Name>
<Height>69</Height> <Weight>112.5</Weight> </STUDENT> <STUDENT> <Name>Alice</
Name> <Height>56.5</Height> <Weight>84</Weight> </STUDENT> <STUDENT>
<Name>Barbara</Name> <Height>65.3</Height> <Weight>98</Weight> </STUDENT> </
LIBRARY>
```

2 部

LIBNAME ステートメントの参照

7 章
LIBNAME ステートメント: 概要

LIBNAME ステートメントの使用

XML ドキュメントのインポートやエクスポートを行うには、LIBNAME ステートメントでエンジンとして XML Engine を指定した上で、SAS ライブラリ参照名を、XML ドキュメントを格納している SAS ライブラリまたは特定の XML ドキュメントに関連付けます。

基本的な例に関しては、3 章, “XML ドキュメントのインポート” (15 ページ)および 2 章, “XML ドキュメントのエクスポート” (9 ページ)を参照してください。

XML LIBNAME Engine について

XML Engine について

エンジンニックネーム **XMLV2** を指定すると、SAS 9.1.3 以降の拡張点や変更点を含む XML Engine 機能にアクセスできます。ニックネーム XMLV2 を指定した場合、LIBNAME ステートメントの拡張機能、新しい XMLMap 機能、廃止された構文の診断機能などが利用できます。

- XMLV2 バージョンは XML 準拠です。
- XMLV2 向けの LIBNAME ステートメント機能には、XMLMAP マークアップタイプ、追加オプション、ライブラリ参照名を SAS ライブラリに割り当てる機能などが含まれています。
- XMLV2 向けの XMLMap 機能には、XMLMap を使用したエクスポートや、XML 名前空間のサポートなどが含まれています。

XMLV2 バージョンと以前のバージョンの比較

XML コンプライアンス

XMLV2 バージョンは XML 準拠です。したがって XML マークアップは、W3C 規格に準拠した、整形式で有効な構成である必要があります。XMLV2 バージョンは XML 準拠であるため、XMLV2 の使用は次のような場合に影響を与えます。

- XMLV2 バージョンを使用してインポートした XML ドキュメントは、より厳密なパーシング規則に合格しなければなりません。たとえば、XML マークアップに関して、XMLV2 バージョンでは大文字小文字を区別します。開始タグと終了タグでは、**<BODY> ...</BODY>**や**<Message>...</Message>**のように、大文字小文字が一致している必要があります。XMLV2 バージョンでは、タグ**<Letter>**とタグ**<letter>**は別のタグとして認識されます。属性名も大文字小文字が区別されます。また、属性値は、**<Note date="09/24/1975">**のように引用符で囲む必要があります。
- XMLV2 バージョンは、XMLMap ファイルが XML 準拠であることを必要とします。すなわち、XMLMap ファイルにおいてマークアップの大文字小文字が区別されます。また、XMLMap マークアップは XMLMap に固有の規則に従う必要があります。タグ名は大文字でなければなりません。要素の属性は小文字で記述する必要があります。たとえば、**<SXLEMAP version="2.1">**のように記述します。同様に、サポートされている XPath 構文でも大文字小文字が区別されます。

XMLMap ファイル

XMLV2 バージョンは、XMLMap バージョン 1.2 で始まる XMLMap ファイルをサポートします。文書化されている MLMap 構文のバージョンは 2.1 です。9 章, “XMLMap 構文: 概要” (91 ページ)を参照してください。

XMLV2 向けの LIBNAME ステートメントの機能

XMLV2 バージョンには次のような LIBNAME ステートメント機能があります。

- 特定の XML ドキュメントにライブラリ参照名を割り当てるのではなく、SAS ライブラリにライブラリ参照名を割り当てる機能。
- XMLMAP マークアップタイプ。
- 追加オプション。XMLV2 のニックネームで利用可能な LIBNAME ステートメントオプションについては、“LIBNAME ステートメントオプション” (75 ページ)を参照してください。
- XMLV2 ニックネームと GENERIC マークアップタイプを組み合わせて使用すると、複数の SAS データセットから 1 つの XML ドキュメントをエクスポートできます。たとえば、Grades.Fred および Grades.Wilma という名前の 2 つの SAS データセットがある場合、次のプログラムは、これら両方の SAS データセットに含まれている成績データを含む XML ドキュメント Grades.xml をエクスポートします。

```
libname stones xmlv2 'c:\Grades.xml';

data stones.fred;
  set grades.fred;
run;

data stones.wilma;
```

```
  set grades.wilma;
run;
```

LIBNAME ステートメントオプション

利用可能な LIBNAME ステートメントオプションの一覧を次の表に示します。

表 7.1 LIBNAME ステートメントオプション

タスク	オプション
エクスポートされた XML ドキュメント内のネストされている要素をインデントします	INDENT= (80 ページ)
出力ファイルで使用する文字セットを指定します	ODSCHARSET= (80 ページ)
出力ファイルで使用する変換テーブルを指定します	ODSTRANTAB= (80 ページ)
デフォルトのタグセットをオーバーライドします	TAGSET= (81 ページ)
連結された XML ドキュメントをインポートします	XMLCONCATENATE= (81 ページ)
SAS 変数の情報を含むタグ形式を指定します	XMLDATAFORM= (82 ページ)
数値の結果を制御します	XMLDOUBLE= (82 ページ)
出力ファイルの SAS データセットのエンコーディングをオーバーライドします	XMLENCODING= (83 ページ)
XML ドキュメントのファイル参照名を指定します	XMLFILEREF= (83 ページ)
XMLMap を指定します	XMLMAP= (84 ページ)
メタデータ関連の情報を含めるかどうかを指定します	XMLMETA= (84 ページ)
W3C 規格に従っていない文字データを処理するかどうかを指定します	XMLPROCESS= (85 ページ)
エクスポートされたメタデータ関連の情報を含める外部ファイルを指定します	XMLSCHEMA= (86 ページ)
XML マークアップタイプを指定します	XMLTYPE= (86 ページ)

8 章

LIBNAME ステートメントの構文

ディクショナリ

LIBNAME ステートメントの構文

XML ドキュメントを処理します。

該当要素: 任意の場所

カテゴリ: データアクセス

構文

LIBNAME *libref engine 'SAS-library | XML-document-path' <option(s)>*;

オプション引数の要約

INDENT=*integer*
: エクスポートされた XML ドキュメント内にあるネストされている各要素をインデントする場合の列数を指定します。

ODSCHARSET=*character-set*
: 出力ファイルで使用する文字セットを指定します。

ODSTRANTAB=*table-name*
: 出力ファイルで使用する変換テーブルを指定します。

TAGSET=*tagset-name*
: XMLTYPE=に指定されたマークアップタイプにより使用されるデフォルトのタグセットをオーバーライドするタグセット名を指定します。

XMLCONCATENATE=NO | YES
: インポートするファイルに、複数の連結された XML ドキュメントを含めるかどうかを指定します。

XMLDATAFORM=ELEMENT | ATTRIBUTE

SAS 変数情報(名前とデータ)を含める要素のタグが、開かれた要素形式であるか、囲まれた属性形式であるかを示します。

XMLDOUBLE=DISPLAY | INTERNAL
数値のインポートまたはエクスポート結果を制御します。

XMLENCODING='*encoding-value*'
出力ファイルの SAS データセットのエンコーディングをオーバーライドします。

XMLFILEREF=*fileref*
エクスポートまたはインポートの対象となる XML ドキュメントの物理的な場所に関連付ける SAS 名を指定します。

XMLMAP=*fileref* | '*XMLMap*'
特定の XMLMap 構文を含む、ユーザーが作成した XML ドキュメントを指定します。

XMLMETA=DATA | SCHEMADATA | SCHEMA
エクスポートするマークアップ内にメタデータ関連情報を含めるかどうかを指定します。または、入力される XML ドキュメント内に含まれているメタデータ関連情報をインポートするかどうかを指定します。

XMLPROCESS=CONFORM | PERMIT
XML Engine が W3C 規格に従っていない文字データをどのように処理するかを指定します。

XMLSCHEMA=*fileref* | '*external-file*'
メタデータ関連の情報を含める外部ファイルを指定します。

XMLTYPE=GENERIC | XMLMAP
XML マークアップタイプを指定します。

必須引数

libref
XML ドキュメントの物理的な場所に関連付ける短縮名として機能する有効な SAS 名を指定します。この名前は、SAS 名の規則に従っている必要があります。ライブラリ参照名の最大長は 8 文字です。

engine
XML ドキュメントのインポートやエクスポートを行う SAS XML LIBNAME Engine のニックネームを指定します。

XMLV2
SAS 9.2 以降の XML Engine 機能にアクセスする XML Engine のニックネームを指定します。

別名 XML92

制限事項 XMLV2 Engine を使用して XML ドキュメントをエクスポートする場合、エクスポートする SAS データセットは最大 19 個まで指定できます。

ヒント システム管理者が XML LIBNAME Engine に別名を割り当てている場合、そのサイトではエンジンニックネームが異なる場合があります。別名が割り当てられているかどうかについては、各サイトのシステム管理者にお問い合わせください。

'***SAS-library* | *XML-document-path***'
エクスポートまたはインポートの対象となる XML ドキュメントの物理的な場所を指定します。物理的な場所は、一重または二重引用符で囲みます。

SAS-library
SAS システムにより認識され、1 つの単位として参照および保存される、1 つ以上のファイルからなる集合体のパス名を指定します。例: **'/u/mydocuments/XML'**。

XML-document-path
パス名、ファイル名、ファイル拡張子を指定します。例: **'/u/mydocuments/XML/myfile.xml'**。

動作環境の情報
ファイルの物理的な場所の指定に関する詳細は、お使いの動作環境向けの SAS ドキュメントを参照してください。

操作 FILENAME ステートメントを使うと、エクスポートまたはインポートの対象となる XML ドキュメントの物理的な場所に対してファイル参照名を割り当てることができます。ファイル参照名とライブラリ参照名が一致する場合、LIBNAME ステートメント内で XML ドキュメントの物理的な場所を指定する必要はありません。たとえば、次のプログラムは、XML ドキュメント Fred.XML の内容を出力します。

```
filename bedrock '/u/xmldata/fred.xml';

libname bedrock xml;

proc print data=bedrock.fred;
run;
```

ライブラリ参照名に一致しない XML ドキュメントのファイル参照名を指定するには、 XMLFILEREF= オプション (83 ページ)を使用します。たとえば、次のプログラムは、XML ドキュメント Wilma.XML の内容を出力します。

```
filename cartoon '/u/xmldata/wilma.xml';

libname bedrock xml xmlfileref=cartoon;

proc print data=bedrock.wilma;
run;
```

オプション引数

GENERIC マークアップタイプの場合、出力される値が SAS 出力形式による影響を受けるかどうかを指定します。

NO
実際のデータ値を XML マークアップに出力します。

YES
フォーマットされたデータ値を XML マークアップに含めるようにします。

制限事項 GENERIC マークアップタイプの場合、フォーマットされたデータ値を含んでいる SAS データセットをエクスポートした後、生成された XML ドキュメントをインポートして元の SAS データセットへ戻そうとすると、そのインポート操作は失敗します。フォーマットされたデータ値を含んでいる SAS データセットをエクスポートすると、

その結果として、異なる変数や異なる変数属性が生成される場合があります。

デフォルト NO

制限事項 このオプションは、GENERIC マークアップタイプに対してのみ使用します

INDENT=*integer*
エクスポートされた XML ドキュメント内にあるネストされている各要素をインデントする場合の列数を指定します。 この値は、0 (インデントなし)～32 の範囲で指定します。この指定は、テキストエディタで表示した場合の見かけを良くするためのものであり、XML 対応のブラウザでは無視されます。

デフォルト 3

制限事項 このオプションは、XML ドキュメントをエクスポートする場合にのみ使用します。

ODSCHARSET=*character-set*
出力ファイルで使用する文字セットを指定します。 文字セットには、表示や印刷で使用される文字、表語文字、数字、句読点、記号、制御文字が含まれています。文字セットの例としては、ISO-8859-1 が挙げられます。

制限事項 このオプションは、XML ドキュメントをエクスポートする場合にのみ使用します。

要件 このオプションを使用する場合、十分な注意が必要です。文字セット、エンコーディング方式、変換テーブルについて十分理解していない場合、専門家による技術的な助言なしにこのオプションを使用しないでください。

ヒント 文字セットと変換テーブル(エンコーディング方式)の組み合わせにより、特定のファイルのエンコーディングが決定されます。

参照項目: *SAS Viya 各国語サポート: リファレンスガイド*の ODSCHARSET=オプション。

ODSTRANTAB=*table-name*
出力ファイルで使用する変換テーブルを指定します。 変換テーブル(エンコーディング方式)とは、特定の文字セット内の文字を数値にマッピングするために使用される規則の集合です。変換テーブルの例としては、EBCDIC から ASCII-ISO に文字を変換するテーブルが挙げられます。*table-name* には、SAS システムが提供する任意の変換テーブルか、またはユーザー定義の変換テーブルを指定できます。この値は、SASUSER.PROFILE カタログ内または SASHELP.HOST カタログ内のどちらかに存在する SAS カタログエントリの名前でなければなりません。

制限事項 このオプションは、XML ドキュメントをエクスポートする場合にのみ使用します。

要件 このオプションを使用する場合、十分な注意が必要です。文字セット、エンコーディング方式、変換テーブルについて十分理解していない場合、専門家による技術的な助言なしにこのオプションを使用しないでください。

ヒント 文字セットと変換テーブルの組み合わせにより、特定のファイルのエンコーディングが決定されます。

参照項目: *SAS Viya 各国語サポート: リファレンスガイド*の ODSTRANTAB=オプション。

TAGSET=*tagset-name*

XMLTYPE=に指定されたマークアップタイプにより使用されるデフォルトのタグセットをオーバーライドするタグセット名を指定します。 生成されるタグを変更するには、カスタマイズしたタグセットを作成し、そのタグセットを TAGSET=オプションの値として指定します。カスタマイズしたタグセットの作成に関する詳細は、TEMPLATE プロシジャの説明を参照してください。

制限事項 このオプションは、XML ドキュメントをエクスポートする場合にのみ使用します。

要件 このオプションを使用する場合、十分な注意が必要です。XML マークアップについて十分に理解していない場合、このオプションを使用しないでください。

参照項目: 6 章, “XML Engine のタグセットの説明と使い方” (63 ページ)

例 “カスタマイズしたタグセットを使用し、XML ドキュメントをエクスポートする” (64 ページ)

注意 **XML ドキュメントをエクスポートする際にタグセットを変更した後、その変更されたタグセットにより生成された XML ドキュメントをインポートした場合、XML Engine はその XML マークアップを SAS システムに独自のフォーマットに戻せないことがあります。**

XMLCONCATENATE=NO | YES

インポートするファイルに、複数の連結された XML ドキュメントを含めるかどうかを指定します。 複数の連結された XML ドキュメントをインポートすると便利な場合があります(たとえば、アプリケーションで、Web フォームにおけるクエリや応答ごとに完全なドキュメントを生成する場合など)。

別名 XMLCONCAT=

デフォルト NO

制限事項 このオプションは、XML ドキュメントをインポートする場合にのみ使用します。

要件 XMLCONCATENATE=YES オプションを使用する場合には注意が必要です。XML ドキュメントが連結された複数の XML ドキュメントから構成されている場合、そのドキュメントの内容は標準的な XML 構成ではありません。このオプションは、利便性のために提供されているものであり、無効な XML マークアップを奨励するものではありません。

例 “連結された XML ドキュメントのインポート” (21 ページ)

XMLDATAFORM=ELEMENT | ATTRIBUTE
SAS 変数情報(名前とデータ)を含める要素のタグが、開かれた要素形式であるか、囲まれた属性形式であるかを示します。 たとえば、変数名が PRICE で、オブザベーションの値が **1.98** である場合、このオプションの値として ELEMENT を指定すると出力**<PRICE> 1.98 </PRICE>**が生成され、ATTRIBUTE を指定すると出力**<COLUMN name="PRICE"value="1.98" />**が生成されます。

デフォルト ELEMENT

制限事項 このオプションは、XML ドキュメントをエクスポートする場合にのみ使用します。

このオプションは、GENERIC マークアップタイプに対してのみ使用します。

XMLDOUBLE=DISPLAY | INTERNAL
数値のインポートまたはエクスポート結果を制御します。

DISPLAY
エクスポートを行う場合、SAS XML LIBNAME Engine は、保存されている数値変数の値を取り出し、その値の適切な表示形式を決定した後、対応する表示値を XML ドキュメントに出力します。表示値は、指定されたエンジンニックネーム、および割り当てられた出力形式による影響を受けます。エンジンニックネーム XMLV2 を指定すると、当該エンジンは割り当てられた出力形式をすべて無視し、出力形式 BEST16 を使用して値を表示します。

インポートを行う場合、SAS XML LIBNAME Engine は XML ドキュメント内の指定された要素から PCDATA (パーシング済みの文字データ)を取り出し、そのデータを数値変数の中身に変換します。

別名 FORMAT

INTERNAL
エクスポートを行う場合、SAS XML LIBNAME Engine は、保存されている数値変数の値を取り出し、生成された属性と値のペア (**rawvalue="value"**形式のもの)にその値を書き出します。SAS システムは、可搬性のあるマシン表現である base64 エンコーディング方式を使用します。(base64 とは、バイナリデータから ASCII テキストへの変換またはその逆変換を行う場合に使用される、MIME 形式に似たエンコーディング方式です。)

インポートを行う場合、SAS XML LIBNAME Engine は、XML ドキュメント内の指定の要素内にある **rawvalue=**属性から、保存されている値を取り出します。その後、同エンジンは、その値を数値変数の中身に変換します。要素内の PCDATA データは無視されます。インポートを行う場合、エンジンニックネーム XMLV2 では XMLDOUBLE=INTERNAL はサポートされていません。

別名 PRECISION

ヒント 通常、XML ドキュメントの内容が可読性よりも重要である場合、その XML ドキュメントのインポートやエクスポートを行う際に XMLDOUBLE=INTERNAL を指定します。

デフォルト DISPLAY

制限事項 XMLDOUBLE=オプションは、GENERIC マークアップタイプに対してのみ指定できます。

例 “数値のエクスポート” (10 ページ)

“数値を含む XML ドキュメントのインポート” (17 ページ)

XMLENCODING='*encoding-value*'
出力ファイルの SAS データセットのエンコーディングをオーバーライドします。 エンコーディング値にハイフンが含まれている場合、その値を引用符で囲みます。

制限事項 このオプションは、XML ドキュメントをエクスポートする場合にのみ使用します。

要件 このオプションを使用する場合、十分な注意が必要です。文字セット、エンコーディング方式、変換テーブルについて十分理解していない場合、専門家による技術的な助言なしにこのオプションを使用しないでください。

ヒント XML ドキュメントを(FTP などを使用して)環境間で転送する場合、ドキュメントの内容に基づいて適切な転送モードを決定する必要があります。ドキュメントに XML 宣言によるエンコーディング属性が含まれている場合や、XML 宣言の前にバイトオーダーマーク(BOM)が記述されている場合、その XML ドキュメントをバイナリモードで転送します。ドキュメントがこれらのどちらの条件も満たしてない場合、類似した環境間で XML ドキュメントを転送するには、そのドキュメントをテキストモードで転送します。

文字セットと変換テーブル(エンコーディング方式)の組み合わせにより、特定のファイルのエンコーディングが決定されます。

参照項目: *SAS Viya 各国語サポート: リファレンスガイド*の XMLENCODING=オプション。

XMLFILEREF=*fileref*
エクスポートまたはインポートの対象となる XML ドキュメントの物理的な場所に関連付ける SAS 名を指定します。 ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。XML Engine は、ファイル参照名により参照される任意のデータにアクセスできます。たとえば、次のプログラムは、XML ドキュメント Wilma.XML の内容を出力します。

```
filename cartoon '/u/xmldata/wilma.xml';

libname bedrock xml xmlfileref=cartoon;

proc print data=bedrock.wilma;
run;
```

ヒント URL アクセス方式を使用して、XML ドキュメントに割り当てられているファイル参照名を参照する場合にも、XMLMap を指定する必要があります。XMLMap を指定すると、XML Engine は XML ドキュメントを単一パスで処理します。XMLMap を指定する必要があるかどうかは、お使いの Web サーバーにより決まります。たとえば、“URL アクセス方

式を使用してファイル参照名を参照する” (55 ページ)を参照してください。

XMLMAP=*fileref* | '*XMLMap*'
特定の XMLMap 構文を含む、ユーザーが作成した XML ドキュメントを指定します。 XMLMap 構文は、インポートやエクスポートを行う際に、XML マークアップの変換方法を XML Engine に伝えます。XMLMap 構文自体も XML マークアップです。

fileref
XMLMap の物理的な場所に関連付けられる SAS 名を指定します。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

ヒント URL アクセス方式を使用して XMLMap にファイル参照名を割り当てるには、お使いの Web サーバーで、XMLMap のファイル拡張子が**.map** ではなく**.xml** であることが必要となる場合があります。

'***XMLMap***'
XMLMap の物理的な場所を指定します。

完全なパス名とファイル名を含める必要があります。ファイル拡張子として.map を使用することを推奨します。物理名は一重引用符または二重引用符で囲みます。

たとえば、次のステートメントは、XML ドキュメント MY.XML をインポートする際に、特定の XMLMap 構文を含んでいる XMLMap ファイルとして MY.MAP を指定します。XML Engine は、この XML ドキュメントを SAS データセット TEST.MY として変換します。この例では、LIBNAME ステートメントのオプションとして XMLMAP=を使用しています。

```
libname test xml '/u/xmldata/my.xml' xmlmap='/u/xmldata/my.map';

proc print data=test.my;
run;
```

制限事項 XMLV2 エンジンニックネームは、バージョン 1.2、1.9、2.1 の XMLMap 構文をサポートしています。XMLV2 エンジンニックネームは、バージョン 1.0 と 1.1 の XMLMap 構文をサポートしていません。

要件 XMLMap を指定する場合、XMLTYPE=XMLMAP を指定するか、またはマークアップタイプの指定を省略します。XMLMAP 以外のマークアップタイプを明示的に指定すると(例: XMLTYPE=GENERIC)、エラーが発生します。

参照項目: 9 章, “XMLMap 構文: 概要” (91 ページ)

例 5 章, “XMLMap を使用し、XML ドキュメントをインポートする” (29 ページ)

XMLMETA=DATA | SCHEMADATA | SCHEMA
エクスポートするマークアップ内にメタデータ関連情報を含めるかどうかを指定します。または、入力される XML ドキュメント内に含まれているメタデータ関連情報をインポートするかどうかを指定します。 メタデータ関連情報とは、テーブルマークアップ内にある列の特性(型、長さ、レベルなど)を記述したメタデータのことです。メタデータ関連情報を含めると、SAS データセットから XML ドキュメントをエクスポートし、同ドキュメントを外部製品上で処理する場合に役立ちます。

DATA
メタデータ関連情報を無視します。DATA を指定すると、エクスポートされるマークアップ内にデータ内容のみが含められます。または、入力 XML ドキュメントに含まれているデータ内容のみがインポートされます。

SCHEMADATA
エクスポートされるマークアップ内にデータ内容とメタデータ関連情報の両方を含めます。または、入力 XML ドキュメントに含まれているデータ内容とメタデータ関連情報の両方をインポートします。

SCHEMA
データ内容を無視します。SCHEMA を指定すると、エクスポートされるマークアップ内にメタデータ関連情報のみが含められます。または、入力 XML ドキュメントに含まれているメタデータ関連情報のみがインポートされます。

デフォルト DATA

制限事項 このオプションは、GENERIC マークアップタイプに対してのみ使用します。

操作 XMLMETA=SCHEMADATA および XMLSCHEMA=オプションを指定すると、LIBNAME ステートメントで指定された XML ドキュメントの物理的な場所にデータが出力されます。メタデータ関連情報は、データとは別に、XMLSCHEMA=オプションに指定された物理的な場所に出力されます。XMLSCHEMA=を省略すると、メタデータ関連情報は、データ内容に埋め込まれた形式で XML ドキュメントに出力されます。

ヒント SAS 9 より前のバージョンでは、XMLMETA=オプションと同じ機能を提供するためにキーワード XMLSCHEMA=を使用していました。SAS 9 では、オプションキーワード XMLSCHEMA=が XMLMETA=に変更されました。SAS 9.1 では、XMLSCHEMA=オプションを使用した新機能が追加されています。

XMLPROCESS=CONFORM | PERMIT
XML Engine が W3C 規格に従っていない文字データをどのように処理するかを指定します。

CONFORM
XML が W3C 規格に従っていることを必要とします。W3C 規格では、文字データの場合、左山かっこ(<)、アンパサンド(&)、アポストロフィー(')などの特定の文字は、**&**のような文字参照または文字列を使用してエスケープする必要があると規定されています。たとえば、属性値に一重引用符や二重引用符を含める場合、一重引用符(')は**'**として、二重引用符(")は**"**として表されます。

PERMIT
W3C 規格に従っていない文字データの受け入れを許可します。この場合、アポストロフィー、二重引用符、アンパサンドのような非エスケープ文字が文字データとして受け入れられます。

制限事項 文字データ内にあるエスケープされていない山かっこは受け入れられません。

XMLPROCESS=PERMIT を使用する場合には注意が必要です。XML ドキュメントが非エスケープ文字を含んでいる場合、そのドキュメントの内容は標準的な XML 構成ではありません。このオプションは、利便性のために提供されているものであり、無効な XML マークアップを奨励するものではありません。

デフォルト CONFORM

例 “非エスケープ文字データを含む XML ドキュメントのインポート” (19 ページ)

XMLSCHEMA=*fileref* | '*external-file*'
メタデータ関連の情報を含める外部ファイルを指定します。

fileref
出力ファイルの物理的な場所に関連付けられる SAS 名を指定します。ファイル参照名を割り当てるには、FILENAME ステートメントを使用します。

'***external-file***'
メタデータ関連情報を含めるファイルの物理的な場所を指定します。完全なパス名とファイル名を含める必要があります。物理名は一重引用符または二重引用符で囲みます。

制限事項 このオプションは、XML ドキュメントをエクスポートする場合にのみ使用します。

このオプションは、GENERIC マークアップタイプで XMLMETA=SCHEMADATA が指定されている場合にのみ使用します。

操作 XMLMETA=SCHEMADATA および XMLSCHEMA=オプションを指定すると、LIBNAME ステートメントで指定された XML ドキュメントの物理的な場所にデータが出力されます。メタデータ関連情報は、データとは別に、XMLSCHEMA=オプションに指定された物理的な場所に出力されます。XMLSCHEMA=を省略すると、メタデータ関連情報は、データ内容に埋め込まれた形式で XML ドキュメントに出力されます。

XMLTYPE=GENERIC | XMLMAP
XML マークアップタイプを指定します。

GENERIC
正しく構成されたシンプルな XML マークアップタイプを指定します。この場合、XML ドキュメントは、ルート(包含)要素と反復要素のインスタンスで構成されます。GENERIC を指定すると、変数の属性がデータ内容から決定されます。

要件 インポートを行う場合、GENERIC マークアップタイプは特定の物理構造を必要とします。

参照項目: “GENERIC マークアップタイプを使用した XML ドキュメントのインポートに必要な物理構造の条件について” (30 ページ)

例 “SAS 日付値、SAS 時刻値、SAS 日時値を含む XML ドキュメントのエクスポート” (9 ページ)

“数値のエクスポート” (10 ページ)

“GENERIC マークアップタイプを使用した XML ドキュメントのインポート” (15 ページ)

XMLMAP

XMLMap により決定される XML マークアップを指定します。XMLMap とは、特定の XMLMap 構文を含んでいる、ユーザー作成の XML ドキュメントです。

XMLMap の構文は、SAS データセットを特定の XML ドキュメントの構造に対応付ける方法を XML Engine に伝えます。LIBNAME ステートメントで XMLMap を指定するには 、XMLMAP=オプショ (84 ページ)を使用します。

制限事項 XMLMap で制御される XML ドキュメントのエクスポートを単一の SAS データセットに対して行うよう制限します。

例 “XMLMap を使用し、階層構造を持つ XML ドキュメントをエクスポートする” (25 ページ)

デフォルト GENERIC

ヒント INDENT=、XMLDATAFORM=、XMLMETA= (利用可能な場合)、TAGSET=などのオプションを指定することによりマークアップを制御できます。

3 部

XMLMap ファイル参照

9 章
XMLMap 構文: 概要

XMLMap 構文の使用

本章では、バージョン 2.1 の XMLMap 構文の XML 要素について説明します。各要素は、ユーザーが通常それらを XMLMap 内に含める順番でリストされています。具体的には次のような順番になります。

- XMLMap 内の最初の要素は SXLEMAP です。これは一次(ルート)包含要素であり、生成される出力ファイルの定義を含んでいます。詳細は“SXLEMAP 要素” (95 ページ)を参照してください。
- 名前空間要素は、XML 名前空間を定義します。名前空間では、URI (Uniform Resource Identifier)を使用して要素名と属性名を修飾することにより、それらの名前を区別します。詳細は“XML 名前空間の要素” (96 ページ)を参照してください。
- XMLMap を使用してエクスポートを行う場合、エクスポート要素を含める必要があります。詳細は“エクスポートの要素” (97 ページ)を参照してください。
- テーブル要素は、SAS データセットを定義します。詳細は“テーブルの要素” (99 ページ)を参照してください。
- 列要素は、SAS データセットの列変数を定義します。詳細は“列の要素” (103 ページ)を参照してください。

注意:

XMLMap マークアップでは、XML 自体と同様に、大文字小文字が区別されます。 タグ名は大文字で、要素の属性は小文字で記述する必要があります。例: **<SXLEMAP version="2.1">**.同様に、サポートされている XPath 構文でも大文字小文字が区別されます。

XMLMap 構文の比較

利用可能な XMLMap 構文の一覧を次の表に示します。記号■は、構文がインポート用かエクスポート用かを示しています。

表 9.1 XMLMap 構文

構文	説明	インポート	エクスポート
SXLEMAP (95 ページ)	一次(ルート)包含要素です	■	■
NAMESPACES (96 ページ)	XML 名前空間を定義するための 1 つ以上の NS 要素を指定します	■	■
NS (97 ページ)	一意の URL を参照することにより XML 名前空間を定義します	■	■
OUTPUT (98 ページ)	SAS データセットをエクスポートする場合に、1 つ以上の HEADING 要素と 1 つの TABLEREF 要素を指定します		■
HEADING (98 ページ)	1 つ以上の ATTRIBUTE 要素を指定します		■
ATTRIBUTE (98 ページ)	ファイル属性情報を指定します		■
TABLEREF (98 ページ)	テーブル名を指定します		■
TABLE (99 ページ)	データセットの定義を指定します	■	■
TABLE-PATH (99 ページ)	変数の場所パスを指定します	■	■
TABLE-END-PATH (101 ページ)	処理を停止するための場所パスを指定します	■	■
TABLE-DESCRIPTION (103 ページ)	SAS データセットの説明を指定します	■	■
COLUMN name= (103 ページ)	変数名を指定します	■	■
COLUMN retain= (103 ページ)	入力バッファの内容を指定します	■	■
COLUMN class= (104 ページ)	変数の型を指定します	■	■

構文	説明	インポート	エクスポート
TYPE (104 ページ)	変数の SAS データ型を指定します	■	■
DATATYPE (105 ページ)	読み込み対象となるデータの型を指定します	■	■
DEFAULT (106 ページ)	欠損値のデフォルト値を指定します	■	■
ENUM (106 ページ)	変数で使用できる有効な値のリストを指定します	■	■
FORMAT (106 ページ)	変数に割り当てる SAS 出力形式を指定します	■	■
INFORMAT (107 ページ)	変数に割り当てる SAS 入力形式を指定します	■	■
DESCRIPTION (107 ページ)	変数の説明を指定します	■	■
LENGTH (107 ページ)	文字変数の最大フィールド格納長を指定します	■	■
PATH (108 ページ)	現在の変数の場所パスを指定します	■	■
INCREMENT-PATH (110 ページ)	カウンタ変数の累積値をインクリメントするための場所パスを指定します	■	■
RESET-PATH (111 ページ)	カウンタ変数の累積値をゼロにリセットするための場所パスを指定します	■	■
DECREMENT-PATH (111 ページ)	カウンタ変数の累積値を 1 デクリメントするための場所パスを指定します	■	■

10 章
XMLMap 構文バージョン 2.1

ディクショナリ

SXLEMAP 要素

生成される出力ファイルの定義を含んでいる一次(ルート)包含要素です。この要素は、正しく構成された XML を生成するための定義に関する制約を提供します。

制限事項: XML ドキュメントをインポートする場合、この定義では複数の SAS データセットを定義できます。SAS データセットから XML ドキュメントをエクスポートする場合、この定義では 1 つの出力 XML ファイルだけを定義できます。

要件 SXLEMAP 要素は必須です。

構文

SXLEMAP version="*number*" name="*XMLMap*" description="*description*"

属性

version="*number*"
XMLMap 構文のバージョンを指定します。ドキュメント化されている XMLMap 構文のバージョンは 2.1 であり、完全な機能を利用できるようにするにはバージョン 2.1 を指定する必要があります。

デフォルト デフォルトのバージョンは、XMLMap 構文の最初のバージョンです。最初のバージョンは、XMLMap 構文の以前のリリースとの互換性のために保持されているものです。既存の XMLMaps はバージョン 2.1 にアップデートすることを推奨します。

制限事項 XMLV2 エンジンニックネームは、バージョン 1.2、1.9、2.1 の XMLMap 構文をサポートしています。XMLV2 エンジンニックネームは、バージョン 1.0 と 1.1 の XMLMap 構文をサポートしていません。

name="*XMLMap*"
オプション属性であり、XMLMap のファイル名を指定します。

description="*description*"
オプション属性であり、XMLMap の説明を指定します。説明は最大 256 文字までです。

詳細

次の例では、SXLEMAP 要素で 3 つの属性を指定しています。この SXLEMAP 要素には 2 つの TABLE 要素が含まれています。

```
<?xml version="1.0" ?>
<SXLEMAP version="2.1" name="Myxmlmap" description="sample XMLMap">
  <TABLE name="test1">
    .
    .
    .
  </TABLE>
  <TABLE name="test2">
    .
    .
    .
  </TABLE>
  </SXLEMAP>
```

XML 名前空間の要素

XML 名前空間を定義します。

構文

NAMESPACES count="*number*"
NS id="*number*" <prefix="*name*">

要素

NAMESPACES count="*number*"
このオプション要素には、XML 名前空間を定義するための 1 つ以上の NS 要素を含めます。次に例を示します。 **<NAMESPACES count="2">**

XMLMap 名前空間要素を使用すると、XML 名前空間により修飾される同じ名前の要素を含む XML ドキュメントをインポートできるようになります。また、XMLMap 名前空間要素を使うと、インポートされた XML ドキュメント内に含まれている XML 名前空間を維持することや、SAS データセットから名前空間を含む XML ドキュメントをエクスポートすることが可能となります。

XML 名前空間とは、URI (Uniform Resource Identifier)を使用して要素名や属性名を修飾することによりそれらの名前を区別する W3C 規格です。たとえば、ある XML ドキュメントに CUSTOMER 要素と PRODUCT 要素が含まれ

ており、両方の要素が 1 つのネストされた ID 要素を含んでいる場合、XML 名前空間は、それぞれのネストされた ID 要素を一意なものにします。

count="*number*"
定義される XML 名前空間の数を指定します。

要件 count=属性は必須です。この属性に指定する値は、NS 要素の総数に一致しなければなりません。

例 “XMLMap に名前空間要素を挿入する” (58 ページ)

NS id="*number*" <prefix="*name*">
このオプション要素は、一意の URL を参照することにより XML 名前空間を定義します。URI とは、インターネット上のリソースを識別する文字列です。URI は、XML パーサーにより単純な文字列として扱われます。URI を指定した場合でも、必ずしもその URI を使用して情報が取り出されるとは限りません。最も一般的な URI としては、インターネットドメインアドレスを特定する URL (Uniform Resource Locator)が挙げられます。URI を名前空間名として使用する場合、XML 名前空間に関する W3C 規格と同じ規則に従う必要があります。例: **<NS id="1" prefix="freq"> http://www.hurricanefrequency.com </NS>**.

注: URI には非エスケープ文字は使用しないことを推奨します。

id="*number*"
XML 名前空間の ID 番号を指定します。

要件 id=属性は必須です。

変数定義では、定義しようとしている要素に先立って、ID 番号を場所パスに含める必要があります。“PATH syntax="*type*"” (108 ページ)を参照してください。

prefix="*name*"
参照される URI に関連付けられる修飾名を指定します。この接頭辞は、要素や属性がどの XML 名前空間に所属しているかを示すために、各要素および属性で使用されます。接頭辞名は、要素名に関する W3C 規格と同じ規則に従う必要があります。

要件 参照される URI は一意でなければなりません。

NS 要素の総数は、NAMESPACES 要素内の count=属性の値と一致する必要があります。

ヒント URI では非エスケープ文字を使用しないことを推奨します。

例 “XMLMap に名前空間要素を挿入する” (58 ページ)

エクスポートの要素

XML ドキュメントをインポートした際に作成された XMLMap を使用して、SAS データセットから XML ドキュメントをエクスポートします。

制限事項: エンジンは、1 つの SAS データセットからのエクスポートのみをサポートしています。

構文

OUTPUT
HEADING
ATTRIBUTE name="*name*" value="*value*"
TABLEREF name="*name*"

要素

OUTPUT
このオプション要素には、SAS データセットをエクスポートする場合に、1 つ以上の HEADING 要素と 1 つの TABLEREF 要素を含めます。

要件 SAS データセットを XML ドキュメントとしてエクスポートする際に XMLMap でバージョン 1.9 または 2.1 を指定した場合、その XMLMap 内に OUTPUT 要素を含める必要があります。

例 “XMLMap を使用し、階層構造を持つ XML ドキュメントをエクスポートする” (25 ページ)

HEADING
このオプション要素には、1 つ以上の ATTRIBUTE 要素を含めます。

ATTRIBUTE name="*name*" value="*value*"
このオプション要素には、エクスポートされる XML ドキュメントに関する追加のファイル属性情報(スキーマ参照やその他の一般的な属性など)を含めます。指定された名前と値のペアは、エクスポートされる XML ドキュメント内で生成される最初の要素に追加されます。たとえば、**<NHL description="Teams of the National Hockey League">**のようになります。

name="*name*"
ファイル属性の名前を指定します。たとえば、**name="description"**のように記述します。

value="*value*"
属性の値を指定します。たとえば、**value="Teams of the National Hockey League"**のように記述します。

TABLEREF name="*name*"
このオプション要素は、エクスポートされる XMLMap 内のテーブル名を指定します。

name="*name*"
エクスポートされる XMLMap 内のテーブル名を指定します。この名前は、その XMLMap 定義内で一意となる、最大 32 文字までの有効な SAS 名でなければなりません。

制限事項 指定できる TABLEREF 要素は 1 つだけです。

要件 この属性に指定された名前は、TABLE 要素の name=属性の値に一致する必要があります。

テーブルの要素

SAS データセットを定義します。

構文

TABLE description="*description*" name="*data-set-name*"
TABLE-PATH syntax="*type*"
TABLE-END-PATH syntax="*type*" beginend="BEGIN | END"
TABLE-DESCRIPTION

要素

TABLE description="*description*" name="*data-set-name*"
この要素には、データセット定義を含めます。例: **<TABLE name="channel">**.

description="*description*"
オプション属性であり、SAS データセットの説明を指定します。説明は最大 256 文字までです。

name="*data-set-name*"
データセットの名前を指定します。この名前は、その XMLMap 定義内で一意となる、最大 32 文字までの有効な SAS 名でなければなりません。

要件 name=属性は必須です。

要件 TABLE 要素は必須です。

操作 TABLE 要素には、TABLE-PATH、TABLE-END-PATH、TABLE-DESCRIPTION、COLUMN の各要素のうち 1 つまたは複数を含めることができます。

TABLE-PATH syntax="*type*"
場所パスは、SAS データセットの変数を収集するには、XML ドキュメント内のどの位置を検索し特定の要素にアクセスする必要があるかを XML Engine に伝えます。場所パスは、XML ドキュメント内の反復要素インスタンス(SAS データセットのオブザベーション境界となるもの)を定義します。オブザベーション境界は、一定の個数の列を伴う任意の個数の行の集合(すなわち表)に変換されます。

たとえば、XML ドキュメント RSS.XML (“XMLMap を使用し、XML ドキュメントを複数の SAS データセットとしてインポートする” (36 ページ)で使用したもの)を使用する場合に、次の TABLE-PATH 要素を指定すると、次の処理が実施されます。

```
<TABLE-PATH syntax="XPath"> /rss/channel/item </TABLE-PATH>
```

1. XML Engine は、開始タグ<ITEM>を検出するまで XML マークアップを読み込みます。
2. XML Engine は入力バッファをクリアし、カウンタを MISSING (デフォルト)に設定した後、COLUMN 要素の定義に基づいて変数名の要素をスキャンします。値が検出されると、それらは入力バッファに読み込まれます。

(XML Engine がカウンタを MISSING にリセットするかどうかは、DEFAULT 要素と、COLUMN 要素の retain=属性により決定されます)。

3. 終了タグ</ITEM>を検出した時点で、XML Engine は、完了した入力バッファを 1 つのオブザベーションとして SAS データセットに書き出します。
4. 開始タグ<ITEM>と終了タグ</ITEM>のシーケンスを検出するたびに、入力ストリームのファイルの終わりを検出するまでか、または TABLE-END-PATH (指定されている場合)が実施されるまで、この処理が繰り返されます。この結果、6 個のオブザベーションが生成されます。

syntax="*type*"
オプション属性であり、場所パス内に記述する構文のタイプを指定します。この構文は、W3C 規格に準拠した有効な XPath 構成になります。次に例を示します。 **syntax="XPath"**

デフォルト XPath

要件 この値は、XPath または XPathENR のどちらかでなければなりません。

XML 名前空間を NAMESPACES 要素で定義する場合、構文のタイプを XPathENR (埋め込み型の名前空間参照を持つ XPath)として指定する必要があります。これは、この構文が XPath 規格とは異なるためです。次に例を示します。 **syntax="XPathENR"**

注意:
表の場所パスを指定します。表の場所パスはオブザベーション境界となるものです。開始タグと終了タグのペアリングシーケンスが原因で、オブザベーション境界の特定がトリッキーになる場合があります。 表の場所パスは、どの終了タグを検出した時点で XML Engine が完了した入力バッファを SAS データセットに出力するかを決定します。適切な終了タグを指定しない場合、結果として独立したオブザベーションではなく、連結されたデータ文字列や予期せぬ列の集合が生成されることがあります。たとえば、“データの連結を避けるために、オブザベーションの境界を決定する” (46 ページ) および“最適な列を選択するために、オブザベーションの境界を決定する” (48 ページ)を参照してください。

要件 TABLE-PATH 要素は必須です。

XML 名前空間を NAMESPACES 要素で定義する場合、定義しようとしている要素に先立って、場所パスに ID 番号を含める必要があります。ID 番号は中かっこで囲みます。次に例を示します。 **<TABLE-PATH syntax="XPathENR">/Table/{1}Hurricane</TABLE-PATH>**

XPath 構成は、XML 構造の各要素に関して UNIX ライクなパス記述を配置する公式規格です。XPath の構文では大文字小文字が区別されます。たとえば、要素タグ名が大文字である場合、そのタグ名は場所パスでも大文字で記述する必要があります。同様に、要素タグ名が小文字である場合、そのタグ名は場所パスでも小文字で記述する必要があります。すべての場所パスは、ルート包含要素(スラッシュ'/'で表される)で始まるか、または"任意の親"バリアント(ダブルスラッシュ'//'で表される)で始ま

る必要があります。W3C で規定されているその他の形式は、現時点ではサポートされていません。

TABLE-END-PATH syntax="*type*" beginend="BEGIN | END"
このオプションの最適化要素は、ファイルの終了前に XML ドキュメントの処理を停止することにより、リソースを節約します。場所パスは、XML ドキュメントの処理を停止するには、XML ドキュメント内のどの位置を検索し特定要素にアクセスする必要があるかを XML Engine に伝えます。

たとえば、XML ドキュメント RSS.XML (“XMLMap を使用し、XML ドキュメントを複数の SAS データセットとしてインポートする” (36 ページ)で使用したもの)を使用する場合、1 つの開始タグ<CHANNEL>と 1 つの終了タグ</CHANNEL>のみが存在します。TABLE-PATH 要素の場所パス**<TABLE-PATH syntax="XPath"> /rss/channel </TABLE-PATH>**により、XML Engine は、最初の開始タグ<ITEM>を検出した後、新規データを入力バッファに格納しないにもかかわらず、XML ドキュメント全体を処理します。これは、残りの要素が適格ではなくなるためです。TABLE-END-PATH 要素の場所パス<**<TABLE-END-PATH syntax="XPath" beginend="BEGIN"> /rss/channel/item </TABLE-END-PATH>**は、開始タグ<ITEM>を検出した時点で処理を停止するよう XML Engine に伝えます。

したがって、これら 2 つの場所パスにより、XML Engine は、XML ドキュメント RSS.XML 全体を処理するのではなく、RSS.XML 内の強調表示されているデータのみを処理して CHANNEL データセットを生成します。

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rss version="0.91">
  <channel>
    <title>WriteTheWeb</title>
    <link>http://writetheweb.com</link>
    <description>News for web users that write back
      </description>
    <language>en-us</language>
    <copyright>Copyright 2000, WriteTheWeb team.
      </copyright>
    <managingEditor>editor@writetheweb.com
      </managingEditor>
    <webMaster>webmaster@writetheweb.com</webMaster>
    <image>
      <title>WriteTheWeb</title>
      <url>http://writetheweb.com/images/mynetscape88.gif
        </url>
      <link>http://writetheweb.com</link>
      <width>88</width>
      <height>31</height>
      <description>News for web users that write back
        </description>
      </image>
    <item>
      <title>Giving the world a pluggable Gnutella</title>
       <link>http://writetheweb.com/read.php?item=24</link>
    <description>WorldOS is a framework on which to build programs
       that work like Freenet or Gnutella-allowing distributed
       applications using  peer-to-peer routing.</description>
    </item>
    <item>
```

```
      .
      .
      .
  </channel>
</rss>
```

syntax="*type*"
オプション属性であり、場所パス内に記述する構文のタイプを指定します。この構文は、W3C 規格に準拠した有効な XPath 構成になります。XML Engine がサポートしている XPath 形式では、生成される SAS データセットから除外する要素や属性を個別に選択できます。次に例を示します。 **syntax="XPath"**

デフォルト XPath

要件 この値は、XPath または XPathENR のどちらかでなければなりません。

XML 名前空間を NAMESPACES 要素で定義する場合、構文のタイプを XPathENR (埋め込み型の名前空間参照を持つ XPath)として指定する必要があります。これは、この構文が XPath 規格とは異なるためです。次に例を示します。 **syntax="XPathENR"**

beginend="BEGIN | END"
オプション属性であり、要素の開始タグを検出した時点または要素の終了タグを検出した時点で、処理を停止するよう指定します。

デフォルト BEGIN

デフォルト XML ドキュメントの最後の終了タグに出会うまで、処理を続行します。

要件 XML 名前空間を NAMESPACES 要素で定義する場合、定義しようとしている要素に先立って、場所パスに ID 番号を含める必要があります。ID 番号は中かっこで囲みます。次に例を示します。 **<TABLE-END-PATH syntax="XPathENR">/Table/{1}Hurricane</TABLE-END-PATH>**

XPath 構成は、XML 構造の各要素に関して UNIX ライクなパス記述を配置する公式規格です。XPath の構文では大文字小文字が区別されます。たとえば、要素タグ名が大文字である場合、そのタグ名は場所パスでも大文字で記述する必要があります。同様に、要素タグ名が小文字である場合、そのタグ名は場所パスでも小文字で記述する必要があります。すべての場所パスは、ルート包含要素(スラッシュ'/'で表される)で始まるか、または"任意の親"バリアント(ダブルスラッシュ'//'で表される)で始まる必要があります。W3C で規定されているその他の形式は、現時点ではサポートされていません。

操作 TABLE-END-PATH 要素は、オブザベーション境界には影響を与えません。オブザベーション境界は、TABLE-PATH 要素により決定されます。

ヒント 処理を停止する場所を指定すると、階層構造は持っているが、反復インスタンスデータには一般に適していない XML ドキュメントを処理する場合に役立ちます。

例 “XMLMap を使用し、XML ドキュメントを複数の SAS データセットとしてインポートする” (36 ページ)

TABLE-DESCRIPTION
オプションの要素であり、SAS データセットの説明を最大 256 文字で指定します。例: **<TABLE-DESCRIPTION> Data Set contains TV channel information </TABLE-DESCRIPTION>**.

列の要素

SAS データセットの変数を定義します。

構文

COLUMN name="*name*" retain="NO | YES" class="ORDINAL"
TYPE
DATATYPE
DEFAULT
ENUM
FORMAT width="*w*" ndec="*d*"
INFORMAT width="*w*" ndec="*d*"
DESCRIPTION
LENGTH
PATH syntax="*type*"
INCREMENT-PATH syntax="*type*" beginend="BEGIN | END"
RESET-PATH syntax="*type*" beginend="BEGIN | END"
DECREMENT-PATH syntax="*type*" beginend="BEGIN | END"

要素

COLUMN name="*name*" retain="NO | YES" class="ORDINAL"
この要素には、変数定義を含めます。例: **<COLUMN name="Title">**.

name="*name*"
変数名を指定します。この名前は、最大 32 文字までの有効な SAS 名でなければなりません。

要件 name=属性は必須です。

retain="NO | YES"
オプション属性であり、各オブザベーションの開始時の入力バッファの内容を決定します。

NO
各オブザベーションの開始時の値を、MISSING または DEFAULT 要素の値(指定されている場合)のどちらかに設定します。

YES
新しい欠損値でない値により置き換えられるまで、現在の値を保持します。YES を指定した場合、DATA ステップ処理で RETAIN ステートメントを使用した場合と同じような結果が得られます。これにより、オブザベーションが出力 SAS データセットに書き出された後も、処理された値を強制的に保持できます。

デフォルト NO

例 “階層データを関連データセットとしてインポートする” (39 ページ)

class="ORDINAL"
オプション属性であり、変数の型を指定します。

ORDINAL
変数が数値カウンタ変数であることを指定します。カウンタ変数は、INCREMENT-PATH 要素または DECREMENT-PATH により指定される場所パスが検出される回数を記録します(カウンタ変数は、DATA ステップの自動変数_N_に似ています)。カウンタ変数の値は、この場所パスが検出されるたびに 1 ずつインクリメントまたはデクリメントされます。カウンタ変数は、同じ名前のデータ要素の個々のオカレンスを特定する場合や、オブザベーションの数をカウントする場合に役立ちます。

制限事項 XML ドキュメントをエクスポートする場合、class="ORDINAL"属性を持つ変数は出力 XML ドキュメントには含められません。

要件 INCREMENT-PATH 要素または DECREMENT-PATH 要素を使用する必要があります。PATH 変数は使用できません。

TYPE 要素では、SAS データ型を数値として指定する必要があります。DATATYPE 要素では、データ型を整数として指定する必要があります。

例 “生成された数値キーを含むキーフィールドを挿入する” (42 ページ)

要件 少なくとも 1 つの COLUMN 要素が必要となります。

操作 COLUMN 要素には、DATATYPE、DEFAULT、ENUM、FORMAT、INFORMAT、DESCRIPTION、LENGTH、TYPE、PATH、INCREMENT-PATH、DECREMENT-PATH、RESET-PATH の各要素のうち、変数属性を記述する要素を 1 つ以上含めることができます。

TYPE
変数の SAS データ型(文字または数値)を指定します。SAS システムは、このデータ型を使用して変数のデータを保存します。たとえば「**<TYPE> numeric </TYPE>**」は、変数の SAS データ型が数値であることを表します。

要件 TYPE 要素は必須です。

ヒント 浮動小数点型を割り当てるには、
<DATATYPE> float </DATATYPE>
<TYPE> numeric </TYPE>

を使用します。

SAS システムで出力形式を適用するには、FORMAT 要素を使用します。

入力におけるデータ型の変換を制御するには、INFORMAT 要素を使用します。例: **<INFORMAT> datatime </INFORMAT>**。

DATATYPE

XML ドキュメントから変数にデータを読み込む際に使用するデータ型を指定します。たとえば、**<DATATYPE> string </DATATYPE>**は、データがアルファベット文字を含むことを指定します。

指定できるデータ型は次のいずれかになります。

string
データはアルファベット文字を含んでおり、計算に使用される数字は含んでいないことを指定します。

integer
データは、計算に使用される整数を含んでいることを指定します。

double
データは、浮動小数点数を含んでいることを指定します。

datetime
入力が有効な日時値を表すことを指定します。

- XML 規格 ISO 8601 フォーマットの形式。デフォルトの形式は *yyyy-mm-ddThh:mm:ss.ffffff* です。
- SAS 入力形式(SAS システムが提供する入力形式かまたはユーザー定義の入力形式)により入力を有効な SAS 日時値に変換できる形式。詳細は、INFORMAT 要素 (107 ページ)を参照してください。

date
入力が有効な日付値を表すことを指定します。この値は、次のいずれかの形式で指定します。

- XML 規格 ISO 8601 フォーマットの形式。デフォルトの形式は *yyyy-mm-dd* です。
- SAS 入力形式(SAS システムが提供する入力形式かまたはユーザー定義の入力形式)により入力を有効な SAS 日付値に変換できる形式。詳細は、INFORMAT 要素 (107 ページ)を参照してください。

time
入力が有効な時間値を表すことを指定します。この値は、次のいずれかの形式で指定します。

- XML 規格 ISO 8601 フォーマットの形式。デフォルトの形式は *hh:mm:ss.ffffff* です。
- SAS 入力形式(SAS システムが提供する入力形式かまたはユーザー定義の入力形式)により入力を有効な SAS 日付値に変換できる形式。詳細は、INFORMAT 要素 (107 ページ)を参照してください。

制限事項 XMLMap 構文のバージョン 1.9 および 2.1 では、それより前のバージョンの XMLMap 構文を受け入れません。

要件 DATATYPE 要素は必須です。

DEFAULT
オプション要素であり、変数に割り当てる欠損値のデフォルト値を指定します。欠損データに非欠損値を割り当てるには、DEFAULT 要素を使用します。たとえば「**<DEFAULT> single </DEFAULT>**」は、値が欠損しているとき、**single** という値を割り当てることを表します。

デフォルト　デフォルトでは、XML Engine は欠損値を MISSING に設定します。

例　“最適な列を選択するために、オブザベーションの境界を決定する” (48 ページ)

ENUM
オプション要素であり、変数で使用できる有効な値のリストを指定します。ENUM 要素には、値をリストする 1 つ以上の VALUE 要素を含めることができます。ENUM 要素を使用すると、XML ドキュメント内の値の有効性が値リストに照らして検証されます。値が有効でない場合、その値は MISSING に設定される(デフォルト)か、または DEFAULT 要素で指定された値に設定されます。なお、DEFAULT に指定するのは、ENUM のいずれかの値でなければなりません。

```
<COLUMN name="filing_status">
  .
  .
  .
  <DEFAULT> single </DEFAULT>
  .
  .
  .
  <ENUM>
    <VALUE> single </VALUE>
    <VALUE> married filing joint return </VALUE>
    <VALUE> married filing separate return </VALUE>
    <VALUE> head of household </VALUE>
    <VALUE> qualifying widow(er) </VALUE>
  </ENUM>
</COLUMN>
```

例　“最適な列を選択するために、オブザベーションの境界を決定する” (48 ページ)

FORMAT width="*w*" ndec="*d*"
オプション要素であり、変数に割り当てる SAS 出力形式を指定します。出力形式名の長さは、文字出力形式の場合は最大 31 文字まで、数値出力形式の場合は最大 32 文字までとなります。SAS 出力形式とは、SAS システムが値の書き出しの際に使用する命令のことです。出力形式を使うと、値が出力される見かけを制御できます。出力形式名の一部にピリオド(.)を含めることはできません。幅および長さは、出力形式名の一部としてではなく、属性として指定します。

ISO 8601 SAS 形式をはじめ、SAS 形式の一覧が *SAS Viya Formats and Informats: Reference* に載っています。

width="*w*"
オプション属性であり、出力形式の幅を指定します。ほとんどの出力形式で、幅は出力データにおける列数を意味します。

ndec="*d*"
オプション属性であり、数値出力形式の decimal スケーリングファクタを指定します。

たとえば次のように記述します。

```
<FORMAT> E8601DA </FORMAT>
<FORMAT width="8"> best </FORMAT>
<FORMAT width="8" ndec="2"> dollar </FORMAT>
```

例 “最適な列を選択するために、オブザベーションの境界を決定する” (48 ページ)

INFORMAT width="*w*" ndec="*d*"
オプション要素であり、変数に割り当てる SAS 入力形式を指定します。入力形式名の長さは、文字入力形式の場合は最大 30 文字まで、数値入力形式の場合は最大 31 文字までとなります。SAS 入力形式とは、SAS システムが値を変数に読み込む(すなわち値を変数に保存する)際に使用する命令のことです。入力形式名の一部にピリオド(.)を含めることはできません。幅および長さは、入力形式名の一部としてではなく、属性として指定します。

ISO 8601 SAS 形式をはじめ、SAS 形式の一覧が *SAS Viya Formats and Informats: Reference* に載っています。

たとえば次のように記述します。

```
<INFORMAT> E8601DA </INFORMAT>
<INFORMAT width="8"> best </INFORMAT>
<INFORMAT width="8" ndec="2"> dollar </INFORMAT>
```

width="*w*"
オプション属性であり、入力形式の幅を指定します。ほとんどの入力形式で、幅は入力データにおける列数を意味します。

ndec="*d*"
オプション属性であり、数値入力形式の decimal スケーリングファクタを指定します。SAS システムは、この属性に指定された数を指数とする 10 の累乗により、入力データを除算します。

例 “最適な列を選択するために、オブザベーションの境界を決定する” (48 ページ)

DESCRIPTION
オプションの要素であり、変数の説明を最大 256 文字で指定します。説明を変数ラベルとして割り当てる例を次に示します。

```
<DESCRIPTION> Story link </DESCRIPTION>
```

LENGTH
XML データの文字変数の最大格納フィールド長を指定します。この値は、SAS データセット内の各変数の値を格納する場合に使用されるバイト数を意味します。この属性値の範囲は、1～32,767 になります。インポート処理時に、この属性に指定した最大長の文字列が XML ドキュメントから読み込まれた後、オブザベーションバッファに転送されます。例: **<LENGTH> 200 </LENGTH>**.

制限事項 LENGTH 要素は数値データでは無効です。

要件 STRING データ型として定義されているデータの場合、LENGTH 要素が必須です。

ヒント LENGTH 要素を使用すると、長いフィールドを切り詰めることができます。多バイト文字列が指定された長さを超えた場合、バイト境界でなく文字境界で切り詰められます。

PATH syntax="*type*"

場所パスとは、現在の変数の値を取り出すには、XML ドキュメント内のどの位置を検索し、特定のタグにアクセスする必要があるかを XML Engine に伝えるものです。また、場所パスは、場所パス形式により指定される機能を実行して変数の値を取り出すよう XML Engine に伝えます。サポートされている XPath 形式を指定すると、生成される SAS データセットに要素や属性を個別に含めることができます。

syntax="*type*"

この属性は、場所パス内に記述する構文のタイプを指定します。この構文は、W3C 規格に準拠した有効な XPath 構成になります。XML Engine がサポートしている XPath 形式では、生成される SAS データセットに要素や属性を個別に含めることができます。

デフォルト XPath

要件 この値は、XPath または XPathENR のどちらかでなければなりません。

XML 名前空間を NAMESPACES 要素で定義する場合、構文のタイプを XPathENR (埋め込み型の名前空間参照を持つ XPath)として指定する必要があります。これは、この構文が XPath 規格とは異なるためです。例: **syntax="XPathENR"**。

PATH 要素の場所パスを指定するには、次の形式のどちらかを使用します。

注意:

これらの形式は、どちらも XML Engine がサポートしている XPath 形式のみになります。 それ以外の有効な W3C 形式を使用した場合、予測できない結果が引き起こされることがあります。

element-form

指定の要素から PCDATA (パーシング済みの文字データ)を選択します。次に示す各種の要素形式を使うと、指定の要素からの選択、指定の属性に基づく指定の要素からの条件付き選択、位置機能を使用した要素の特定オカレンスに基づく指定の要素からの条件付き選択が行えます。

```
<PATH> /LEVEL/ITEM </PATH>
<PATH> /LEVEL/ITEM[@attr="value"] </PATH>
<PATH> /LEVEL/ITEM[position()=n]|[n] </PATH>
```

次の例では、要素形式の使い方を示します。使用例については、“PATH 要素に場所パスを指定する” (56 ページ)を参照してください。

- 次の場所パスは、CONFERENCE 要素を検出するまで XML マークアップをスキャンするよう XML Engine に指示します。XML Engine は、開始タグ<CONFERENCE>と終了タグ</CONFERENCE>の間に記述されている値を取り出します。

```
<PATH> /NHL/CONFERENCE </PATH>
```

- 次の場所パスは、founded=属性の値が 1993 である TEAM 要素を検出するまで XML マークアップをスキャンするよう XML Engine に指示します。XML Engine は、開始タグ<TEAM>と終了タグ</TEAM>の間に記述されている値を取り出します。

<PATH> /NHL/CONFERENCE/DIVISION/TEAM[@founded="1993"] </PATH>

- 次の場所パスは、位置機能を使用して、TEAM 要素の 5 番目のオカレンスを検出するまで XML マークアップをスキャンするよう XML Engine に指示します。XML Engine は、開始タグ<TEAM>と終了タグ</TEAM>の間に記述されている値を取り出します。

<PATH> /NHL/CONFERENCE/DIVISION/TEAM[position()=5] </PATH>

位置機能には次の短いバージョンを使用することができます。

<PATH> /NHL/CONFERENCE/DIVISION/TEAM[5] </PATH>

attribute-form

属性から値を選択します。次の属性形式を使うと、指定の属性からの選択や、別の属性の値に基づく指定の属性からの条件付き選択が行えます。

<PATH> /LEVEL/ITEM/@attr </PATH>
<PATH> /LEVEL/ITEM/@attr[@attr2="value"] </PATH

次の例では、属性形式の使い方を示します。使用例については、“PATH 要素に場所パスを指定する” (56 ページ)を参照してください。

- 次の場所パスは、TEAM 要素を検出するまで XML マークアップをスキャンするよう XML Engine に指示します。XML Engine は abbrev=属性から値を取り出します。

<PATH syntax="XPath"> /NHL/CONFERENCE/DIVISION/TEAM/@abbrev </PATH>

- 次の場所パスは、TEAM 要素を検出するまで XML マークアップをスキャンするよう XML Engine に指示します。XML Engine は、abbrev=属性の値が ATL である場合に founded=属性から値を取り出します。これら 2 つの属性は、同じ要素に対して指定されたものでなければなりません。

<PATH> /NHL/CONFERENCE/DIVISION/TEAM/@founded[@abbrev="ATL"] </PATH>

要件

PATH が必須であるかどうか、または許可されているかどうかは、COLUMN 要素の class="ORDINAL"属性により決定されます。class="ORDINAL"属性を指定しない場合(デフォルト)、PATH 要素が必須となり、INCREMENT-PATH、DECREMENT-PATH、RESET-PATH 要素は許可されません。class="ORDINAL"属性を指定した場合、PATH 要素は許可されず、INCREMENT-PATH または DECREMENT-PATH 要素が必須となり、RESET-PATH 要素はオプションになります。

XML 名前空間を NAMESPACES 要素で定義する場合、定義しようとしている要素に先立って、場所パスに ID 番号を含める必要があります。ID 番号は中かっこで囲みます。例: **<PATH syntax="XPathENR">/Table/Hurricane/{1}Month</PATH>**。“XMLMap に名前空間要素を挿入する” (58 ページ)を参照してください。

XPath 構成は、XML 構造の各要素に関して UNIX ライクなパス記述を配置する公式規格です。XPath の構文では大文字小文字が区別されます。たとえば、要素タグ名が大文字である場合、そのタグ名は場所パスでも大文字で記述する必要があります。同様に、要素タグ名が小文字である場合、そのタグ名は場所パスでも小文字で記述する必要があります。すべての場所パスは、ルート包含要素(スラッシュ'/'で表される)で始まる

か、または"任意の親"バリアント(ダブルスラッシュ'//'で表される)で始まる必要があります。W3C で規定されているその他の形式は、現時点ではサポートされていません。

例 “PATH 要素に場所パスを指定する” (56 ページ)

INCREMENT-PATH syntax="*type*" beginend="BEGIN | END"

COLUMN 要素の class="ORDINAL 属性を指定した場合に確立される、カウンタ変数用の場所パスを指定します。この場所パスは、入力データ内のどの箇所で、カウンタ変数の累積値を 1 インクリメントするかを XML Engine に伝えます。

syntax="*type*"

オプション属性であり、場所パス内に記述する構文のタイプを指定します。この構文は、W3C 規格に準拠した有効な XPath 構成になります。XML Engine がサポートしている XPath 形式では、生成される SAS データセットに要素や属性を個別に含めることができます。例: **syntax="XPath"**。

デフォルト XPath

要件 この値は、XPath または XPathENR のどちらかでなければなりません。

XML 名前空間を NAMESPACES 要素で定義する場合、構文のタイプを XPathENR (埋め込み型の名前空間参照を持つ XPath)として指定する必要があります。これは、この構文が XPath 規格とは異なるためです。例: **syntax="XPathENR"**。

beginend="BEGIN | END"

オプション属性であり、要素の開始タグを検出した時点または要素の終了タグを検出した時点で、処理を停止するよう指定します。

デフォルト BEGIN

要件 XML 名前空間を NAMESPACES 要素で定義する場合、定義しようとしている要素に先立って、場所パスに ID 番号を含める必要があります。ID 番号は中かっこで囲みます。例: **<INCREMENT-PATH syntax="XPathENR">/Table/Hurricane/{1}Month</INCREMENT-PATH>**。

XPath 構成は、XML 構造の各要素に関して UNIX ライクなパス記述を配置する公式規格です。XPath の構文では大文字小文字が区別されます。たとえば、要素タグ名が大文字である場合、そのタグ名は場所パスでも大文字で記述する必要があります。同様に、要素タグ名が小文字である場合、そのタグ名は場所パスでも小文字で記述する必要があります。すべての場所パスは、ルート包含要素(スラッシュ'/'で表される)で始まるか、または"任意の親"バリアント(ダブルスラッシュ'//'で表される)で始まる必要があります。W3C で規定されているその他の形式は、現時点ではサポートされていません。

変数がカウンタ変数でない場合、PATH 要素が必須となり、INCREMENT-PATH 要素および RESET-PATH 要素は許可されません。変数がカウンタ変数である場合、PATH 要素は許可されず、INCREMENT-PATH または DECREMENT-PATH 要素のどちらかが必須となります。

例 “生成された数値キーを含むキーフィールドを挿入する” (42 ページ)

RESET-PATH syntax="*type*" beginend="BEGIN | END"
COLUMN 要素の class="ORDINAL 属性を指定した場合に確立される、カウンタ変数用の場所パスを指定します。この場所パスは、XML ドキュメント内のどの箇所で、カウンタ変数の累積値をゼロにリセットするかを XML Engine に伝えます。

syntax="*type*"
オプション属性であり、場所パス内に記述する構文のタイプを指定します。この構文は、W3C 規格に準拠した有効な XPath 構成になります。XML Engine がサポートしている XPath 形式では、生成される SAS データセットに要素や属性を個別に含めることができます。例: **syntax="XPATH"**。

デフォルト XPath

要件 この値は、XPath または XPathENR のどちらかでなければなりません。

XML 名前空間を NAMESPACES 要素で定義する場合、構文のタイプを XPathENR (埋め込み型の名前空間参照を持つ XPath)として指定する必要があります。これは、この構文が XPath 規格とは異なるためです。例: **syntax="XPathENR"**。

beginend="BEGIN | END"
オプション属性であり、要素の開始タグを検出した時点または要素の終了タグを検出した時点で、処理を停止するよう指定します。

デフォルト BEGIN

要件 変数がカウンタ変数でない場合、RESET-PATH は使用できません。変数がカウンタ変数である場合、RESET-PATH の指定はオプションです。

XML 名前空間を NAMESPACES 要素で定義する場合、定義しようとしている要素に先立って、場所パスに ID 番号を含める必要があります。ID 番号は中かっこで囲みます。例: **<RESET-PATH syntax="XPathENR">/Table/Hurricane/{1}Month</RESET-PATH>**。

XPath 構成は、XML 構造の各要素に関して UNIX ライクなパス記述を配置する公式規格です。XPath の構文では大文字小文字が区別されます。たとえば、要素タグ名が大文字である場合、そのタグ名は場所パスでも大文字で記述する必要があります。同様に、要素タグ名が小文字である場合、そのタグ名は場所パスでも小文字で記述する必要があります。すべての場所パスは、ルート包含要素(スラッシュ'/'で表される)で始まるか、または"任意の親"バリアント(ダブルスラッシュ'//'で表される)で始まる必要があります。W3C で規定されているその他の形式は、現時点ではサポートされていません。

DECREMENT-PATH syntax="*type*" beginend="BEGIN | END"
COLUMN 要素の class="ORDINAL 属性を指定した場合に確立される、カウンタ変数用の場所パスを指定します。この場所パスは、入力データ内のどの箇所で、カウンタ変数の累積値を 1 デクリメントするかを XML Engine に伝えます。

syntax="*type*"
オプション属性であり、場所パス内に記述する構文のタイプを指定します。この構文は、W3C 規格に準拠した有効な XPath 構成になります。XML Engine がサポートしている XPath 形式では、生成される SAS データセットに要素や属性を個別に含めることができます。次に例を示します。 **syntax="XPath"**

デフォルト XPath

要件 この値は、XPath または XPathENR のどちらかでなければなりません。

XML 名前空間を NAMESPACES 要素で定義する場合、構文のタイプを XPathENR (埋め込み型の名前空間参照を持つ XPath)として指定する必要があります。これは、この構文が XPath 規格とは異なるためです。次に例を示します。 **syntax="XPathENR"**

beginend="BEGIN | END"
オプション属性であり、要素の開始タグを検出した時点または要素の終了タグを検出した時点で、処理を停止するよう指定します。

デフォルト BEGIN

要件 変数がカウンタ変数でない場合、DECREMENT-PATH は使用できません。
変数がカウンタ変数である場合、DECREMENT-PATH または INCREMENT-PATH のどちらかが必須となります。

XML 名前空間を NAMESPACES 要素で定義する場合、定義しようとしている要素に先立って、場所パスに ID 番号を含める必要があります。ID 番号は中かっこで囲みます。次に例を示します。 **<DECREMENT-PATH syntax="XPathENR">/Table/Hurricane/{1}Month</DECREMENT-PATH>**

XPath 構成は、XML 構造の各要素に関して UNIX ライクなパス記述を配置する公式規格です。XPath の構文では大文字小文字が区別されます。たとえば、要素タグ名が大文字である場合、そのタグ名は場所パスでも大文字で記述する必要があります。同様に、要素タグ名が小文字である場合、そのタグ名は場所パスでも小文字で記述する必要があります。すべての場所パスは、ルート包含要素(スラッシュ'/'で表される)で始まるか、または"任意の親"バリアント(ダブルスラッシュ'//'で表される)で始まる必要があります。W3C で規定されているその他の形式は、現時点ではサポートされていません。

推奨資料

本書の内容に関連する参考文献として推奨する文献を次に示します。

- *The Little SAS Book: A Primer*
- *SAS Viya ステートメント: リファレンス*
- *SAS Viya Data Set Options: Reference*
- *SAS Viya 各国語サポート: リファレンスガイド*
- お使いの動作環境向けの SAS ドキュメント
- Base SAS ソフトウェアに関する情報: support.sas.com/base
- XML (Extensible Markup Language)に関する情報:www.w3.org/XML

SAS 刊行物の一覧については、 sas.com/store/books から入手できます。必要な書籍についての質問は SAS 担当者までお寄せください:

SAS Books
SAS Campus Drive
Cary, NC 27513-2414
電話: 1-800-727-0025
ファクシミリ: 1-919-677-4444
メール: sasbook@sas.com
Web アドレス: sas.com/store/books

用語集

ASCII
参照項目: 情報交換用米国標準コード.

Document Object Model (DOM)
プラットフォームや言語に依存せずに、ドキュメント内のオブジェクトを使用して表示や対話を行う方法。通常、DOM は HTML や XHTML や XML ドキュメントなどに適用されます。

DOM
参照項目: Document Object Model.

DTD
参照項目: 文書型定義.

EBCDIC (拡張 2 進化 10 進コード)
IBM のメインフレームやミッドレンジコンピュータ上でデータを表現するために使用されるシングルバイトおよびマルチバイトエンコーディングのファミリ。

FTP
参照項目: ファイル転送プロトコル(File Transfer Protocol).

libref (ライブラリ参照名) (ライブラリ参照名)
SAS ライブラリに一時的に関連付けられる名前。たとえば、MYLIB.MYFILE という名前の場合、MYLIB がライブラリ参照名であり、MYFILE はその SAS ライブラリ内にあるファイルになります。 *関連項目:* SAS ライブラリ.

ODS テンプレート (グラフィックステンプレート)
出力のフォーマット時に出力がどのように表示されるかを記述したもの。ODS テンプレートは、コンパイル済みエントリとして、テンプレートストア(項目ストア)内に保存されます。一般的なテンプレートタイプとしては、STATGRAPH、STYLE、CROSSTABS、TAGSET、TABLE などがあります。

PCDATA
参照項目: 構文解析済み文字データ.

SAS 出力形式 (出力形式)
データの種類に基づいてデータ値を書き出したり表示したりするのに使用される SAS 言語要素の種類。データの種類には、数値、文字、日付、時間、タイムスタンプがあります。

SAS データセット (データセット)
ネイティブ SAS ファイル形式のコンテンツを含むファイル。SAS データセットには、SAS データファイルと SAS データビューの 2 種類があります。

SAS データビュー (データビュー)
SAS データセットの一種。他のファイルからデータ値を取り込みます。SAS データビューには、変数(列)のデータ型や長さなどの情報と、他の SAS データセットから、または SAS 以外のソフトウェアベンダーのファイル形式でデータを格納しているファイルからデータ値を取り出すのに必要となる情報のみが含まれています。

SAS データファイル
SAS データセットの一種。データ値と、データに関連付けられたディスクリプタ情報を含みます。ディスクリプタ情報には、変数のデータ型や長さ、データの作成に使用されたエンジンの名前などが含まれています。 *関連項目:* SAS データセット, SAS データビュー.

SAS 入力形式 (入力形式)
データの種類に基づいてデータ値を読み込むのに使用される SAS 言語要素の種類。データの種類には、数値、文字、日付、時間、タイムスタンプがあります。

SAS 変数 (変数)
SAS データセット内または SAS データビュー内の列。各変数のデータ値は、すべてのオブザベーション(行)の単一の特性を表します。

SAS ライブラリ
SAS システムにより定義され、認識され、アクセスされる 1 つ以上のファイルであり、1 つの単位として参照および保存されるものです。各ファイルは特定の SAS ライブラリのメンバになります。

SAX
参照項目: Simple API for XML.

Simple API for XML (SAX)
XML-DEV グループのメンバーによって XML ドキュメント用に開発された、イベントベースの順次アクセス型パーサー用 API です。SAX は、XML ドキュメントからデータを読み取るためのメカニズムを提供します。このメカニズムは、Document Object Model (DOM)が提供するものの代わりとなります。

URI (ユニフォームリソース識別子)
World Wide Web 上でファイル、イメージ、サービスなどのリソースを識別するために使用される文字列。URL は URI の一種です。 *関連項目:* ユニフォームリソースロケータ.

URL
参照項目: ユニフォームリソースロケータ.

UTC
参照項目: 協定世界時.

W3C
参照項目: World Wide Web Consortium.

World Wide Web Consortium (W3C)
World Wide Web の長期的な成長を保証するためにオープン標準を開発している国際的なコミュニティ。

XML
参照項目: 拡張マークアップ言語.

XML LIBNAME エンジン
XML ドキュメントを処理する SAS エンジン。XML Engine は、SAS ファイルに固有の出力形式を XML マークアップに変換することにより、SAS データセットから XML ドキュメントをエクスポートします。また、XML マークアップを SAS データセットに変換して、外部 XML ドキュメントをインポートします。

XMLMap ファイル
XML ドキュメントの変換方法を SAS XML LIBNAME Engine に伝える XML タグを含んでいるファイル。

XML スキーマ
XML ドキュメントの構造、内容、セマンティクスを定義します。XML スキーマは、通常、ドキュメント型の構造と内容に関する制約によって表現されます。

XML 名前空間
拡張マークアップ言語(XML)ドキュメント内で使用される要素名や属性名を、URI 参照により識別される名前空間を関連付けることによって、それらの要素名や属性名を修飾する簡単な方法を提供する W3C 規格です。XML 名前空間は、要素名の衝突を回避します。たとえば、ある XML ドキュメントに CUSTOMER 要素と PRODUCT 要素が含まれており、両方の要素が 1 つのネストされた ID 要素を含んでいる場合、XML 名前空間は、それぞれのネストされた ID 要素を一意なものにします。

XML パス言語 (XPath)
XML ドキュメント内のアイテムの検索および処理方法を記述するクエリ言語。XPath 指定は、XML 構造の各要素に対して、UNIX ライクなパス記述を割り当てます。

XPath
参照項目: XML パス言語.

XPathENR
参照項目: 組み込み名前空間参照を含む XPath.

エンコーディング
コード化した文字セットをコード値にマッピングすること。

オブザベーション
SAS データセットの行。オブザベーション内のデータ値はすべて、顧客や状態などの単一のエンティティに関連付けられます。各オブザベーションには、各変数に対する 1 つのデータ値か欠損値インジケータのどちらかが含まれます。

階層データベース
複数のセグメントからなるツリー構造として構成されるデータベース。DL/I データベースは階層的なデータ構造を持ちます。

拡張 2 進化 10 進コード
参照項目: EBCDIC.

拡張マークアップ言語 (XML)
マークアップ言語の一つであり、内容、意味、使用法などに関する情報をタグ付けすることにより情報を構造化します。構造化された情報は、内容(ワードや数字など)と内容の役割を示すインジケータから構成されます。たとえば、セクション見出しに含まれている内容は、データベーステーブルに含まれている内容とは異なる意味を持っています。

キーフィールド
参照項目: シーケンスフィールド.

協定世界時 (UTC)
世界協定で決められた時刻標準。コンピュータやネットワークサービスで時計や時間を制御するために使用されます。具体的には、英国グリニッジを通る子午線上のローカル時刻(グリニッジ標準時)とほぼ同じものになります。現在では、グリニッジ標準時よりも協定世界時という用語の方がよく使われます。

組み込み名前空間参照を含む XPath (XPathENR)
XML 名前空間をサポートするためにエンジンにより使用される XMLMap 要素。

グラフィックステンプレート
参照項目: ODS テンプレート.

構文解析済み文字データ (PCDATA)
混合コンテンツ(任意の順番で任意の回数出現する文字データと子要素を含む要素)を指定するために XML で使用されるデータ定義。

シーケンスフィールド (キーフィールド)
データベース内のセグメントを識別し、同セグメントへのアクセスを提供するフィールド。シーケンスフィールドには、レコードのキーが含まれます。これは、キーに基づいて並べられたデータセットの各レコード内の同じ位置に存在します。

出力形式
参照項目: SAS 出力形式.

情報交換用米国標準コード (ASCII)
各種のコンピュータシステムで利用できる 128 種類の基本的な文字セットを提供する 7 ビットのエンコーディング規格。この規格は、アルファベットの大文字小文字、句読点、数字(0〜9)、制御文字を符号化します。この 128 種類の文字セットは、これ以外のほとんどのエンコーディングにも含まれています。 *関連項目:* EBCDIC.

タグセット
SAS 出力形式から特定のタイプのマークアップ言語出力を作成する方法を定義するテンプレート。タグセットは、ハイパーテキストマークアップ言語(HTML)、拡張マークアップ言語(XML)、LaTeX のようなマークアップ出力を生成します。 *関連項目:* マークアップ言語.

データセット
参照項目: SAS データセット.

データビュー
参照項目: SAS データビュー.

入力形式
参照項目: SAS 入力形式.

ファイル参照
参照項目: ファイル参照名(fileref).

ファイル参照名(fileref) (ファイル参照)
外部ファイルまたはディレクトリやフォルダなどの集合保存場所に対して一次的に割り当てられる名前のこと。SAS システムでは、ファイル参照名を使用してファイルや保存場所を識別します。 *関連項目:* libref (ライブラリ参照名).

ファイル転送プロトコル(File Transfer Protocol) (FTP)
ネットワークを介したコンピュータ間でのファイル転送に使用される通信プロトコル。

文書型定義 (DTD)
マークアップ言語(SGML、XML、HTML) の文書型を定義するマークアップ宣言の集合。文書型定義は、ドキュメントを表示または印刷するアプリケーションによって、特定のタグがどのように解釈されるかを定義します。

変数
参照項目: SAS 変数.

マークアップ言語
レイアウトや特定のコンテンツを定義するためにテキスト内に埋め込まれているコードの集合。

メタデータ
捕捉されアーカイブ化されているデータへのアクセスを容易にするために、データベース内に保存され監理されているデータに関する記述データ。

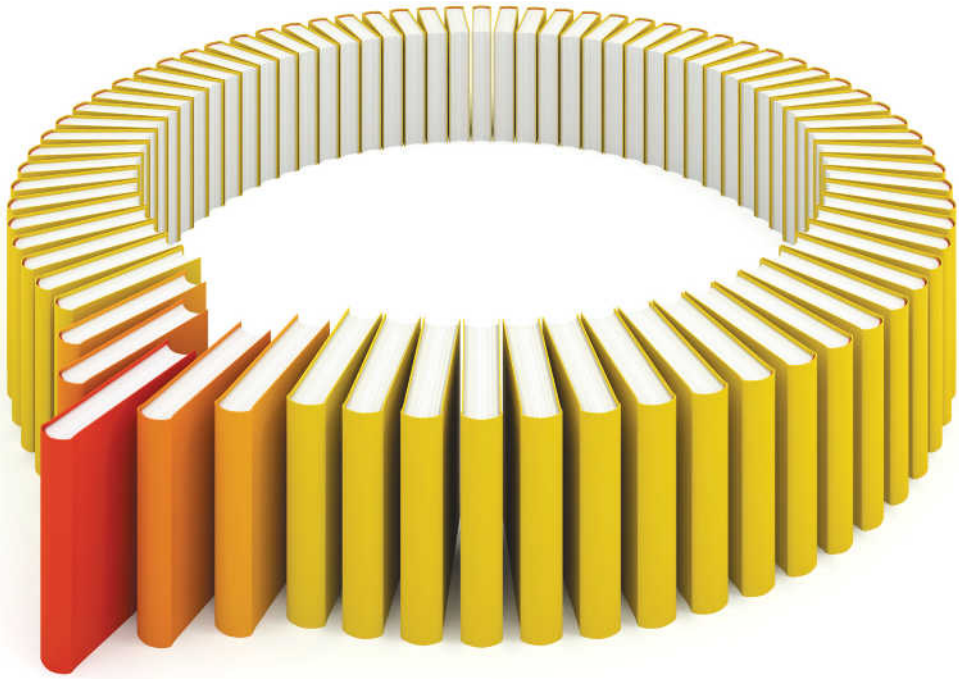
ユニフォームリソース識別子
参照項目: URI.

ユニフォームリソースロケータ (URL)
インターネット上やイントラネット上のリソースにアクセスまたは識別するために、Web ブラウザやその他のソフトウェアアプリケーションにより使用される文字列。リソースは、Web ページ、電子イメージファイル、オーディオファイル、JavaServer ページ、その他の任意の種類の電子オブジェクトなどです。完全な形式の URL には、当該リソースのアクセスに使用する通信プロトコルや、当該リソースのディレクトリパスおよびファイル名が含まれています。

ライブラリ参照名
参照項目: libref (ライブラリ参照名).

キーワード

あ



Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

support.sas.com/bookstore
for additional books and resources.