WSDK

WSDK:
WEB サービスのインターオペラビリティ（相互運用性）について

Version 1.0 November 17, 2005

Table of Contents

1 イントロダクション

1.1 WEB サービス

WEB サービスは SOAP 仕様によって規定されています。それらは WSDL の仕様に沿ったファイル（WSDL ファイル）によって定義されます。WSDL ファイルは、型とエレメントの定義のために XML スキーマを利用します。

XML スキーマ（以後 XS という）は広範囲で、XML を使う際の完全な範囲で記述するための複雑な仕様ですが、多くの先進的な仕様は、実際には WEB サービスアプリケーションであまり使われません。

メッセージコンテント XSD と WSDL を使った、標準の XS の特性である「プロフィール」のサブセットを使うことは、WEB サービスの相互運用性についてのグッドプラクティスと考えられます。

そのとき、その定義によってサーバーとクライアントの実装が（さまざまなプログラム言語とツールで）開発されます。しかし、多くのツールは「たとえば、特定のプログラム言語で実装されたプログラムから、

WSDL と XS の定義を抽出するといった「リバースメソッド」を要求します。このアプローチは WEB サービスのすべての末端が同じ定義を利用するときに便利ですが、XML スキーマの型のマッピングに関する、相互運用性の問題にぶつかります。

1.2 WS-I スタンダード

WSDK は「WS-I Basic Profile（http://www.ws-1.org/Profiles/BasicProfile-1.0-2004-04-16.html）に準拠しています。しかし、いくつかの特徴はいまだ完全には実装されていません。

WS-I は、WEB サービス間での相互運用性を改良するためのものです。WS-I は以下のように注記しています。

「個別のサービスの相互運用性を完全に保障することは不可能です。しかし、プロファイルは、日々、実装における経験が明らかにしてきた多くの共通的な問題を解決します。」

WS-I の WEB サイトは、サービスが「Basic Profile」に適合しているかどうかを評価するためのサンプルアプリケーションや、いくつかのテストツールを公開しています、

多くの相互運用性の問題は特殊な XML スキーマの型を扱うことから発生します。この「WSDK：WEB サービスの相互運用性について」の資料は、どうやって Curl WSDK 1.0 の WSDL パッケージが XML スキーマのタイプを Curl 言語として表現するかの要約です。

2 XML スキーマと対応する型

2.1 単純型 (Simple Types)

すべてのプリミティブな単純型はサポートされています。XMLSimpleValue は直接対応する Curl の型がないような XML スキーマの型にたいして使用されます。また、XMLSimpleValue は基本的な String 値、および割り当てられた XML Schema データ型式へのアクセスを与えます。

XML Type	Curl Type
xs:string	String
xs:boolean	bool
xs:int	int
xs:double	double
xs:float	float
xs:long	int64
xs:short	int16
xs:byte	int8
xs:base64Binary	{Array-of uint8}
xs:hexBinary	{Array-of uint8}
xs:unsignedShort	uint16
xs:unsignedByte	uint8
xs:integer	int64
xs:positiveinteger	int64
xs:negativeinteger	int64
xs:nonPositiveinteger	int64
xs:nonNegativeinteger	int64
xs:unsignedInt	int64
xs:unsignedLong	int64
xs:decimal	double
xs:dateTime	DateTime
xs:anyURI	Url
xs:QName	XDMName
xs:date	XMLSimpleValue
xs:time	XMLSimpleValue
xs:duration	XMLSimpleValue
xs:gYear	XMLSimpleValue
xs:gYearMonth	XMLSimpleValue
xs:gMonth	XMLSimpleValue
xs:gMonthDay	XMLSimpleValue
xs:Day	XMLSimpleValue
xs:normalizedString	XMLSimpleValue
xs:token	XMLSimpleValue
xs:language	XMLSimpleValue
xs:Name	XMLSimpleValue
xs:NCName	XMLSimpleValue
xs:ENTITY	XMLSimpleValue
xs:ENTITIES	XMLSimpleValue
xs:NMTOKEN	XMLSimpleValue
xs:NMTOKENS	XMLSimpleValue
xs:ID	XMLSimpleValue
xs:IDREF	XMLSimpleValue
xs:IDREFS	XMLSimpleValue

生成された単純型は以下の内容をフォローします。

- 列挙対は Curl の「define-enum」と同様に表現されます
- 制限（restriction）は基本的なプリミティブ型の別名として、表現されます
- リストは基本的な型のパラメタを持つ配列として表現されます
- ユニオンは「any」の別名として表現されます

2.2 複雑型（Complex Types）

一般的に、特定のエレメントと属性に対応するフィールドと一緒に複雑型はクラスとして表現されます。それぞれのフィールドの型はメンバ（エレメントや属性）の型を基本とします。メンバの扱いは cardinality[基数]とコンテントのモデルに依存します。

しかし、いくつかのケースでは、特定のストラクチャは直接対応するクラスを持たず、異なる表現形式（XDMElement　や HashTable など）が利用されます。これらのケースは以降の章で示します。

2.2.1 基数（Cardinality）

必須エレメント[minOccurs=1 maxOccurs=1]のとき、non-nullable なフィールドとして表現されます。

任意エレメント[minOccurs=0 maxOccurs=1]のとき、nullable なフィールドとして表現されます。

繰り返しエレメントが[maxOccurs>1]のとき、パラメータ化された配列として表現されます。

「nilliable」として指定されたエレメントはいつも nullable のフィールドとして表現されます。

2.2.2 複雑コンテント（Complex Content）

指定されたエレメントは、「all」と「seaquence」モデルグループのなかのクラスの中のフィールドとして表現されます。

指定されたエレメントは、コンテナをあらわすために、「choice」モデルグループの中の「HashTable」を利用したものです。

「group」の定義を参照することで、間接的に指定された要素は、構成（constituent）エレメントに拡張されます。

「any」を使って指定された任意のエレメントは、コンテナを表現する「XDMElement」になります。

属性もフィールドとして表現されます。

「attributeGroup」の定義を参照することで、間接的に指定された属性は、構成（constituent）属性に拡張されます。

「anyAttribute」の定義を参照することで、間接的に指定された任意の属性は、コンテナを表現する「XDMElement」になります。

混合コンテント（Mixed content）は「XDMElement」になります。

拡張による型は指定されたベースに対応するクラスからインヘリットされる定義のクラスとして生成されます。

制限による型は、通常、XDMElement として生成されます。定義がコード化された配列について説明するのに、慣習上使用されるパターンに合うとき、それはパラメタ化された配列として表現されます。.

2.2.3 単純コンテント（Simple Content）

Simple content は（単純内容）はよく「atomic」なエレメントだと記述されます。それはただ、テキストのエレメントだけ（属性も）含むことができるというものです。「content」という名前でテキスト値は表現されます。

2.3 グローバル定義（Global Definitions）

グローバルに名前付けられたエレメントとして構造が指定されることを XML スキーマは許します。これらは型の定義と同様に扱われます。

グローバルに名前付けられた属性は、シングルメンバと一緒に、属性のグループと同様に扱われます。

2.4 ローカル定義（Local Definitions）

グローバルタイプ以外で構造がローカルとして指定されることを XML スキーマは許します。そのような構造の表現がクラスの定義に対応するとき、エレメントの名前に基づいた名前が利用されます。

2.5 スキーマ構成（Schema Composition）

スキーマもしくは WSDL ファイルは、「import」や「include」を使って他のスキーマファイルから別の定義を取り込むことがあります。これらはトップレベルのスキーマ定義と同様に処理されます。実際には、よく知られている SOAP/WSDL スキーマ定義も明確にはインポートされていません。.

- http://www.w3.org/2001/XMLSchema
- http://schemas.xmlsoap.org/wsdl/
- http://schemas.xmlsoap.org/soap/encoding/

- http://www.w3.org/XML/1998/namespace

2.6 名前（Names）

正当な XML 名は正当な Curl 名と一致しない場合があります。XML の名前空間や各プログラム言語の間では、許されたキャラクターセットが異なるためです。

WEB サービス定義が Curl に吸収されるとき、正当な Curl の名前は基本的な Curl 規約に準拠します。名前空間プリフィックス、もしくはニューメリックサフィックスを追加することで名前の競合はチェックされ、解決されます。

2.7 SOAP エンコード（SOAP encoding）

SOAP は、コード化に従って操作入力と出力が、明白な XS タイプの使用なしで解釈されるのを許容します。 この'コード化された'用法は WS-I 相互運用性ガイドラインで推奨されていません。上記で説明された、より明白な'リテラル'な用法を使用すべきです。

'コード化された'サービスに使用される仕様は以下の通りです。SOAP 仕様は'encodingStyle'属性の値によって特定されたスタイルをコード化する arbitary を考慮します。実際には、いつも利用される規定のエンコードスタイルがあり、いかの URL で規定されています。

http://schemas.xmlsoap.org/soap/encoding/

メッセージ要素は、それらのタイプについて説明するのに'xsl:type'という属性を使用します。'xsi:nil'が真の状態でマークされたメッセージ要素はヌルとして扱われます。 配列の要素は'wsi:type'、'soap:arrayType'のために、ディメンションと連結された要素を含む値と一緒に、'soapenc:Array'を利用します。（また、仕様には、あいまいな配列と標準的でないオフセットのための条項がありますが、これらは、めったに使用されなません。WSDK はこれらをサポートしません。）

メッセージ要素("multiref")は共有される可能性があります:それらはユニークな ID 属性と一緒に、一度だけ定義され、ref 属性を通してどこからでも参照されます。元の要素はメッセージの中に配置済みか、メッセージ部の外に独立しています。いくつかのサービスやツールは、プリミティブな値や、一度現れるだけの値のために"multiref"エンコードを利用します。

2.8 サービスメソッド署名（Service Method Signatures）

WSDL のサービスポートに対応する Curl のサービスクラスはそれぞれの SOAP 処理ごとのメソッドを定義します。また、各メソッドには同期、非同期の二つをそれぞれ用意します。それらのメソッドや属性、戻り値に関するシグネチャは、WSDL ファイルで定義されている「入力（input）」と「出力（output）」のメッセージから生成されます。そのメッセージに関する仕様は処理の形式が「rpc」か「document」によって異なります。「document」形式の場合、「入力（input）」と「出力（output）」のメッセージは、型の中の「要素」の定義を参照するひとつの「パート」に含まれるはずです。「rpc」形式の場合、メッセージは複数のパートに分かれる可能性があります。各「パ

ート」の入力メッセージは属性を表し、各「パート」の出力は返り値を表すような場合です。

SOAP メッセージはヘッダーを含むことができます。ヘッダが指定されたとき、入力ヘッダーはキーワード引数として表現され、出力ヘッダーは追加的戻り値として表現されます。

3 WSDK インターオペラビリティ

3.1 既知の問題

以下は既知の問題です。

- 繰り返しのエレメントは必ず、自身のコンテナにネストされていなければなりません

これは WSDK1.0 の制限であり、本来の XML スキーマにはモデルグループの中のエレメントの基数についての制限はありません。しかし、このそれ自身のコンテナの中でネストして繰り返されるエレメントは、WEB サービスのインターオペラビリティのために、グッドプラクティスだとみなされます。

- エレメントと型のために使われる名前で、同じものは許されません

XML スキーマはエレメントとタイプの名前を区別する仕様であり、これは WSDK1.0 特有の制限です。しかしエレメントと型の名前を別にすること、それらを区別する命名規則を利用する共通のプラクティスです。

3.2 XS の対応範囲

以下の項目は WEB サービス定義ではめったに利用されません。これらにはインターオペラビリティの問題があり、WSDK では予期しない振る舞いをします。

- xs:union
- xs:choice
- xs:list
- xs:attributeGroup
- xs:anyAttribute
- xs:any

同様に、SOAP ヘッダーはあまり利用されません。そのため、サービスエンプロイングヘッダーはまず間違いなくインターオペラビリティの問題にぶつかります。

3.3 問題調査のための情報

WEB サービスインターオペラビリティの問題に関する調査、報告の際、事象の解明には、以下の情報が役立ちます。

1. WSDL ファイルとその参照ファイル（include と import）
2. メッセージのリクエストとレスポンスのサンプル。これらは Curl の HTTP モニターで取得できます。

4 WSDL の例

4.1 XS の Curl での表現形式

4.1.1 構成

4.1.1.1 XS

```
<xs:complexType name="ResultElement">
 <xs:all>
   <xs:element name="summary" type="xs:string"/>
   <xs:element name="URL" type="xs:string"/>
   <xs:element name="snippet" type="xs:string"/>
   <xs:element name="title" type="xs:string"/>
   <xs:element name="cachedSize" type="xs:string"/>
   <xs:element name="relatedInformationPresent" type="xs:boolean"/>
   <xs:element name="hostName" type="xs:string"/>
   <xs:element name="directoryCategory" type="typens:DirectoryCategory"/>
   <xs:element name="directoryTitle" type="xs:string"/>
 </xs:all>
</xs:complexType>
```

4.1.1.2 Curl

```
{define-class public open ResultElement
  field public summary:String = ""
  field public url:String = ""
  field public snippet:String = ""
  field public title:String = ""
  field public cached-size:String = ""
  field public related-information-present:bool
  field public host-name:String = ""
  field public directory-category:DirectoryCategory = {uninitialized-value-for-
type DirectoryCategory}
  field public directory-title:String = ""
}
```

4.1.2 配列[リテラル]

4.1.2.1 XS

```
<xs:complexType name="ArrayOfPlaceFacts">
 <xs:sequence>
   <xs:element minOccurs="0" maxOccurs="unbounded" name="PlaceFacts"
type="tnxs:PlaceFacts" />
 </xs:sequence>
</xs:complexType>
```

4.1.2.2 Curl

```
{let public constant ArrayOfPlaceFacts:ClassType =
     {Array-of PlaceFacts}
}
```

4.1.3 配列[エンコード]

4.1.3.1 XS

```
<xs:complexType name="ResultElementArray">
 <xs:complexContent>
   <xs:restriction base="soapenc:Array">
      <xs:attribute ref="soapenc:arrayType"
wsdl:arrayType="typens:ResultElement[]"/>
   </xs:restriction>
 </xs:complexContent>
</xs:complexType>
```

4.1.3.2 Curl

```
{let public constant ResultElementArray:ClassType =
     {Array-of ResultElement}
}
```

4.1.4 列挙体

4.1.4.1 XS

```
<xs:simpleType name="PlaceType">
 <xs:restriction base="xs:string">
   <xs:enumeration value="UnknownPlaceType" />
   <xs:enumeration value="AirRailStation" />
   <xs:enumeration value="BayGulf" />
   <xs:enumeration value="CapePeninsula" />
   <xs:enumeration value="CityTown" />
   <xs:enumeration value="HillMountain" />
   <xs:enumeration value="Island" />
   <xs:enumeration value="Lake" />
   <xs:enumeration value="OtherLandFeature" />
   <xs:enumeration value="OtherWaterFeature" />
   <xs:enumeration value="ParkBeach" />
   <xs:enumeration value="PointOfInterest" />
   <xs:enumeration value="River" />
 </xs:restriction>
</xs:simpleType>
```

4.1.4.2 Curl

```
{define-enum public PlaceType
    unknown-place-type = "UnknownPlaceType",
```

```
    air-rail-station = "AirRailStation",
    bay-gulf = "BayGulf",
    cape-peninsula = "CapePeninsula",
    city-town = "CityTown",
    hill-mountain = "HillMountain",
    island = "Island",
    lake = "Lake",
    other-land-feature = "OtherLandFeature",
    other-water-feature = "OtherWaterFeature",
    park-beach = "ParkBeach",
    point-of-interest = "PointOfInterest",
    river = "River"
}
```

4.1.5　匿名列挙体のリスト

4.1.5.1　XS

```
<xs:simpleType name="Themes">
 <xs:list>
   <xs:simpleType>
     <xs:restriction base="xs:string">
       <xs:enumeration value="Photo" />
       <xs:enumeration value="Topo" />
       <xs:enumeration value="Relief" />
     </xs:restriction>
   </xs:simpleType>
 </xs:list>
</xs:simpleType>
```

4.1.5.2　Curl

```
{let public constant Themes:ClassType = {Array-of String}}
```

4.1.6　単純コンテント (Simple Content)

4.1.6.1　XS

```
      <s:complexType name="category-type">
        <s:simpleContent>
          <s:extension base="s:string">
            <s:attribute name="id" type="s1:category-id-type" />
          </s:extension>
        </s:simpleContent>
      </s:complexType>
```

4.1.6.2　Curl

```
  field public content:String = ""
  field public id:Category-id-type
```

4.2　メソッド署名 (Method Signatures)

4.2.1　ドキュメント形式

```
<s:element name="GetPlaceList">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="placeName" type="s:string"
/>
      <s:element minOccurs="1" maxOccurs="1" name="MaxItems" type="s:int" />
```

```
      <s:element minOccurs="1" maxOccurs="1" name="imagePresence"
type="s:boolean" />
    </s:sequence>
  </s:complexType>
</s:element>
<s:element name="GetPlaceListResponse">
  <s:complexType>
    <s:sequence>
      <s:element minOccurs="0" maxOccurs="1" name="GetPlaceListResult"
                 type="tns:ArrayOfPlaceFacts" />
    </s:sequence>
  </s:complexType>
</s:element>

<wsdl:message name="GetPlaceListSoapIn">
  <wsdl:part name="parameters" element="tns:GetPlaceList" />
</wsdl:message>
<wsdl:message name="GetPlaceListSoapOut">
  <wsdl:part name="parameters" element="tns:GetPlaceListResponse" />
</wsdl:message>

<wsdl:operation name="GetPlaceListInRect">
  <wsdl:input message="tns:GetPlaceListInRectSoapIn" />
  <wsdl:output message="tns:GetPlaceListInRectSoapOut" />
</wsdl:operation>

<wsdl:operation name="GetPlaceList">
  <soap:operation style="document"
    soapAction="http://terraserver-usa.com/terraserver/GetPlaceList" />
  <wsdl:input> <soap:body use="literal" /> </wsdl:input>
  <wsdl:output> <soap:body use="literal" /> </wsdl:output>
```

4.2.2 RPC 形式

```
<message name="doSpellingSuggestion">
  <part name="key" type="xsd:string"/>
  <part name="phrase" type="xsd:string"/>
</message>

<message name="doSpellingSuggestionResponse">
  <part name="return" type="xsd:string"/>
</message>

<operation name="doSpellingSuggestion">
  <input message="typens:doSpellingSuggestion"/>
  <output message="typens:doSpellingSuggestionResponse"/>
</operation>

<operation name="doSpellingSuggestion">
  <soap:operation soapAction="urn:GoogleSearchAction"/>
  <input>
    <soap:body use="encoded" namespace="urn:GoogleSearch"
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
    <soap:body use="encoded" namespace="urn:GoogleSearch"
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
</operation>
```