「気象庁防災情報 XML フォーマット」スキーマとサンプル電文の動作検証について

「気象庁防災情報 XML フォーマット」仕様に付属しています各種スキーマとサンプル電文について、XML コンソーシアムの協力により、各種 XML 関連ソフトウェアにおける動作状況について検証していただきました。検証結果については、同仕様をご利用いただく際の参考とするべく、仕様とともに公開いたします。

【要旨】

各種ベンダー系、及びフリー系ソフトウェアにおいて、スキーマの読み込みとサンプル電文の妥当性検証、その他データバインディング等の検証において、良好な動作結果をご報告いただきました。

なお、一部フリー系ソフトウェアでは union 要素の処理で問題が発生しましたが、これについては回避方法も併せてご報告いただきました。

【検証内容と結果】

検証対象であるソフトウェアと動作環境、及び検証内容とその結果については別紙のとおりです。

なお、本検証結果については検証条件における動作結果であり、気象庁並びに各ソフトウェア開発元において、「気象庁防災情報 XML フォーマット」仕様の動作保証、及び完全対応を表明するものではありません。

【謝辞】

XML コンソーシアムによる検証作業においては、多大なるご協力をいただきました。ここに感謝を申し上げます。

別紙

各ソフトウェアによる動作検証の詳細結果について

対象ソフト		
	動作環境 検証内容	結果
・DB2 Database for Linux, UNIX, and Windows Version 9.5 (Fix pack 3) ・DB2 Express-C 9.5.2 for Linux (32-bit) [Ubuntu 8.10 Serverのみ] ・IBM Data Studio Developer Version 2.1 ・WebSphere Application Server Community Edition V2.1.1		
	・Red Hat Enterprise Linux Advanced Platform 5.3 ・SUSE Linux Enterprise Server 11 ・Ubuntu 8.10 Server [DB2 Express-Cのみ] ・Windows Server 2003 Enterprise Edition SP2 ・Windows XP Professional SP2	
	1 **XML スキーマ検証** データベースにXML スキーマを登録しXML スキーマの妥当性を検証する。	XML スキーマをデータベースに登録できることを確認した。
	2 **XML インスタンス検証**	
	a) データベースにサンプルXML インスタンスを登録し検証する。その際にXML スキーマを使ったValidation を実施する。	サンプルXML を全てXML スキーマでValidation し、データベースに登録できることを確認した。
	b) XMLスキーマのxs:any要素部分を検証するため、当該部分に新しい要素を追加したサンプルXMLインスタンスをデータベースに登録し検証する。その際にXML スキーマを使ったValidation を実施する。	サンプルXML"70_01_01(090424)kisetsu.xml"の</ObservationAddition>の後に以下を追加したインスタンスでも、XML スキーマでValidation し、データベースに登録できることを確認した。 <NewElement xmlns="http://xml.kishou.go.jp/jmaxml1/addition1/">新しい要素の追加 </NewElement>
	3 **XML データ処理** データベースに登録されたXML インスタンスに対して以下の処理・操作を実施する。	
	a) XML パターンに基づく索引作成	指定したXML パターンに基づく索引が作成できることを確認した。
	b) 全文検索索引の作成	全文検索のための索引が作成できることを確認した。
	c) XQuery による検索	試験用のXQuery でXML インスタンスの検索を実行し、結果が得られることを確認した。
	d) XQuery による検索 (全文索引)	XQuery で、1 つの文に“最低気温”と“平年並”を含む、もしくは“最低気温”と“高い”を含むXML インスタンスから、Bodyの部分を結果として返すようにした、全文検索を含む検索を実行し、結果が得られることを確認した。
	e) XQuery Update Facility によるXML インスタンスの部分更新	XQuery でXML インスタンスのバージョンに関わる3 つの部分を変更する部分更新を実行し、結果が得られることを確認した。
	f) XSLT スタイルシートによるHTML への変換	HTML に変換するためのスタイルシート(XSLT)を作成し、SQL による (XSLTRANSFORM を利用) HTML 変換を実施して問題なく変換できることを確認した。
	g) SOAP バインディグによるWeb サービスの作成と実行	SQL/XML(XMLQUERY, XMLEXISTSを利用)でタイトル (title) を入力パラメータとするWeb サービス化して実行し、問題なく結果が返ることを確認した。
	h) HTTP バインディグによるWeb サービスの作成と実行	SQL/XML(XMLQUERY, XMLEXISTSを利用)でタイトル (title) を入力パラメータとするWeb サービス化して実行し、問題なく結果が返ることを確認した。
	i) REST (HTTP GET バインディグ) によるWeb サービスの作成と実行	SQL/XML(XMLQUERY, XMLEXISTSを利用)でタイトル (title) を入力パラメータとするWeb サービス化して実行し、問題なく結果が返ることを確認した。

・WebOTX Application Server V8.2

・Windows Server 2008
・Java SE Development Kit 6

1 検証対象の XML スキーマを参照した WSDL を作成し、その WSDL を元に HTTP/SOAP プロトコルを用いた Web サービス、Web サービスクライアントを実装する。これについて、次の検証を実施する。	
① 全てのサンプル XML インスタンスについて、それを封入した SOAP メッセージが正常に伝達できるかどうかを検証する。	JAX-WS は、WSDL から Java のソースコードを生成する際、検証対象のXML スキーマを完全に Java のプリミティブ型とJavaBean にマッピングさせることができた。JAX-WS で実装したWeb サービス、Web サービスクライアントは、全てのサンプル XML インスタンスについて正常に動作することを確認した。 JAX-RPC は、WSDL から Java のソースコードを生成する際、SAAJ の型にマッピングされた。 JAX-RPC で実装したWeb サービス、Web サービスクライアントは、全てのサンプル XML インスタンスについて正常に動作することを確認した。 Java サーブレット内で SAAJ API を用い、SOAP Body 内の XML 処理に低レベル API（DOM、SAX、StAX）を使用して Web サービスを実装した。また、Java アプリケーションにて SAAJ API を用い、Web サービスクライアントを実装した。SAAJ で実装した Web サービス、Web サービスクライアントは、全てのサンプル XML インスタンスについて正常に動作することを確認した。
② SOAP メッセージの伝達先にて、全ての SOAP メッセージ、および SOAP メッセージに封入されたサンプル XML インスタンスについて、WSDL バリデーション、およびXML スキーマバリデーションを実施し、バリデーションが成功するかどうかを検証する。	JAX-WS は、サービスエンドポイントに「バリデーションを行う」設定をすることで、WSDL バリデーション、XML スキーマバリデーションを動作させた。JAX-RPC、SAAJ はバリデーション処理をサービスエンドポイントの実装に埋め込むことにより動作させた。全てのランタイムにて、全てのサンプル XML インスタンスについてのバリデーションが成功した。
③ 全てのサンプル XML インスタンスについて、それを封入した SOAP メッセージをFast Infoset 仕様に基づいてバイナリXML 化し、正常に伝達できるかどうか、および伝達先で正常にデコードできるかどうかを検証する。	全てのランタイムにて、全てのサンプル XML インスタンスについて、正常に伝達できること、な らびに正常にデコードできることを確認した。また、サ ンプル XMLインスタンスについて、比較的サイズが大きいもの（数十 KB 以上）では、バイナリ XML の電文長は通常の SOAP の電文長に対して約33%～25%に圧縮され、高い圧縮効果があることを確認した。
④ SOAP メッセージに封入する全てのサンプルXML インスタンスに対し、WS-Security仕様に定められた手順で任意の複数個所について署名、暗号化を行って伝達し、SOAP メッセージの伝達先にて署名検証、復 号化が正常に動作するかどうかを検証する。	全てのサンプル XML インスタンスについて、署名・署名検証、暗号化・復号化が正常に動作することを確認した。
2 検証対象のXML スキーマに沿った内容のXML を伝達するアプリケーションをREST/RESTfulで実装し、正常に伝達できるかどうかを検証する。	Java サーブレットを用いて実装し、正常に伝達できることを確認した。

・WebOTX Enterprise Service Bus V8.11	
・Windows Server 2008 ・Java SE Development Kit 6	
1 全てのサンプル XML インスタンスに対して独自開発の高速 XSLT プロセッサであるCHDL(Content Handler Description Language)を用いて簡単な XSLT を実行し、正常に変換されるかどうかを検証する。	検証対象の XML スキーマにて日本語が使われていることから、サンプル XML 文書に対してXSLT を実行した後の XML 文書に文字コードの不具合（文字化け）が発生しないかどうかという観点で検証した。その結果、文字コードの不具合は発生せず、正常に XSLT が実行できることを確認した。
2 ESB を使用し、様々な通信プロトコルで全てのサンプル XML インスタンスが正常に送受信できるかどうかを検証する。	HTTP（SOAP、REST/RESTful）、 JMS、File、FTP の各通信プロトコルにて、全てのサンプルXML インスタンスを外部システムと正常に送受信できることを確認した。RMI、CORBA、JCA、JDBC の各通信プロトコルについては、ESB の実装仕様として送受信するXML 書式を規定しているため、サンプル XML インスタンスの書式を変換することで、外部システムや外部リソースと正常に送受信できることを確認した。また、これ以外に ESB 内部で動作するシーケンス機構や XML 変換機構などの全てのコンポーネントにおいても、全てのサンプル XMLインスタンスを正常に処理できることを確認した。
・Interstage Application Server V 9.1（Fujitsu XML Processor）	
・Windows XP Professional SP2 ・Java 2 Runtime Environment, Standard Edition Version 1.4.2 (1.4.2_16) ・Java 2 Platform Standard Edition Runtime Environment Version 5.0 (1.5.0_17)	
XMLスキーマ検証 および XMLインスタンス検証	問題なし
・PFU XML コンバータV2.1 L10	
・PC機(CPU: Core 2 Duo/Memory: 1GByte) ・Windows Server 2003 R2 ・Java 2 Runtime Environment, Standard Edition Version 1.4.2	
防災情報XML データを入力データとして処理し、現行データ(平文形式)に変換可能かを検証した。あわせて、ＸＭＬコンバータの特長である高速変換が防災情報XMLデータでも可能かを検証した。 具体的な検証内容は以下の通り。	変換可能であることを確認した（特殊な変換要件については、ユーザー固有の変換処理が可能な外部処理での対応を含む）。 また、高速変換(1ミリ秒前後で変換)が防災情報XMLデータでも可能であることを確認した。
a) 防災情報XML データについて、入力データ形式定義ファイルとして定義可能か	検証対象とした防災情報XML データについて、入力データ形式定義を行うことができた。
b) 平文形式の現行電文を、出力データ形式定義ファイルとして定義可能か	検証対象とした現行電文（平文形式）について、出力データ形式定義を行うことができた。 特殊な変換要件で使用する外部処理の指定を含む。
c) 特殊な変換要件については、外部処理で対応可能か	特殊な変換要件であっても、外部処理を作成/定義することで対応することが可能であった。
d) 変換速度を測定し、ＸＭＬコンバータの特長である高速変換が防災情報XMLデータでも可能か	検証対象とした変換処理を1ミリ秒前後で処理できることを確認した。

・uCosminexus Application Server Standard 8 (Cosminexus XML Processor 8)		
	・Red Hat Enterprise Linux ES release 4 (Nahant Update 5) ・Windows XP Professional SP2 ・Java 2 Platform Standard Edition Runtime Environment Version 5.0	
	uCosminexus Application Server Standardに含まれるCosminexus XML Processor(JAXP1.3に準拠)が提供するAPIを利用し検証用プログラムを作成し、以下について検証した。	
	1 **XMLスキーマ検証 (正常系)** 全サンプルXMLデータ (146ヶ) に対してXMLスキーマ検証する。	成功した。 全サンプルXMLデータに対する検証結果が正常と検知された。
	2 **XMLスキーマ検証 (異常系)** 全サンプルXMLデータ (146ヶ) に対してXMLスキーマ検証する。スキーマファイルの内容を一部、異常となるように変更し、検証する。	成功した。 全サンプルXMLデータに対する検証結果が異常と検知された。
・HiRDB/Single Server 08-04 ・HiRDB XML Extension 08-04		
	・Windows XP Professional SP2 ・Red Hat Enterprise Linux AS release 3	
	1 **XMLインスタンス検証** 全サンプルXMLデータをDBに格納可能か確認する。	全サンプルXMLデータをINSERT文によりDBに格納できることを確認した。
	2 **XMLデータ処理** 全サンプルXMLデータをDBに格納し、以下の4点のXMLデータ処理が可能か確認する。	
	(a) XMLSERIALIZE関数、XMLEXISTS述語、XMLQUERY関数を使用した検索処理が可能か。	下記の例に示すような検索SQLを実行できることを確認した。 ①/jmx:Report/jmx:Control/jmx:Titleが”季節観測”であるXMLデータを取り出す。 ②/jmx:Report/jmx:Control/jmx:Titleの要素のみを取り出す。
	(b) 部分構造インデクス (XMLデータ中の特定の要素や属性の値をキー値とするインデクス) を作成可能か。	部分構造インデクスを定義できることを確認した。
	(c) 部分構造インデクスを使用した検索が可能か。	(b)で定義した部分構造インデクスを使用した検索ができることを確認した。
	(d) 全文検索インデクスを使用した検索が可能か。	全文検索インデクスを使用した検索ができることを確認した。

・Oracle Database 11g Release 1 (11.1.0.7)

1 **XML スキーマ検証** データベースにXML スキーマを登録する。	XML スキーマをデータベースに登録できることを確認した。
2 **XML インスタンス検証** データベースにサンプルXMLインスタンスを登録する。その際、XML スキーマによる妥当性検証を同時に実行する。以下の3パターンで動作確認。	
a) データベース表	全てのサンプルXMLインスタンスを登録できることを確認した（同時にXMLスキーマによる妥当性検証を実施）。また、XMLスキーマに反するXMLインスタンスを1つ作成し、このインスタンスは登録不可であることを確認した。
b) パーティション化されたデータベース表（ハッシュ・パーティションを利用）	全てのサンプルXMLインスタンスを登録できることを確認した（同時にXMLスキーマによる妥当性検証を実施）。ハッシュ・パーティションを利用していることから、全てのパーティションにデータが均等に割り振られていることを確認した。また、XMLスキーマに反するXMLインスタンスを1つ作成し、このインスタンスは登録不可であることを確認した。
c) XMLデータ列が圧縮形式で定義されたデータベース表（SecureFiles を利用）	全てのサンプルXMLインスタンスを登録できることを確認した（同時にXMLスキーマによる妥当性検証を実施）。また、XMLスキーマに反するXMLインスタンスを1つ作成し、このインスタンスは登録不可であることを確認した。
3 **XML データ処理** データベースに登録されたXMLインスタンスに対して以下の処理・操作を実施する。	
a) 特定のタグ・属性に対する索引の作成	XPathで指定した特定ノードに対してB*Tree索引およびビットマップ索引を作成できることを確認した。また、検索処理で索引が有効であることを確認した。
b) 特定のタグ以下に対する索引の作成	XPathで指定した要素以下の全てのノードに対して、タグ構造に基づく索引を作成できることを確認した（XMLIndex索引のパス・サブセットを利用）。また、検索処理で索引が有効であることを確認し
c) 全体に対する索引の作成	以下の2通りの方法で実施し、文書全体に対して、タグ構造に基づく索引を作成できることを確認した。また、検索処理で索引が有効であることを確認した。 1) XML索引（XMLIndex 索引） 2) 全文検索索引（Oracle Text の CONTEXT 索引 / PATH_SECTION_GROUP を利用）
d) SQL/XML関数による検索	SQL/XMLによって（extractValue を利用）、特定ノードの値を取得できることを確認した。
e) XQueryによる検索	サンプルXMLインスタンスそれぞれに対して、FLOWR構文によるXQueryを実行し（xmlQuery を利用）、結果が正しく返されることを確認した。
f) XMLインスタンスの部分更新	SQL/XMLによって（updateXML を利用）、XPathで指定された特定ノードを正しく更新できることを確認した。

・.NET Framework 3.5		
	・Windows Vista Enterprise	
	スキーマの読み込み、及びインスタンスのValidationとして、スキーマを使用した全インスタンスに対する総当りValidationを行う。	成功した。
・Visual Basic 2008 Express Edition		
	・Windows XP Professional SP3	
	1 XmlValidatingReaderにより、スキーマを使った全インスタンスの妥当性検証を行う。	妥当性検証に成功した。また、異常のあるインスタンスに対してはエラーを検知することにも成功した。
	2 スキーマをimportして独自に作成したwsdlを使って、Webサービスを呼び出すコードを自動生成する。自動生成したコードを開発環境に取り込み、コンパイルできるか確認する。	成功した。
・Visual C# 2008 Express Edition		
	・Windows XP Professional SP3	
	スキーマをimportして独自に作成したwsdlを使って、Webサービスを呼び出すコードを自動生成する。自動生成したコードを開発環境に取り込み、コンパイルできるか確認する。	成功した。
・libxml2-devel-2.7.3-1.fc10.i386、(xmlstarlet-1.0.1-6.fc9.i386) ・xerces-c-src_2_8_0、gcc (GCC) 4.3.2 20081105 (Red Hat 4.3.2-7) ・xerces-c-3.0.1、gcc (GCC) 4.3.2 20081105 (Red Hat 4.3.2-7)		
	・Fedora release 10 (Cambridge) Linux 2.6.27.5-117.fc10.i686	
	1 以下のコマンドにより、スキーマを使った全インスタンスの妥当性検証を行う。 xmlstarlet val -s jmx.xsd ファイル名.xml	妥当性検証に成功した。また、異常のあるインスタンスに対してはエラーを検知することにも成功した。
	2 xerces付属サンプルのSAX2Countを改造し、以下のコマンドによりスキーマを使った全インスタンスの妥当性検証を行う。 SAX2Count2 -v=always ファイル名.xml	妥当性検証に成功した。また、異常のあるインスタンスに対してはエラーを検知することにも成功した。

・Java SE Development Kit 6 (jdk1.6.0_12)		
	・Windows Vista Ultimate SP1	
	1 JDKのxjcコマンドを使用し、Javaのクラスを自働生成する。 xjc jmx.xsd javax.xml.bind.JAXBContextを使用して読み込み、スキーマを使って妥当性検証を行って、Javaのクラスにバインドする。	妥当性検証に成功した。また、異常のあるインスタンスに対してはエラーを検知することにも成功した。
	2 javax.xml.parsers.SAXParserFactoryにより、スキーマを使った全インスタンスの妥当性検証を行う	成功した。
・Java 2 Platform Standard Edition Development Kit 5.0 (jdk1.5.0_07)		
	・Windows Vista Ultimate SP1	
	javax.xml.parsers.SAXParserFactoryにより、スキーマを使った全インスタンスの妥当性検証を行う	成功した。
・AltovaXML 2008 (AltovaXML Version 2008 sp1)		
	・Windows XP Professional SP3	
	スキーマを使った全インスタンスの妥当性検証を行う。	成功した。
・Multi Schema Validator Ver.20080213		
	・Windows XP Professional SP2	
	スキーマを使った全インスタンスの妥当性検証を行う。	成功した。

・axis-1_4

・Windows Vista Ultimate SP1
・Java SE Development Kit 6 (jdk1.6.0_12)

スキーマをimportして独自に作成したwsdlを使って、Webサービス(サーバー側)を呼び出すコードを自動生成する。自動生成したコードを開発環境に取り込み、コンパイルできるか確認する。 java org.apache.axis.wsdl.WSDL2Java -s jmx.wsdl	NullablefloatとNullableintegerに対して自動生成されたコードがコンパイルできない。 NullablefloatとNullableinteger内のunionを、単なるString型と書き換えることで回避可能（※1）。 Axisの問題。 参考＞https://issues.apache.org/jira/browse/AXIS-2523

・axis2-1.4.1

・Windows Vista Ultimate SP1
・Java SE Development Kit 6 (jdk1.6.0_12)

1 スキーマをimportして独自に作成したwsdlを使って、Webサービス(サーバー側)を呼び出すコードを自動生成する。自動生成したコードを開発環境に取り込み、コンパイルできるか確認する。 wsdl2java.bat -uri jmx.wsdl -ss	成功した。
2 スキーマをimportして独自に作成したwsdlを使って、Webサービス(クライアント側)を呼び出すコードを自動生成する。自動生成したコードをクラスにバインドして、SOAP呼び出しを行う。 wsdl2java.bat -uri jmx.wsdl -g -u	一部のインスタンスで、実行時にバインドで失敗した。サーバ側でも同様の問題が発生する。 NullablefloatとNullableinteger内のunionを、単なるString型と書き換えることで回避可能（※1）。 Axis2の問題。 参考＞https://issues.apache.org/jira/browse/AXIS2-4001

※1 AxisとAxis2では以下のとおりjmx_eb_nullable.xsdを書き換えることにより回避可能であることを作業者にて確認している。

```
<?xml version="1.0" encoding="UTF-8"?><xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:jmx_eb="http://xml.kishou.go.jp/jmaxml1/elementBasis1/"
elementFormDefault="qualified"
targetNamespace="http://xml.kishou.go.jp/jmaxml1/elementBasis1/">
<xs:simpleType name="nullablefloat">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:simpleType name="nullableinteger">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
</xs:schema>
```