報 告

気象庁防災情報 XML フォーマットの詳細と策定経緯

杉山 善昭 *1・竹田 康生 *2・山腰 裕一 *3・清本 真司 *4・
中村 政道 *5・長谷川 昌樹 *4・平原 隆寿 *6・横井 貴子 *7

目 次

*1 予報部業務課 *2 予報部予報課 *3 観測部気象衛星課 *4 地震火山部地震予知情報課
*5 仙台管区気象台技術部地震火山課 *6 地球環境・海洋部海洋気象課海洋気象情報室 *7 総務部総務課広報室

1 はじめに*

気象庁防災情報 XML フォーマット（以下「気象庁 XML」という.）は，従来の情報ごとに異なる電文形式に対する後継フォーマットとして，XML 技術を採用した気象庁としての新しい提供形式である.

平成 22 年度からの市町村を単位とした気象警報の発表などの計画を控えて，より詳細で高度化された防災情報の提供様式を検討すべき時節の到来と捉え，「気象情報部外提供推進委員会」（委員長：総務部参事官．以下「委員会」という.）の下に，部外における気象情報の更なる有効活用に資する提供形式の検討・決定を目的とした「気象情報提供形式検討部会」（部会長：総務部企画課企画調整官）を設置し，約 3 年にわたる仕様策定作業を行った．策定にあたっては，昨今の技術革新の激しい中でも 10 年程度は利用可能なフォーマットとすることを目指し，XML 技術の専門家が集う XML コンソーシアムの協力を仰ぐとともに，利用者からの声を検討に反映させるなど，従前のフォーマット策定過程にない取り組みを行った．これらの成果が，気象庁 XML の仕様や運用指針として結実した.

ここでは，気象庁 XML の策定・普及過程における業務的・技術的観点などの様々な議論を取りまとめるとともに，その際に気付いた点や反省点などを記述することにより，今後の気象庁 XML の管理・運営に資することを期待する.

2 気象庁 XML 策定・普及体制**

2.1 背景

気象庁で XML を採用した最初の情報は，気象警報･注意報であり，平成 16 年 3 月に遡る．これは，XML1.0 が勧告された平成 10 年（1998）から数年経ち，様々な業界において XML が具体的業務に実装されていった時期である.

当時，気象警報・注意報・予報の改善・拡充の一環として平成 15 年度から報道機関等へ XML 化のメリット等を説明していた．その際，「XML 化の対応には数年掛かる」，「短期間に対応するのであればコード化する方がよい」，「地震火山関係の電文はコード化が主流であり気象庁としての電文形式に係る方向性を明確にすべき」など，XML 技術の導入に係る気象庁としての方針が問われることとなった．旧形式の気象警報・注意報 XML 電文の提供開始後も，「共通化した XML でないと困る」，「地震情報も含めた気象庁全体で本線は XML だと宣言して欲しい」といった要望が引き続き寄せられた.

情報処理技術を積極的に業務へ取り込んできた気象庁であるが，この時の経験から，新しい技術を導入し情報内容を高度化しても，利用者の対応に配慮しなければ情報を使ってもらうことができないということを強く認識することとなった.

また，この頃，既に米国では“CAP”（第 13.1 節参照）という名の XML を用いた警報等の伝達が開始されていた．組織や事象に捕らわれず，米国政府内のマルチハザードを伝達するための統一的な提供形式として，具体的な実装や利活用が進んでいた．このような国際動向も気象庁 XML を策定する上での大きな背景となった.

2.2 庁内体制始動・仕様策定

気象庁が，情報の内容や提供手段ではなく提供形式（フォーマット）について，気象から地震・津波・火山などまでの数多くの防災情報を対象として，一体的に整理するとともに一つの仕様として策定したのは今回が初めてであるといえる．ここでは，このような初めての取り組みについて，部会における検討経緯などを含め気象庁 XML の策定までの流れを整理する.

検討体制の発足は冒頭のとおり気象情報提供形式検討部会が設置された平成 19 年 3 月 27 日である．同部会では，既に多様な分野で応用が盛んあることなどを理由に XML に的を絞って具体的な技術検討を進めるとともに，気象庁の電文の XML 化による利便性を探求した．検討を進めるほどに作業が具体化され，技術的にも深い議論を

* 第 1 章　山腰

** 第 2 章　第 2.1 節～第 2.4 節　山腰，　第 2.5 節～第 2.6 節　杉山

するようになっていった．場所や時間などの形式を議論する中で，形式の共通性を意識するばかりに情報の意味するところが異なってしまうのではないかなどといった議論もあった．

同部会での約 1 年の検討を経て，平成 20 年 1 月 7 日の委員会で気象庁 XML の策定に向けたロードマップが決定された．具体的には，提供形式を XML としたこと，これを気象庁 XML として仕様の作成を行うために庁内に作業グループ（以下「WG」という．）を発足させるとともに後述の XML コンソーシアムの協力を得ること，大きな業務変革等とスケジュールを整合させることなどをまとめた部会の中間報告を承認した．この内容を 2 月 1 日に気象庁の決意表明として報道発表した．

その後，ロードマップに沿って各方面の利用者に対して方針説明等を行いつつ，仕様書の作成を進めていった．仕様書の具体化においては，利用者の理解や準備を促すため，事前にドラフト版を作成し意見募集を行った（平成 20 年 5 月 16 日及び平成 21 年 1 月 27 日委員会決定）．このような利用者からの意見等に対応した後，XML 電文の提供開始をおよそ 1 年後に控えた平成 21 年 5 月 15 日に気象庁 XML の仕様初版を公開した．

2.3 XML コンソーシアムの協力

気象庁は情報の内容に対しては専門家であるが，提供形式とした XML 技術については業界動向などの客観的な情報収集が必要と考えた．また，多くの情報を一元的に従来の提供形式から XML 形式へ変更するためには，気象庁内多数の担当者の技術力向上を図るとともに，利用者が実装する際に必要となる IT 技術・開発者の目線を必要としていた．このような観点から，気象庁 XML の策定には XML の専門家からの協力が重要だと考えていたところ，日本における XML の利活用の促進を目指す非営利団体の XML コンソーシアムに，部会事務局である企画課担当者の 1 通の問い合わせメールがたどり着き，これが以降 3 年間にわたる協力関係の始まりとなった．XML コンソーシアムは，これまでに ContactXML（住所録に利用），ContentsBusinessXML（コンテンツ取引に利用），TravelXML（旅行業に利用）で，XML 技術による情報の標準化についての実績があり，複数のテーマ別部会により XML を主軸とした実装寄りの研究・開発・検証を行っている団体であったことから，気象庁 XML の策定における最適な協力団体であった．

XML コンソーシアムから得た協力のうち主なものを紹介する．策定作業全般にわたり，庁内の個々の電文の XML 化を担当する WG メンバーとの間では定常的な勉強会を行うとともに，担当者の考えをレビューしていただいた．また，気象庁 XML 公開前の市場製品に対する動作検証は，当庁単独では実現しない XML コンソーシアムならではの技術協力であった．XML コンソーシアムでは，会員企業に対して協力メンバーを募集し，各種プラットフォーム上での検証結果を取りまとめ，その結果をホームページ等で公開した．

このように，システム市場を代表する方々の評価・検証結果の公表を含め，策定作業の各段階において協同して公表することにより，利用者や開発者に対する気象庁 XML への移行に向けた訴求に効果があったと考えている．

2.4 関連機関等の動き

気象庁の活動と時を同じくして，防災関係の情報共有化の動きが加速していた．

内閣府の防災担当部門と科学技術政策担当部門がともに進めていた「社会還元加速プロジェクト」の一つ「きめ細かい災害情報を国民一人ひとりに届けるとともに災害に役立つ情報通信システムの構築」を目標とした各種検討がある．例えば，情報の規格化が必要であるとして，当時，文部科学省における災害関連のメタ情報の XML 化や国土交通省における河川管理情報の XML 化などの取り組みをレビューし，どのように災害リスク情報を共有・促進するかを議論しはじめたところであった．ここで時を得て気象庁 XML の検討状況を説明できたため，具体的な取り組みであり先駆的な事例であると多くの防災関係機関からも XML 化を歓迎された．

また，デジタル放送への切替えを契機にメディア向けの XML 規格（TVCML[1] 等）が検討され

るとともに，自治体における一般行政情報の共通化（第 13.3 節参照）が進められていた．

このように，防災関係機関における XML 技術をキーワードとした情報共有化は必然の流れであったといえる．

2.5 普及・活用促進体制

気象庁 XML の仕様は平成 21 年 5 月 15 日に公開されることとなり，策定から普及へと軸足を移すことになった．このため，公開前日に当たる 5 月 14 日の委員会において，気象情報提供形式検討部会は改組し，気象庁防災情報 XML フォーマット普及・活用促進部会として，これまでの活動を引き継ぎつつ新たな活動段階へと踏み出した．活動の方向性としては，これまで XML のフォーマットがいかにあるべきかを検討してきたところから，具体的なサンプルや解説資料の充実などを基盤作業として，各種部外機関に対する気象庁 XML の説明，売り込みを図ることが主たる活動となった．

また，活動主体が普及・活用促進へと移るのに併せる形で，これまで協力頂いた XML コンソーシアムにおいても気象庁 XML をいかに活用するかという観点で研究テーマの選定が行われた．具体的には，当時 XML コンソーシアムにおいて活動していたビジネス・イノベーション研究部会と SOA 部会を中心として気象庁 XML の利活用業務モデルの策定とシステム設計，Web サービス実証部会と次世代 Web 活用部会，及び関西部会により同業務モデルのデモ実装が実施された．これらは平成 22 年 3 月 17 日，18 日の XML コンソーシアム Week においてその成果が発表された．

なお，XML コンソーシアムは平成 22 年 3 月 31 日をもって，10 年間にわたる活動を終了している．また，気象庁防災情報 XML フォーマット普及・活用促進部会も，運用開始とともに運用管理が主たる業務となることから，平成 23 年 5 月 9 日に気象庁防災情報 XML フォーマット運用管理・利活用促進部会に改組して，維持運営を中心に業務を担っている．

2.6 資料の公開

一連の報道発表や意見募集のみならず，今回の気象庁 XML の各種仕様等の情報については専用のホームページを「気象庁防災情報 XML フォーマット 情報提供ページ」(http://xml.kishou.go.jp/)（以下「気象庁 XML の HP」という．）として公開している．

これまで，気象庁で利用している各種フォーマットに関する情報については技術情報として，気象業務支援センター経由等で利用者提供を行ってきたが，今回の気象庁 XML に関しては気象庁 XML の HP 上で，一般国民向けにオープンな形で公開することとなった．これは XML 技術が汎用的であり関連技術がオープンな市場技術であること，気象庁 XML の普及には関連情報の広い公開が必要であることなどから，また XML コンソーシアムからの強い提言もあって，このような形で公開することとした．

当初，技術資料は以下のものを公開していた．

・気象庁防災情報 XML フォーマット仕様書
・同辞書・スキーマファイル
・コード管理表・個別コード表
・サンプル電文

議論を進めていく中で，仕様書だけでは説明ができないデータ部分の運用等の説明や，外部からスタイルシート (XSLT) の提供の要望が深まった．このことから，平成 22 年 5 月 14 日に以下の資料を追加公開した．

・気象庁防災情報 XML フォーマット運用指針
・サンプルスタイルシート

その後も引き続き詳細について検討を進めていったが，その過程で，気象庁 XML の運用の共通化としての観点から，一部電文の運用については他の電文と調整がとれていないことが明確化した．このことは，各業務に基づき違う運用を行っている XML 化対象電文を，気象庁 XML として統一する作業として，必ずしも理想通りにはいかないことを示した．一方で，この運用の違いは，丁寧に説明すれば明らかに異質な運用とまではいえないことから，仕様書や辞書，サンプル電文のみでは利用者に対する情報が不足していると考え，新たに電文ごとの解説資料を作成することと

し，平成 22 年 9 月 15 日に以下の資料を追加公開した，

・電文ごとの解説資料

基礎資料の提供としては，これらを提示することにて現在（平成 24 年 10 月）まで至っている．

3 XML について*

XML（Extensible Markup Language）は文書の電子化技術 SGML（Standard Generalized Markup Language）のサブセットとして，1998 年 2 月に W3C（World Wide Web Consortium）より勧告された．XML は現在，構造化された文書や直列化（Serialized）されたデータの保存形式として，最も一般的な方式・言語の一つである．

XML には次のような特徴がある．

・要素のネスティング（入れ子構造）によって，ツリー構造による擬似多次元情報の表現が可能
・基本的に自由度の高い表現記法
・DOM（Document Object Model）や SAX（Simple API for XML）といった標準 API のほか，データバインディング技術などが充実している
・XML データを直接取り扱うことが可能なデータベースや言語（XQuery）などが発達してきている
・SOAP など Web サービスの基盤技術として利用されており，疎結合システム間連携に幅広く利用されている

詳細については，たのしい XML[2]，リファレンスとして atmarkIT の XML 用語辞典 [3]，w3schools.com[4] などが参考になる．

4 XML スキーマ**

4.1 XML スキーマ言語

(1) XML スキーマ言語

XML スキーマ言語とは，XML 文書構造を定義する各種言語である．XML の構造を定義する意義については幅広い議論があるが，一般的には，XML の構造や値の基底型等の構造情報を提供することにより，作成側では XML インスタンス作成の際のひな形と妥当性検証（バリデーション）機能を，利用側では XML インスタンスの設計図を受けることができるようになることである．このことは，二者間における情報伝達の重要なインターフェース情報となることを意味する．なお，XML の構造定義に関する議論の一つとしては，「ボヘミアンと貴族の階級闘争」[5] を参照のこと．

(2) 気象庁 XML における W3C XML Schema の採用（仕様 2.1 項）

XML スキーマ言語には代表的なものとして DTD[6]，W3C XML Schema [7]，RELAX NG[8] の 3 種類がある．DTD については名前空間を適切に扱えない問題があることから除外するが，気象庁 XML で採用する XML スキーマ言語として，W3C XML Schema と RELAX NG のいずれを採用するかについては様々な議論があった．結論として，RELAX NG は ISO/IEC 19757-2 として国際規格にもなっているが，処理可能なプロセッサーは W3C XML Schema の方が多いことから，気象庁 XML ではこちらを採用することとした．

(3) XML スキーマと XML 文書構造

XML スキーマ言語は XML 文書構造を定義する言語であり，基本的にはあらゆる XML インスタンスの構造を定義できる．しかしながら，XML スキーマ言語の構文上，構造定義しづらい XML インスタンスがある．このような XML インスタンスは，構造定義が可能であっても XML スキーマ処理系による実装不安定性（汎用ライブラリー等が正しく動作しない）がある場合や，不具合の原因となる場合もありうる上，そもそも構造定義しづらいことは XML スキーマ言語の理念と相いれない構造となっている場合が多い．このことから，本章において解説している XML スキーマの利用形態が元となって，XML インスタンスの構造が決定されている部分が多々ある．このため，気象庁 XML の構造を本質的に理解するた

* 第 3 章　杉山

** 第 4 章　杉山，　第 4.2 節　杉山・竹田，　第 4.3 節（4）項　竹田

めには本章について理解しておく必要がある．

なお，気象庁 XML の策定の過程においては，XML スキーマにより XML インスタンスの構造を決めるのみならず，XML スキーマを作成してみた結果，XML スキーマ上で表現の難しいものについては XML インスタンスの構造の方を修正している場合が多い．このような策定過程を経たところだが，一般論としても，XML インスタンス構造の決定に当たっては，XML インスタンスの検討，XSLT 等を用いた XML インスタンスの利用，XML インスタンスの構造を定義する XML スキーマの作成の 3 つの作業を繰り返し実施して，それぞれの問題点を整理・修正していくことが必要になるかと思われる．

4.2 デザインパターン

W3C XML Schema による XML スキーマのデザインパターンとして，大きく分けて 4 種類の記法がある．これらの記法は，W3C XML Schema における要素（element）と型（complexType, simpleType）の定義がグローバルとなっているか，それともローカルとなっているかによって XML スキーマ内での参照の方法が変わることから，その典型的な記法として整理されたものである．

グローバルな定義とは，要素又は型の宣言が XML スキーマのルート要素である schema 要素直下にて定義されるものである．このように定義された要素又は型は，XML スキーマのいずれの場所でも要素参照及び型参照にて利用すること（呼び出すこと）が可能である．一方，ローカルな定義とはグローバルな定義の子孫として定義される要素又は型である．このように定義された要素又は型は，定義した要素及び型の子孫でのみ要素参照及び型参照にて利用が可能となる．このような参照の可能範囲を有効に利用し，記法として利活用しやすくしたものが以下のデザインパターンである．なお，型とはオブジェクト指向でいうところのクラス，要素はインスタンスの定義に近い．

(1) Russian Doll 型（ロシア人形）

全ての要素と型の定義がローカルとなっているものである．

全ての要素と型は使い回しが利かないため，XML インスタンスに沿った形で構造を定義することとなり，ネスティング構造（入れ子・親子構造）の階層についても XML インスタンスに近いものとなる．（第 1 図リスト b）

なお，ロシア人形とはマトリョーシカのこと．

(2) Salami Slice 型（サラミスライス）

全ての要素の定義がグローバルで，型の定義がローカルとなっているものである．

全ての要素の定義がルート要素の schema 要素直下に定義されているため，スキーマ定義上のどこからでも定義された要素が利用できる．どこからでも要素の利用が可能であるということは，インスタンス全体で要素名の一意性が求められることとなり，このことから，XML インスタンスから見た場合，同じ要素名は必ず同じ構造となる．また，要素定義がグローバルとなることから，XML インスタンスの階層構造にかかわらず，XML スキーマでは一定以上のネスティングが発生しない．（第 1 図リスト c）

(3) Venetian Blind 型（ベネチアン・ブラインド）

Salami Slice 型とは反対に，全ての要素の定義がローカルで，型の定義がグローバルとなっているものである．

全ての型の定義がルート要素の schema 要素直下に定義されているため，スキーマ定義上のどこからも定義された型が利用できる．型名では一意性が求められるが，XML インスタンスから見ると型名は見えず，また，ある一つの型に対して要素名称を複数割り当てることが可能であることから，要素名による構造上の制限は発生しない．このことは，XML 設計上の自由度が制限されないことを意味する．また，型定義がグローバルとなることから，XML インスタンスの階層構造にかかわらず，XML スキーマでは一定以上のネスティングが発生しない．（第 1 図リスト d）

(4) Garden of Eden（エデンの園）

全ての要素と型の定義がグローバルとなってい

リストa:サンプルXML

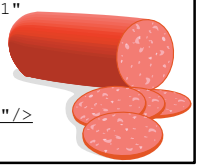
```
<?xml version="1.0" encoding="UTF-8" ?>
<Report xmlns="http://xml.kishou.go.jp/testxml/">
  <Title>テスト電文</Title>
  <DateTime>2009-01-09T02:25:34Z</DateTime>
  <Head>
    <Headline>テストです. </Headline>
  </Head>
</Report>
```

リストb:Russian Doll型

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:test="http://xml.kishou.go.jp/testxml/"
elementFormDefault="qualified"
targetNamespace="http://xml.kishou.go.jp/testxml/">
  <xs:element name="Report">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1" name="Title"
              type="xs:string"/>
        <xs:element maxOccurs="1" minOccurs="1"
              name="DateTime" type="xs:dateTime"/>
        <xs:element maxOccurs="1" minOccurs="1" name="Head">
          <xs:complexType>
            <xs:sequence>
              <xs:element maxOccurs="1" minOccurs="1"
              name="Headline" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

リストc:Salami Slice型

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:test="http://xml.kishou.go.jp/testxml/"
elementFormDefault="qualified"
targetNamespace="http://xml.kishou.go.jp/testxml/">
  <xs:element name="Report">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1"
              ref="test:Title"/>
        <xs:element maxOccurs="1" minOccurs="1"
              ref="test:DateTime"/>
        <xs:element maxOccurs="1" minOccurs="1"
              ref="test:Head"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Title" type="xs:string"/>
  <xs:element name="DateTime" type="xs:dateTime"/>
  <xs:element name="Head">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="1" minOccurs="1"
              ref="test:Headline"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Headline" type="xs:string"/>
</xs:schema>
```



第 1 図 スキーマの各デザインパターンの見本

リスト b ～ e はリスト a：サンプル XML に対する XML スキーマ．リスト b：Russian Doll 型ではサンプル XML と同じ入れ子構造となる．リスト c：Salami Slice 型では要素定義をグローバル化しているため，xs:schema 直下には要素定義（下線）しか現れない．リスト d：Venetian Blind 型では型定義をグローバル化しているため，xs:schema 直下には型定義（下線）しか現れないが，ルート要素になりうるものは要素定義が出現する．リスト e：Garden of Eden 型では要素定義も型定義もグローバル化しているため，いずれも xs:schema 直下にしか現れない．

リストd:Venetian Blind型

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:test="http://xml.kishou.go.jp/testxml/"
elementFormDefault="qualified"
targetNamespace="http://xml.kishou.go.jp/testxml/">
  <xs:element name="Report" type="test:type.report"/>
  <xs:complexType name="type.report">
    <xs:sequence>
      <xs:element maxOccurs="1" minOccurs="1" name="Title"
              type="xs:string"/>
      <xs:element maxOccurs="1" minOccurs="1" name="DateTime"
              type="xs:dateTime"/>
      <xs:element maxOccurs="1" minOccurs="1" name="Head"
              type="test:type.head"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="type.head">
    <xs:sequence>
      <xs:element maxOccurs="1" minOccurs="1"
               name="Headline"
               type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

リストe:Garden of Eden型

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:test="http://xml.kishou.go.jp/testxml/"
elementFormDefault="qualified"
targetNamespace="http://xml.kishou.go.jp/testxml/">

  <xs:complexType name="type.report">
    <xs:sequence>
      <xs:element maxOccurs="1" minOccurs="1"
              ref="test:Title"/>
      <xs:element maxOccurs="1" minOccurs="1"
              ref="test:DateTime"/>
      <xs:element maxOccurs="1" minOccurs="1"
              ref="test:Head"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="type.head">
    <xs:sequence>
      <xs:element maxOccurs="1" minOccurs="1"
              ref="test:Headline"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="Report" type="test:type.report"/>
  <xs:element name="Title" type="xs:string"/>
  <xs:element name="DateTime" type="xs:dateTime"/>
  <xs:element name="Head"
               type="test:type.head"/>
  <xs:element name="Headline"
               type="xs:string"/>

</xs:schema>
```

第 1 図　続き

るものである．

Salami Slice 型と同様，スキーマ定義上のどこからも定義された要素が利用でき，XML インスタンスから見た場合，同じ要素名は必ず同じ構造となる．また，スキーマ定義上のどこからも定義された型が利用できる．要素定義及び型参照がグローバルとなることから，XML インスタンスの階層構造にかかわらず，一定以上のネスティングが発生しない．一般的に機械可読性が高く，人間可読性は著しく低い．（第 1 図リスト e）

IDE ソフトウェアの NetBeans 等のプラグインでは，XML の構造設計の際に，これらのデザイ

ンパターンを選んでXMLスキーマの自動出力が可能となっている．

気象庁ではこれらのデザインパターンの内，ベネチアン・ブラインド型を採用した．記述や読解の平易さ，型の再利用などのメリットもあるが，ベネチアン・ブラインド型の記法は辞書（第5.1節参照）の記載やXMLスキーマの自動生成において，非常に親和性が高かったことも大きな理由である．

なお，気象庁XMLの全ての辞書において要素の定義をローカルに定義しているわけではなく，全ての電文に共通して利用する基本要素や部品を定義している共通辞書（基本要素）では，要素名も全ての電文で共通のものを用いるべきとの考え方により，グローバルの要素定義もしている．また，将来の拡張のため，共通辞書（追加要素）でも追加される要素を予めW3C XML Schemaのany要素で定義した部分から参照されるよう，要素定義をグローバルに行っている．

4.3 W3C XML Schema構文における注意点

W3C XML Schemaによる各種構造等を示す構文について，気象庁XML策定の際の注意点を以下にまとめる．

(1) コンテンツモデルの組立て

コンテンツモデルとは，個々の要素等の出現がどのようになるかといったモデルを示す．

このコンテンツモデルにおいて，W3C XML Schemaでは，sequenceグループ要素，choiceグループ要素，allグループ要素の3つのグループ要素が指定できる．これらのうち，記載順序順に出現することが求められるsequenceグループ要素については利用するが，いずれかの出現を選択するchoiceグループ要素，及び全ての要素を必須とするallグループ要素については以下の理由により利用しないこととした．

- 要素が出現する順序は事前に知り得た方が不具合の発生を抑止できること．
- 気象業務全般及びシステム的にXMLの各要素の出現順を定めることが可能であること．
- choiceグループ要素で可能なことは，順番を決めた上でsequenceグループ要素とminOccurs属性値の0の指定により，実質的に利用可能であること．逆に順番を完全に不定とすることによる不具合の発生の可能性が高いこと．
- allグループ要素で可能なことは，sequenceグループ要素とminOccurs属性値の1の指定により，実質的に利用可能であること．
- choiceグループ要素についてはライブラリー等の実装上，適切に扱われるかどうかに不安があるというXMLコンソーシアムからの助言があったこと．

(2) 単純型派生データ型の利用制限

単純型派生データ型はlist型，restriction型，union型の3つの型がある．list型については，空白文字区切りによる要素値の列挙が可能であることから，XML要素の繰り返し構造を省くことによりXML全体のデータ容量を減らすことが可能であり，当初はlist型の多用を検討していた．しかしながら，要素値の取扱いについては要素の繰り返しによって実施すべきであり，また，その値の取扱いにおいてはXML外の処理系（ライブラリーや実装）を利用しなければいけないことから，list型は極力避けるべきというXMLコンソーシアムからの助言を参考として，list型は基本的に利用していない．list型を限定的に利用しているのは以下の場合である．

- 2つ以上出現する事例は少ない場合．例外的に利用する場合でも，列挙されている個々の要素値を分解せずに一連の文字要素として取り扱っても問題とならない場合．
- 非常に多数の要素値を示さなければならない場合で，要素値の示す情報の業務的な重要度が余り高くない場合．
- 他の規格によりlist型が指定されている場合．

restriction型，union型における利用制限は行っていない．

なお，union型においては，XMLコンソーシアムによる動作検証の際，SOAP通信ソフトウェアであるAxisとAxis2において自動生成されたコードがコンパイルできない，及び実行時バイン

ドができない不具合が発生した．これはAxisとAxis2側の既知の問題ではあるが，修正される見込みが当分ないことから，利用者側でスキーマを改変して動作するようにするなど，これらは利用者処理側で対応する必要がある．なお，このことはXMLコンソーシアムにおいて動作検証を実施した際の結果として判明したことから，同報告においてもこの内容を解説している．

(3) オブジェクト指向的構造要素

オブジェクト指向的構造要素には，abstract属性やfinal属性がある．抽象型であることを示すabstract属性や派生型の禁止を示すfinal属性を利用することにより，オブジェクト指向におけるデータ構造をそのままXML化するために用いることが可能である．しかしながら，バージョン管理の運用とXML構造のオブジェクト指向と併せたデータモデルを実験的に構築してみたが，運用及び実装系における処理が想定通り動作しなかったことから，これらオブジェクト指向的構造要素の利用はしないこととした．

(4) any要素

一般的にXMLの要素の追加や変更を行う場合，新しいスキーマを事前に配布，周知することとなる．一方，例えば平成23年の東日本大震災後の電力不足等の対応のために「高温注意情報」を急遽開始したように，気象庁の防災情報は，緊急的対応として，新しい電文の配信や要素を追加した電文の配信を実現する必要がある．これら対応の実現のため，気象庁XMLでは未定項目としてW3C XML schemaにおけるany要素を利用することとしている．

any要素は，W3C XML schemaにおいてワイルドカードとして任意の要素の出現を定義するものである．any要素は属性として，最大出現回数を示すmaxOccurs属性，最小出現回数を示すminOccurs属性のほかに，出現する要素の定義がなされている名前空間を指定するnamespace属性，妥当性検証の方法を指定するprocessContents属性をとることができる．

namespace属性については，スペースで区切られた名前空間のリスト，任意の名前空間の要素を示す「##any」，このスキーマが属している名前空間以外の要素を示す「##other」，このスキーマが属している名前空間の要素を示す「##targetNamespace」，名前空間で修飾されていない要素を示す「##local」が指定できる．namespace属性と次項のUnique Particle Attribution制約は関連が大きいので，辞書の作成者は合わせて理解する必要がある．

processContents属性については，要求された名前空間のスキーマを取得し，それらの名前空間からの要素を検証する必要がある「strict」，要求された名前空間のスキーマを取得し，それらの名前空間からの要素を検証するが，スキーマを取得できない場合でもエラーとしない「lax」，XMLプロセッサーは，指定した名前空間からの要素を検証しない「skip」が指定できる．processContents属性は，公開するときにはバージョン管理も考慮して「lax」を指定しているが，辞書やサンプル電文の作成段階においては厳密に妥当性検証をするために「strict」を指定し厳密に行うことが望ましい．

any要素の具体的利用の検討については，第10章にとりあげる．

(5) Unique Particle Attribution制約

Unique Particle Attribution（UPA）制約は，インスタンス上のある要素について，XMLスキーマ上の定義が複数の方法にて解釈できる状態になってはいけないという制約を示す．具体的には，第2図のようなXMLスキーマがある場合に，同図のXMLインスタンスについてはXMLスキーマ上の要素定義を行っている2行のいずれにでも解釈できるが，UPAとはこのように二重解釈が絶対にされない状態を示す．W3C XML SchemaはUPA制約を遵守するように要請している．

UPA制約はXML設計上，配慮が必要なポイントである．W3C XML schemaにはany要素という，任意の要素の出現（ワイルドカード）を定義する構文があるが，このany要素の周辺ではUPA制約違反が発生しやすいため注意が必要である．具体的には第3図リストaに示すとお

り，any 要素により出現する要素の属する名前空間指定（namespace 属性）を「任意」(“##any”) や「このスキーマが属している名前空間」(“##targetNamespace”) とする場合に多発する（名前空間指定省略時は「任意」扱い）．UPA 制約違反を回避する手段としては次のようなものがある．

・namespace 属性を“##any”や“##targetNamespace”にしない（第 3 図リスト b)．

リストa:XML**スキーマ**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:test="http://xml.kishou.go.jp/testxml/"
elementFormDefault="qualified"
targetNamespace="http://xml.kishou.go.jp/testxml/">
 <xs:element name="a">
    <xs:complexType>
      <xs:sequence>
         <xs:element maxOccurs="1" minOccurs="0" name="b"
              type="xs:string"/>
         <xs:element maxOccurs="1" minOccurs="0" name="b"
              type="xs:string"/>
       </xs:sequence>
     </xs:complexType>
   </xs:element>
</xs:schema>
```

リストb:**サンプル**XML

```
<?xml version="1.0" encoding="UTF-8"?>
<a>
  <b>xyz</b>
</a>
```

第 2 図　Unique Particle Attribution（UPA）制約違反となる例

リスト a の XML スキーマでは要素 <b> が最小出現数 0 以上として 2 回定義しているため（下線部），リスト b のサンプル XML の <b> 要素がいずれの定義に一致するかが不明瞭となっている．

リストa:UPA**制約違反となる**XML**スキーマ**

```
<xs:element name="Name" minOccurs="1" maxOccurs="1" type="xs:string" />
<xs:element name="Code" minOccurs="0" maxOccurs="1" type="xs:string" />
<xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax" />
```

リストb:**名前空間の指定で回避**

```
<xs:element name="Name" minOccurs="1" maxOccurs="1" type="xs:string" />
<xs:element name="Code" minOccurs="0" maxOccurs="1" type="xs:string" />
<xs:any minOccurs="0" namespace="##other" maxOccurs="unbounded" processContents="lax" />
```

リストc:**出現回数の指定で回避**

```
<xs:element name="Code" minOccurs="0" maxOccurs="1" type="xs:string" />
<xs:element name="Name" minOccurs="1" maxOccurs="1" type="xs:string" />
<xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax" />
```

リストd:**入れ子構造を**1**段階掘り下げて回避(**XML**インスタンスも変更)**

```
<xs:element name="Name" minOccurs="1" maxOccurs="1" type="xs:string" />
<xs:element name="Code" minOccurs="0" maxOccurs="1" type="xs:string" />
<xs:element name="OptionElement" minOccurs="0" maxOccurs="1"type="jmx_wb:type.OptionElement" />
<xs:complexType name="type.OptionElement">
  <xs:sequence>
    <xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax" />
  </xs:sequence>
</xs:complexType>
```

第 3 図　UPA 制約違反の回避方法

リスト a が違反となる XML スキーマ．リスト b ～ d が回避する代表的な手法．XML スキーマの minOccurs 属性値は当該要素の最小出現回数，maxOccurs 属性値は最大出現回数を定義する．その他の構文については W3C XML Schema の仕様書を参照．

・any 要素とする前後に出現する要素（兄弟要素）の出現回数を固定する（第 3 図リスト c）.
・any 要素の出現場所を一段階深掘りして単独の子要素となるようにする（第 3 図リスト d）.

気象庁 XML では，名前空間を指定するか「このスキーマが属している名前空間以外」（“##other”）のいずれかになるようにしている.

なお，UPA 制約は根深く，“##other”指定により要素追加における UPA 制約違反を回避したとしても，メジャーバージョンアップによる XML スキーマの整理の段階で再び UPA 制約に抵触する可能性が高い．これを回避するためには整理の段階で他の名前空間により拡張したものを，定義元の名前空間に備え付け直す等の大規模な対応をする必要がある（第 4 図）.

a 要素辞書に要注意

UPA	XMLインスタンス	W3C XML Schema
×	<jmx_hoge:TemperaturePart> 　<jmx_eb:Temperature>14.5</jmx_eb:Temperature> </jmx_hoge:TemperaturePart>	【jmx_hoge】 <xsd:sequence> 　<xsd:element ref="jmx_eb:Temperature" minOccurs="0" maxOccurs="1"/> 　<xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##other"/> </xsd:sequence>
	Temperatureの名前空間が"jmx_hoge"と異なり，"jmx_eb"となっていることから，ベースとなっているXMLの名前空間と異なる．このため，anyのnamespace="##other"に抵触することに加えて，一つ前の要素がminOccurs="0"(省略可)となっていることから，スキーマから見るとjmx_eb:Temperatureをどちらにあたるのかが曖昧となる．	
○	<jmx_hoge:TemperaturePart> 　<jmx_eb:Temperature>14.5</jmx_eb:Temperature> </jmx_hoge:TemperaturePart>	【jmx_hoge】 <xsd:sequence> 　<xsd:element ref="jmx_eb:Temperature" **minOccurs="1"** maxOccurs="1"/> 　<xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##other"/> </xsd:sequence>
	jmx_eb:Temperatureがanyのnamespace="##other"に抵触するが，一つ前の要素がminOccurs="1"となっていることから，スキーマから見てanyではないことが判別できる．	

b 拡張後マイナーバージョンアップに要注意

UPA	XMLインスタンス	W3C XML Schema
×	<jmx_hoge:TemperaturePart> 　<jmx_eb:Temperature>14.5</jmx_eb:Temperature> 　<jmx_eb_add01:Humidity>30</jmx_eb_add01:Humidity> </jmx_hoge:TemperaturePart>	【jmx_hoge】(Ver1.00) <xsd:sequence> 　<xsd:element ref="jmx_eb:Temperature" minOccurs="1" maxOccurs="1"/> 　<xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##other"/> </xsd:sequence> 【jmx_hoge】(Ver1.01) <xsd:sequence> 　<xsd:element ref="jmx_eb:Temperature" minOccurs="1" maxOccurs="1"/> 　<xsd:element ref="jmx_eb_add01:Humidity" minOccurs="0" maxOccurs="1"/> 　<xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##other"/> </xsd:sequence>
	Ver1.00からマイナー拡張したVer1.01(名前空間は変更せず)のスキーマを作成したが，拡張部分であるjmx_eb_add01:Humidityの出現回数がminOccurs="0"(省略可)となっていることから，スキーマから見るとjmx_eb_add01:Humidityがanyにあたるのかが曖昧となる．	
○	<jmx_hoge:TemperaturePart> 　<jmx_eb:Temperature>14.5</jmx_eb:Temperature> 　<jmx_eb_add01:Humidity>30</jmx_eb_add01:Humidity> </jmx_hoge:TemperaturePart>	【jmx_hoge】(Ver1.00) <xsd:sequence> 　<xsd:element ref="jmx_eb:Temperature" minOccurs="1" maxOccurs="1"/> 　<xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##other"/> </xsd:sequence> 【jmx_hoge】(Ver1.01) <xsd:sequence> 　<xsd:element ref="jmx_eb:Temperature" minOccurs="1" maxOccurs="1"/> 　<xsd:element ref="jmx_eb_add01:Humidity" **minOccurs="1"** maxOccurs="1"/> 　<xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##other"/> </xsd:sequence>
	jmx_eb_add01:Humidityがanyのnamespace="##other"に抵触するが，一つ前の要素がminOccurs="1"となっていることから，スキーマから見てanyではないことが判別できる．	
○	<jmx_hoge:TemperaturePart> 　<jmx_eb:Temperature>14.5</jmx_eb:Temperature> 　<jmx_eb_add01:Humidity>30</jmx_eb_add01:Humidity> </jmx_hoge:TemperaturePart>	【jmx_hoge】(Ver1.00) <xsd:sequence> 　<xsd:element ref="jmx_eb:Temperature" minOccurs="1" maxOccurs="1"/> 　<xsd:any minOccurs="0" maxOccurs="unbounded" namespace="##other"/> </xsd:sequence>
	そもそもマイナーバージョンアップのスキーマを作成しない．	

第 4 図　XML フォーマットの拡張時等に発生しやすい UPA 制約違反とその回避例

c 要素の拡張は親タグごと

<table>
<tr><th>UPA</th><th>XMLインスタンス</th><th>W3C XML Schema</th></tr>
<tr><td rowspan="2">○</td><td><jmx_hoge:TemperaturePart>
　<jmx_eb:Temperature>14.5</jmx_eb:Temperature>
</jmx_hoge:TemperaturePart>
<jmx_hoge:AnyElementPart>
　<jmx_eb:WindSpeed>14.5</jmx_eb:WindSpeed>
</jmx_hoge:AnyElementPart>
<jmx_hoge:AnyElementPart>
　<jmx_eb_add01:Humidity>14.5</jmx_eb_add01:Humidity>
</jmx_hoge:AnyElementPart></td><td>【jmx_hoge】(Ver1.00)
<xsd:sequence>
　<xsd:element name="TemperaturePart" minOccurs="0" maxOccurs="unbounded" type="jmx_hoge:type.TemperaturePart"/>
　<xsd:element name="AnyElementPart" minOccurs="0" maxOccurs="unbounded" type="jmx_hoge:type.AnyElementPart"/>
</xsd:sequence>
<xsd:complexType name="type.TemperaturePart">

　<xsd:sequence>
　　<xsd:element ref="jmx_eb:Temperature" minOccurs="1" maxOccurs="unbounded"/>
　</xsd:sequence>

</xsd:complexType>
<xsd:complexType name="type.AnyElementPart">

　<xsd:sequence>
　　<xsd:any minOccurs="1" maxOccurs="unbounded" namespace="##other"/>
　</xsd:sequence>

</xsd:complexType></td></tr>
<tr><td colspan="2">anyを入れる箱としての親要素を用意し，それの出現個数を任意とすると，安定した拡張性が望める．
基本バージョン(ver1.00)の要素辞書(jmx_eb)を追加することも出来るし，追加バージョン(ver1.01)の要素辞書(jmx_eb_add01)を追加することも出来る．</td></tr>
<tr><td rowspan="2">※</td><td><jmx_hoge:TemperaturePart>
　<jmx_eb:Temperature>14.5</jmx_eb:Temperature>
</jmx_hoge:TemperaturePart>
<jmx_hoge:AnyElementPart>
　<jmx_eb:WindSpeed>14.5</jmx_eb:WindSpeed>
</jmx_hoge:AnyElementPart>
<jmx_hoge:AnyElementPart>
　<jmx_eb_add01:Humidity>14.5</jmx_eb_add01:Humidity>
</jmx_hoge:AnyElementPart></td><td>【jmx_hoge】(Ver1.10)
<xsd:sequence>
　<xsd:element name="TemperaturePart" minOccurs="0" maxOccurs="unbounded" type="jmx_hoge:type.TemperaturePart"/>
　<xsd:element name="AnyElementPart" minOccurs="0" maxOccurs="unbounded" type="jmx_hoge:type.AnyElementPart"/>
</xsd:sequence>
<xsd:complexType name="type.TemperaturePart">

　<xsd:sequence>
　　<xsd:element ref="jmx_eb:Temperature" minOccurs="1" maxOccurs="unbounded"/>
　</xsd:sequence>

</xsd:complexType>
<xsd:complexType name="type.AnyElementPart">

　<xsd:sequence>
　　<xsd:choice>
　　　<xsd:element ref="jmx_eb:WindLocation" minOccurs="1" maxOccurs="1"/>
　　　<xsd:element ref="jmx_eb_add01:Humidity" minOccurs="1" maxOccurs="1"/>
　　　<xsd:any minOccurs="1" maxOccurs="unbounded" namespace="##other"/>
　　</xsd:choice>
　</xsd:sequence>

</xsd:complexType></td></tr>
<tr><td colspan="2">jmx_hoge:AnyElementPart以下に現れる要素として，後から追加されたjmx_eb:WindSpeedとjmx_eb_add01:Humidityを明確にするためのマイナーバージョンアップ(名前空間は同じ)は行える．
ただし，厳密に定義するためにはxsd:choiceを辞書として記述できないといけないことから，現状では対応できていない．</td></tr>
<tr><td rowspan="2">○</td><td><jmx_hoge2:TemperaturePart>
　<jmx_eb:Temperature>14.5</jmx_eb:Temperature>
</jmx_hoge2:TemperaturePart>
<jmx_hoge2:WindSpeedPart>
　<jmx_eb:WindSpeed>14.5</jmx_eb:WindSpeed>
</jmx_hoge2:WindSpeedPart>
<jmx_hoge2:HumidityPart>
　<jmx_eb_add01:Humidity>14.5</jmx_eb_add01:Humidity>
</jmx_hoge2:HumidityPart></td><td>【jmx_hoge2】(Ver2.00)
<xsd:sequence>
　<xsd:element name="TemperaturePart" minOccurs="0" maxOccurs="unbounded" type="jmx_hoge2:type.TemperaturePart"/>
　<xsd:element name="WindSpeedPart" minOccurs="0" maxOccurs="unbounded" type="jmx_hoge2:type.WindSpeedPart"/>
　<xsd:element name="HumidityPart" minOccurs="0" maxOccurs="unbounded" type="jmx_hoge2:type.HumidityPart"/>
　<xsd:element name="AnyElementPart" minOccurs="0" maxOccurs="unbounded" type="jmx_hoge2:type.AnyElementPart"/>
</xsd:sequence>
<xsd:complexType name="type.TemperaturePart">

　<xsd:sequence>
　　<xsd:element ref="jmx_eb:Temperature" minOccurs="1" maxOccurs="unbounded"/>
　</xsd:sequence>

</xsd:complexType>
<xsd:complexType name="type.WindSpeedPart">

　<xsd:sequence>
　　<xsd:element ref="jmx_eb:WindSpeed" minOccurs="1" maxOccurs="unbounded"/>
　</xsd:sequence>

</xsd:complexType>
<xsd:complexType name="type.HumidityPart">

　<xsd:sequence>
　　<xsd:element ref="jmx_eb_add01:Humidity" minOccurs="1" maxOccurs="unbounded"/>
　</xsd:sequence>

</xsd:complexType>
<xsd:complexType name="type.AnyElementPart">

　<xsd:sequence>
　　<xsd:any minOccurs="1" maxOccurs="unbounded" namespace="##other"/>
　</xsd:sequence>

</xsd:complexType></td></tr>
<tr><td colspan="2">メジャーバージョンアップを図り，WindSpeedPartもHumidityPartも新設して新たな名前空間を与える．</td></tr>
</table>

第 4 図　続き

5 辞書と XML スキーマ*

5.1 辞書

気象庁 XML は，これまで各業務が独自に定めてきた情報の形式を統一して作成する標準フォーマットであるため，可能な限り構造の共通化を行った．XML スキーマは，前述のとおり妥当性検証等の機能による共通化の作業の効率化や，利用者側のシステム開発の自動化など機械処理において利用価値が高いが，本質的に機械可読のための形式であり，人間による作業においては利用しづらい．

気象庁 XML の策定作業において，各担当者が W3C XML Schema について十分な知識があるわけではなかったことや，利用者も気象庁 XML の構造について理解しやすいよう，人間可読性を重視するため，マイクロソフトエクセル形式による辞書を XML スキーマとともに作成することとした．なお，社会的にはこの「辞書」を「XML 要素辞書」「項目表」と呼ぶ場合もある．

辞書は単に人間にとって読みやすくするだけでなく，次の観点でも利用する．

- 大規模な XML 構造，多種の XML 構造を整理するため．
- 現モデルを新モデルにマッピングする際に，各項目を摺り合わせる（ハーモナイゼーションともいう．）目的で利用するため．
- XML スキーマは原則人間が読むものではない（機械可読形式）ので，人が整理して理解するため．

気象庁 XML の辞書形式は，親要素定義，個別の各子要素定義，属性定義，要素値・属性値のとりうる値のそれぞれにおいて，1 行ずつ記載していく形式となっている．また，XML スキーマの定義と辞書の構造に関する定義については必ず対応している．これは辞書を元として XML スキーマをツールにより自動生成しているからである．一方，子要素や属性のとりうる値については，辞書上の「とりうる値」列に記載されている値であっても，XML スキーマには反映しないことが可能な仕様となっている．この点については，次項のとおり XML スキーマを厳密に定義しないことによる利点があるためである．

辞書の書き方，XML スキーマの自動生成ツールに関する詳細については第 12.1 節を参照頂きたい．

5.2 XML スキーマとオフライン仕様

XML スキーマは XML の構造を適切かつ厳密に表現することが可能となる．更に XML インスタンスの妥当性検証により，その XML インスタンスが XML スキーマに適合しているかを厳密に確認することができる．

このことは非常に強力な妥当性の検証機となる反面，XML スキーマを厳密に定義することは，XML 構造の変更といったバージョンアップや，状況変化に伴うとりうる値（enumeration）の追加においても，強力な妥当性検証により不適合との判断を下される可能性が高いことを示す．また，その対応のためにシステム改修が多数発生することが予測される．XML 電文に紐付く業務の大きな変更であればシステム改修が発生するのは当然と言えるが，業務や修正点の規模に関わらず一律システム修正が必要となるのは当庁の本意ではない．このため，XML の構造変化とならないとりうる値の変更等において XML スキーマの変更とならないよう，XML スキーマにおいてその値を反映しない運用とし，一方で値の確認が必要な人のためにその内容が分かるようにした仕様を提示できるようにした．このような仕様をオフライン仕様と呼ぶこととし，気象庁 XML においては仕様書本文や辞書，運用指針等によりオフライン仕様を明確化している．

このように，気象庁 XML では XML スキーマではなく，オフライン仕様により定義している要件も多い．しかしながら，何もかもをオフライン仕様とすると今度は曖昧な仕様となりかねない．このため，XML スキーマで定義するか，オフライン仕様で定義するかについては要件個別に検討を行っている．具体的な使い分けについて，以下にまとめる．

---

* 第 5 章　杉山

(1) とりうる値

とりうる値（enumeration）については，前項のとおり，XML スキーマで定義可能な値であるが，オフライン仕様として辞書やコード表にて示している場合が多い．基本的に以下のいずれかの場合は XML スキーマによる enumeration として定義し，それ以外は XML スキーマ側ではとりうる値の基底型を，オフライン仕様側ではとりうる値等詳細を定義としている．

・出現する値が明らかに確定している場合．
・とりうる値に追加・変更が発生するような業務変更自体の想定が大規模であり，いずれにせよシステム改修を伴うことが想定される場合．
・利用者による重要な検索キーとして利用されることが想定される場合．

(2) any 構文のとりうる名前空間値

W3C XML schema の any 要素（第 4.3 節（4）項参照）では namespace 属性により出現する任意の要素を指定できるが，これについては以下のルールによって，XML スキーマ側とオフライン仕様側とで定義を変えている．

・属する名前空間以外の名前空間全てを許容する場合（namespace 属性値“##other”），XML スキーマ側は namespace 属性値を“##other”とする定義のみを記載するが，オフライン仕様では想定している名前空間全てを併せて列挙する．

5.3 XML スキーマの注釈について

気象庁 XML の XML スキーマには注釈情報として以下の 3 点が含まれている．

(1) documentation 要素値

annotation 要素下の documentation 要素値には，対象となるスキーマに関連する以下の情報を含めている．

・気象庁防災情報 XML フォーマットの一部である旨の表記と権利表記．
・更新履歴（バージョン情報，日時，変更概要）

なお，運用指針 Ver1.2 ではマイナーバージョンアップでも変更履歴を記載となっているが実際にはしていない．詳細は第 10.2 節参照．

(2) import 要素の次に出現するコメントアウト

import 要素の次に出現するコメント記載に「Network Schema Location」として，コメントアウトした import 要素行が並ぶ．これは通常の有効な import 行が XML スキーマの名前空間と同名のローカルに保存されているスキーマファイルを読み込む設定となっているのに対して，コメントアウトした import 要素にはインターネットにより取得可能な XML スキーマファイルの取得先を記述している．システム的にインターネットからスキーマファイルを取得したい場合は，当該 URL にアクセスするようにしておけばよいが，取得できない場合にシステムが停止するリスクを考えると推奨はせず，あくまでスキーマの所在位置を示しているものとして理解した方がよい．

(3) Enumeration's コメント記載

末尾に「Enumeration's」としてコメントした simpleType 要素行が並ぶ．これは辞書からスキーマを自動生成する際に，とりうる値（enumeration）として定義した値の一覧となる．

6 気象庁 XML の概要*

気象庁 XML は，気象庁における各種防災情報を一元的に取り扱うための情報基盤として，電文フォーマットに XML 形式を採用し，仕様と管理を一元化している．

情報はフォーマットの運用管理を明確化するため，構造化と併せる形で名前空間を分けている．具体的には，電文全体を「管理部」「ヘッダ部」「内容部」に分け，管理部を含めた全体構造で 1 つ，ヘッダ部で 1 つ，内容部に当たる気象要素固有の情報構成に対して 3 つ，及び共通気象要素と拡張用でそれぞれ 1 つずつの計 7 つの名前空間を利用している．これら名前空間を示す URL に対応する形で，気象庁 XML の HP を開設しており，最

* 第 6 章　杉山

新の XML スキーマも同 URL から入手可能となっている.

「管理部」は電文に関するメタ情報を提供する.「ヘッダ部」は防災情報を各電文共通的に取得可能な部分であり，この部分は電文間で画一的な処理により同等の情報を適切に取得できるようにしている．これにより，情報の主たる部分についてその選別処理や取得処理は共通化が可能となっている.「内容部」は情報固有の詳細な情報を含めるところであり，この部分についてはむしろ情報ごとに独立した構成となっている．しかしながら，電文間に共通する気象要素については共通辞書（追加要素）及びそれらを組み合わせた型（サブセット）を共用して処理の流用・統一化が図れるようにしている.

使われる値や型も可能な限り共通化を図っている．一般的に使われる時刻情報や地理空間情報についてはそれぞれ W3C XML Schema の Datatypes : dateTime 型 [7] と ISO6709 といった国際的な規格を利用することで共通化している．また，気象要素値を示す要素の構造についても，示す気象要素に関係なく持ちうる属性が同じであり，同じ意味を持つように共通化している.

このような大構造の共通化のほか，詳細部分の共通化を図るほか，バージョン管理や名前空間管理の方針を事前に示すことにより，フォーマットそのものの運用管理体系を分かりやすくし，利用者における拡張等の準備を行えるようにもしており，これらの対応により利用者における気象庁 XML の処理が平易となるよう考慮している.

7　全体構造の解説*

7.1　基本構造

(1) 文字コード（仕様 1.3.1 項）

文字コードは「UTF-8」を使う.

当初の庁内の議論においては，これまで利用してきた歴史的経緯から Shift_JIS（JIS X 0208 附属書 1 シフト符号化表現）が優勢であったが，XML 規格 [6] 2.2 項において「使用できる文字は，タブ（Tab）・復帰（CR）・改行（LF）・Unicode・ISO/IEC 10646（UCS）とする.」（原文は英語）となっており，また W3C による XML 日本語プロファイル [9]（JIS でも TR X 0015:2002 にて規定）においても「文字符号化スキームは UTF-16 か UTF-8 を推奨する.」（原文は英語）となっていることから，将来性等を考慮して UTF-8 を利用することとした.

また，UTF-8 には半角カナと NEC 特殊文字，及び IBM 拡張文字が含まれるが，運用指針 6 項にあるとおり，これらは使用しない.

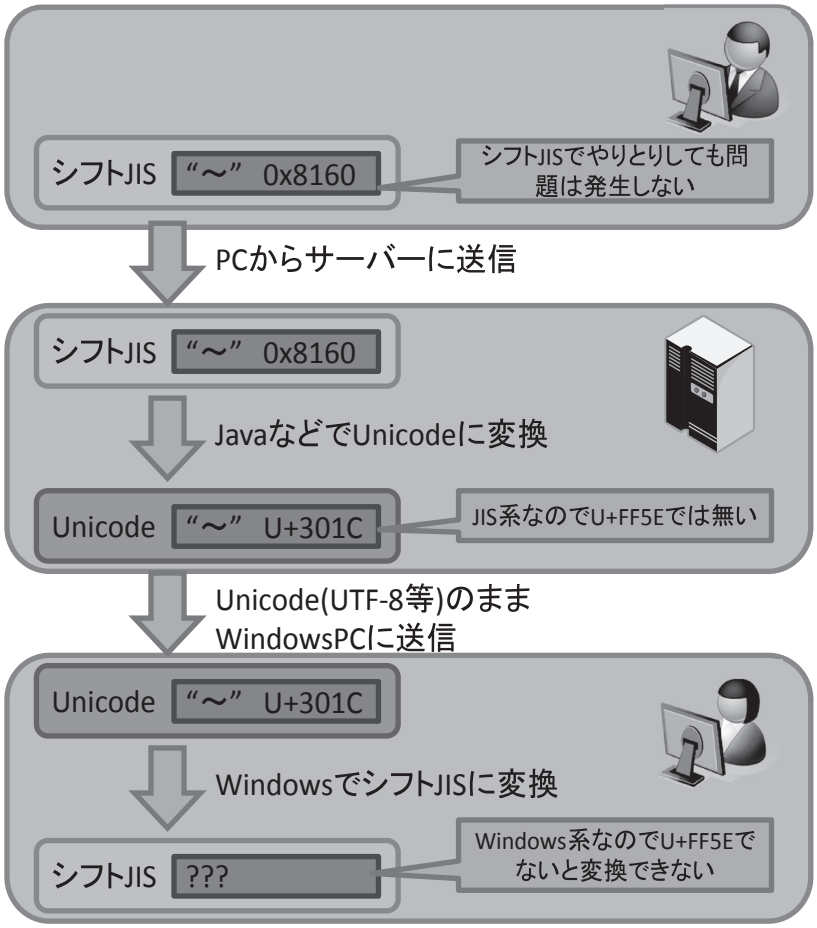
UTF-8 は Unicode のエンコーディング形式の一つであるが，この Unicode においてはシステム・ソフトウェア間における変換表相違問題がある．これは Unicode が似た形象の文字を複数定義しており，また Shift_JIS や EUC（Extended Unix Code）における各文字と Unicode 上での文字の対応について，規格としては説明しなかったことにより，システムやソフトウェア間で変換ルールが一致しなくなったためである．具体的には大きく分けて CP932 に代表されるマイクロソフト系と JIS 系に分かれ，この間で変換ルールが異なることから，Shift_JIS・EUC → Unicode，Unicode → Shift_JIS・EUC のように変換を 2 度行うと問題となりやすい．Shift_JIS を例とすると，具体的な影響は次のとおりとなる.（第 1 表，第 5 図）

- Shift_JIS → Unicode を行う場合には特段の問題は発生しない．ただし，システム・ソフトウェアによる変換ルールの差異により，作成される Unicode のコード値は個々に異なる場合が発生する．またこのため，判定・文字列解析を行う場合はどのコード値かを意識する必要はある.
- Unicode → Shift_JIS を行う場合，変換元の Unicode コード値に対応する変換テーブルで変換する場合は問題が発生しないが，対応する変換テーブルでない場合，Shift_JIS に変換できないコード値が存在することからその文字は文字化け（一般的には“?”や“〓”になる）となる．例えば U+00A5 から Shift_JIS に変換するテーブルは CP932 にはなく JIS 規

* 第 7 章　杉山，　第 7.1 節（8）項　杉山・竹田，

第 1 表　Unicode とシフト JIS の変換におけるマッピングの差異の例

文字	シフトJIS	JIS規格(Java/nkf等)	CP932(Windows変換)
¥	0x5C(YEN SIGN/円記号)	U+00A5(YEN SIGN)	U+005C(REVERSE SOLIDUS)
~	0x7E(OVERLINE/オーバライン)	U+203E(OVERLINE)	U+007E(TILDE)
—	0x815C(EM DASH/ダッシュ(全角))	U+2014(EM DASH)	U+2015(HORIZONTAL BAR)
〜	0x8160(WAVE DASH/波ダッシュ)	U+301C(WAVE DASH)	U+FF5E(FULLWIDTH TILDE)
‖	0x8161(DOUBLE VERTICAL LINE/双柱)	U+2016(DOUBLE VERTICAL LINE)	U+2225(PARALLEL TO)
−	0x817C(MINUS SIGN/負符号，減算記号)	U+2212(MINUS SIGN)	U+FF0D(FULLWIDTH HYPHEN-MINUS)
¢	0x8191(CENT SIGN/セント記号)	U+00A2(CENT SIGN)	U+FFE0(FULLWIDTH CENT SIGN)
£	0x8192(POUND SIGN/ポンド記号)	U+00A3(POUND SIGN)	U+FFE1(FULLWIDTH POUND SIGN)
¬	0x081CA(NOT SIGN/否定)	U+00AC(NOT SIGN)	U+FFE2(FULLWIDTH NOT SIGN)



第 5 図　Unicode とシフト JIS の相互変換による文字化け発生の原因

格にしかないことから，JIS 規格変換テーブルを用いるか，個別に U+00A5 → U+005C にする文字個別処理をした上で CP932 による処理を行う必要がある．

・Unicode → Unicode の場合，つまり Unicode の文字を Unicode の処理・表示系で取り扱う場合は，一般的にマイクロソフト系，JIS 系にかかわらず Unicode の全ての文字列について，表示するための字体がシステムに用意されていることから，コード値の違いによらず，画面等では問題なく表示することが可能となる．この場合，見た目上は似て非なる文字が表示されていることから，判定・文字列解析を行う場合はどのコード値かを意識する必要はある．

変換ルールに関する詳細は JIS TR X 0015:1999[10] を参照のこと．

このことについて，実装後に各電文における作成コード値を確認した．この結果，問題となる文字についてはおおむね使用しないか使用禁止文字となっていたが，ダッシュ“―”，波ダッシュ“～”，負記号“−”の 3 文字については，電文単位では統一された利用となるものの，電文間ではマイクロソフト系，JIS 系，使用しないと利用状況が個別にばらばらとなった．このことから，運用指針 6.2 項にもあるとおり，利用状況を説明することにより，利用者におけるシステム動作不良の軽減を図ることとした．

なお，将来的にはコード値の統一が望ましい．

（2）改行コード（仕様 1.3.1 項）

改行コードは「改行」（LF：0x0A）を使う．

なお，XML 規格 [6]2.11 項では「入力された（文書実体を含む）外部解析対象実体の中の全ての行末（連続する 2 文字 #xD #xA，及び #xA を後ろに伴わない #xD）を，解析を行う前に XML プロセサが単一の #xA に変換することによって正規化する処理とする．」（原文は英語）となっている．

（3）要素と属性の使い分け（仕様 1.3.2 項）

基本的に要素名により構造化を行い，属性については要素値に対するメタデータを与える．

旧形式府県天気予報 XML[11] では，以下の理由により key-value 型（Map 型）の構造を多用していた（第 6 図）．

・要素名を固定，属性値を可変とすることにより XML 構造を固定しつつ拡張性，汎用性を高めることが可能である．

・XML 構造を固定することによりスキーマの更新を極力無くすことが可能である．

・リレーショナルデータベースへの格納においてはシンプルな構造とすることがベターである．

しかしながら，XML コンソーシアムとの議論では以下の意見が出された．

・要素値の意味を属性値に持たせる場合，属性値抜きでは全く理解できず洗練性が低く見える．

・値を取得するために属性値を検索キーとすることに変わりなく，プログラム等の改修が必要なことには差がない．

・スキーマの更新は，XML のバージョン管理をバリデーションにより可能とするという観点もあることから，むしろ利点と見て積極的に利用する考え方もある．

・属性値を多用する構造が多く見られたのは DTD を利用していた経緯によるところが多く，最近は要素での構造化に主流が移っている．

これらの議論により，XML コンソーシアムからの意見を重要視して，気象庁 XML では key-

旧形式府県天気予報系XMLの例

```
<property name="特定域風">
  <t>
    <v k="地域名">石巻地域/陸上</v>
    <v k="風向">西</v>
    <v k="風速">4</v>
    <s></s>
    <v k="地域名">石巻地域/海上</v>
    <v k="風向">南西</v>
    <v k="風速">4</v>
  </t>
      :
```

第 6 図　Key-Value Store 型（KVS 型）XML 格納形式の例

例は旧形式府県天気予報系 XML（名前空間“ http://adess.kishou.go.jp/xml10”）より．要素 v の属性 k の値が Key にあたり，その Key の示す値が v の要素値となる．

value 型を採用しないこととした.

要素名により構造化することにより，属性値についてはメタデータを与える．考え方として，属性値がなくても構造上理解できるようなもののみを属性値とすることとした．具体的には以下のとおりである.

・要素値の単位，コード体系
・要素値に対する補足的・付加的情報
・要素値に関する人間可読的な情報

要素値の単位，コード体系については，基本的になくても理解できるよう，常に同じ情報種別の XML 電文における同じ構造の要素下においては同じ属性値が出現することを原則とする.ただし，この原則について，実際はかなり緩く運用されており，同一要素名において単位系を示す属性値を複数併記して，同一物理値の別表現として併用している電文もある．(第 7 図)

(4) XML 構造と言語（仕様 1.3.3 項関連）

要素名，属性名については英数字を用いる．それぞれ，それら要素・属性を意味する英語によることとし，複数単語からなる場合については，要素名は Upper Camel Case，属性名は Lower Camel Case により記述する（第 8 図）．英単語は読みやすさを重視し原則として省略しない．ただし，頻繁に出現し,かつ文字数が多い単語についてのみ,誤解を生じない範囲での省略形の利用を認める.この場合，単語の省略を行っていることを明示する.

名前空間については，W3C 勧告 XML 名前空間 1.1[12] 2.3 項「%-escaped された文字を名前空間で使用しないことを強く推奨する」(原文は英語）となっており，URI で非 ASCII 文字を用いる場合は %-escaped する必要がある（IETF RFC 3986[13]）ことからも，日本語及び URI 記法に変換した日本語を利用しない.

要素値，属性値については言語の制限を設けない.

(5) 要素のネスティング（仕様 1.3.3 項関連）

要素のネスティング（入れ子）については，基本的に以下のルールによる.

ア　地理空間，時間空間，防災・観測要素のそれぞれは並行概念として構造化し，親子関係としない.

これらの概念はそもそもお互いに独立しているものであるが，これまでは情報の特性に合わせて親子関係としてきたものが多い．今回の気象庁 XML においては，汎用性を高めること，及び検索時にどの要素で検索する場合でも同等の DOM ツリーの走査により検索できるよう，これらは並行概念として構造化する（第 9 図）.

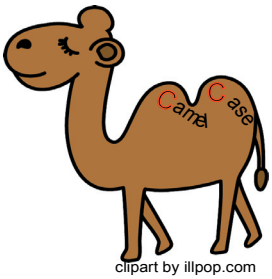
イ　地理空間において，行政区画や防災上の区画構成を元とした親子関係を原則として行わない.

いわゆる気象庁予報警報規程における地方予報区，府県予報区，一次細分区等の区画構成の上級，下級といった区画構成の概念を，そのまま XML における親子関係としない（第 9 図）．このような構成は取扱い上，一見合理的に見えるが，実際に情報の取扱いを検討した場合には，以下のとおりとなることが多い.

・その場の状況により利用者が欲している区域構成の粒度は必ず決まっており，親子階層を

```
台風指示報より
    <jmx_eb:WindSpeed type="最大風速" unit="ノット" condition="中心付近" description="中心付近の最大風速35ノット">35</jmx_eb:WindSpeed>
    <jmx_eb:WindSpeed type="最大風速" unit="m/s" condition="中心付近" description="中心付近の最大風速18メートル">18</jmx_eb:WindSpeed>
    <jmx_eb:WindSpeed type="最大瞬間風速" unit="ノット" description="最大瞬間風速50ノット">50</jmx_eb:WindSpeed>
    <jmx_eb:WindSpeed type="最大瞬間風速" unit="m/s" description="最大瞬間風速25メートル">25</jmx_eb:WindSpeed>
```

第 7 図　単位系の使い分けにより同一物理値の別表現として併用している例
unit 属性値によりノットで値を取得する場合と，m/s で値を取得する場合で併用している．ただし，出現順序等は固定しているので，値を取り間違う等は発生しない.



第 8 図　Camel Case の図

掘り下げて走査する利点がない．

・利用者が欲している区域構成の粒度が決まっていることから，むしろ粒度別に取りまとめた方が利用しやすい場合がある．

・全体的に見て，深掘り構造は利用者としても XML 技術としても取り扱いづらい．

このため原則としては区域構成を元とした親子関係は行わない．逆に実際の取扱い上，上記利点・欠点がむしろ逆となる場合は，例外的に親子関係とした構造化としても構わない．

ウ　同一要素を繰り返す場合，取り扱いの平易化やインスタンスの明確化を図るために，上位に繰り返しを束ねる要素を置く．この場合の束ねる要素名は束ねられる要素名を複数形化した単語とする（第 10 図リスト a）．

繰り返し出現する要素の XML 的取扱いのデファクトスタンダードはない．例えば，JavaEE における servlet deployment descriptor（配備記述子：いわゆる web.xml）においては，単純に必要な回数分同一要素を繰り返して記述する（第 10 図リスト b）．一方，XSLT の for-each 構文や DOM に対する iterator を考えると繰り返される要素はひとくくりとしてグルーピングした方が取扱いは平易となる．いずれも観点が違うだけなのでどちらが良い悪いはないが，利用しやすさと構造把握の平易さを優先し，親要素にて束ねるものとする．

(6) 呼び出した要素の名前空間の属し方

共通要素をまとめた要素辞書（第 7.2 節 (2) 項参照）のように，ある構造の中で他の名前空間

リストa：親子関係としてきた例（旧形式気象警報・注意報XML）

```
<area code="8500" name="佐賀県">
  <area code="8510" name="南部">
    <area code="8511" name="佐賀多久地区">
      <kind level="注意報" name="雷" type="継続">
        <property endTime="27日昼過ぎ" name="雷"
              overTime="にかけて　以後も続く"/>
        <addition name="付加事項">竜巻</addition>
      </kind>
      <kind level="注意報" name="強風" type="継続">
        <property direction1="南" endTime="27日昼過ぎ"
             name="風" peakTime="26日昼過ぎ">
          <content areaName="陸上" name="最大風速"
              unit="メートル">12</content>
          <content areaName="海上" name="最大風速"
              unit="メートル">12</content>
        </property>
      </kind>
      <kind level="注意報" name="波浪" type="継続">
        <property endTime="27日明け方" name="波">
          <content name="波高"
              unit="メートル">1.5</content>
        </property>
      </kind>
    </area>
  <area code="8512" name="鳥栖地区">
        :
```

リストb：並行概念とした例（気象庁XML）

```
<Item>
  <Kind>
    <Name>波浪注意報</Name>
    <Code>16</Code>
    <Status>発表</Status>
    <Property>
      <Type>波</Type>
      <AdvisoryPeriod>
        <EndTime>
          <Date>3日</Date>
          <Term>夕方</Term>
        </EndTime>
        <OverTime>にかけて　以後も続く</OverTime>
      </AdvisoryPeriod>
      <PeakTime>
        <Date>2日</Date>
        <Term>夜のはじめ頃</Term>
      </PeakTime>
      <WaveHeightPart>
        <Base>
          <jmx_eb:WaveHeight type="波高" unit="m"
              description="2.5メートル"
            >2.5</jmx_eb:WaveHeight>
        </Base>
      </WaveHeightPart>
    </Property>
  </Kind>
  <Area>
    <Name>富山市</Name>
    <Code>1620100</Code>
  </Area>
  <ChangeStatus>警報・注意報種別に変化有</ChangeStatus>
</Item>
  :
```

第 9 図　要素のネスティング構造

リスト a は旧形式気象警報・注意報 XML，リスト b は気象庁 XML．リスト a では地域を示す area 要素が細分区構造に併せてネスティング構造をしている上に，警報・注意報の種別を示す kind 要素がその子要素をなして，更に property 要素にて時間情報を補足している．リスト b では大きく分けて警報・注意報を大枠で表す Kind 要素と地理的情報である Area 要素から成り立ち，Kind 要素の中で警報・注意報を示す Name 要素と詳細情報である Property の下に各種時刻情報を含めている構造で，それぞれは並行要素となっており，ネスティング構造とはなっていない．

リストa:気象庁XML

```
<Headline>
 <Text> 5日06時55分ころ，地震による強い揺れを感じました．震度3以上が観測された地域をお知らせします．</Text>
 <Information type="震度速報">
  <Item>
   <Kind><Name>震度4</Name></Kind>
   <Areas codeType="地震情報／細分区域">
    <Area><Name>岩手県沿岸南部</Name><Code>211</Code></Area>
    <Area><Name>岩手県内陸北部</Name><Code>212</Code></Area>
    <Area><Name>宮城県北部</Name><Code>220</Code></Area>
    <Area><Name>宮城県中部</Name><Code>222</Code></Area>
   </Areas>
  </Item>
  <Item>
       :
```

リストb: JavaEEにおける配備記述子(web.xml)

```
<servlet>
 <servlet-name>default</servlet-name>
 <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
 <init-param>
  <param-name>debug</param-name>
  <param-value>0</param-value>
 </init-param>
 <init-param>
  <param-name>listings</param-name>
  <param-value>false</param-value>
 </init-param>
 <load-on-startup>1</load-on-startup>
</servlet>
<servlet>
 <servlet-name>jsp</servlet-name>
 <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
 <init-param>
  <param-name>fork</param-name>
  <param-value>false</param-value>
 </init-param>
 <init-param>
  <param-name>xpoweredBy</param-name>
  <param-value>false</param-value>
 </init-param>
 <load-on-startup>3</load-on-startup>
</servlet>
<servlet-mapping>
 <servlet-name>default</servlet-name>
 <url-pattern>/</url-pattern>
</servlet-mapping>
<servlet-mapping>
 <servlet-name>jsp</servlet-name>
 <url-pattern>*.jsp</url-pattern>
</servlet-mapping>
```

第 10 図　繰り返し構造の例

リスト a は気象庁 XML，リスト b は JavaEE における配備記述子の例．リスト a では，複数の Aera 要素を束ねる目的で Areas 要素が出現している．リスト b では，servlet 要素や init-param 要素，servlet-mapping 要素が複数回の並行出現がされているが束ねられていない．

に属する要素又は型を利用する際，利用したい要素の名前空間が呼び出した側に属するのか呼び出された側に属するのかは検討する点となる（第 11 図）．XML スキーマ（辞書）の記述について，呼び出した側の名前空間に要素が属する場合，呼び出した側の当該要素を定義する element 要素の type 属性値（基底型）が呼び出された側の complexType 要素の型名（親要素）とする型参照となり，呼び出された側の名前空間に要素が属する場合，呼び出される側で予め要素を定義した上で，呼び出した側はその要素名を element 要素の ref 属性値とする要素参照を用いる．

気象庁 XML では，気象要素等を示す要素名は，それを呼び出した状態に属するのではなく，一連の要素の定義として呼び出される側にまとまっておく方が自然であるとして，一律呼び出された側に属するようにしている．

(7) 単位表記（仕様 1.3.9 項）

物理量を示す各単位をテキスト表記する一元的手法は現在の所ないようである．一例として ISO2955（Information processing – Representation of SI and other units in systems with limited character sets）もあるが，これでも 2 乗分の 1 を示す表記として「m-2」と「1/m2」の両方が認められているなど，画一性に欠ける．また，気象庁としては SI 単位系に限らず慣用的に用いられている単位系も用いる必要がある．このことから，以下の記法を元に，その都度定めていくこととする．

- 原則として，SI 単位系を用い，単位を示す一般的な英数字表記を用いる（“m/s”等）．
- SI 単位系でなくとも一般的なものはその英数字表記を用いる（“kt”等）．
- 乗数を示す接頭辞も用いる（“cm”等）．
- 一般的でないもの，英数字表記が誤解を招きやすいものは日本語で表記する（“メートル”等）．
- 同一の単位系については気象庁 XML 内で同一の表記とする．

いずれにしても，同一の意味を示す単位が違う

辞書(名前空間)	親要素	子要素	属性	基底型	解説
jmx_mete	type.TemperaturePart1				要素参照のパターン
		jmx_eb:Temperature		jmx_eb:type.Temperature	
	type.TemperaturePart2				型参照のパターン
		Temperature		jmx_eb:type.Temperature	

jmx_eb	(element)	Temperature		type.Temperature	要素参照の場合は、この行を参照している. jmx_eb:Temperatureの型はtype.Temperatureとなる.
	type.Temperature			xs:string	型参照とjmx_eb:Temperatureはこの型を参照している.
			type	xs:string	
			unit	xs:string	
			refID	xs:unsignedByte	

要素参照のXML例

```
<jmx_mete:Property>
  <jmx_eb:Temperature type="" unit="" refID=""/>
</jmx_mete:Property>
```

要素参照の場合，要素名ごと相手辞書(ここでは要素辞書)を利用することになる.

型参照のXML例

```
<jmx_mete:Property>
  <jmx_mete:Temperature type="" unit="" refID=""/>
</jmx_mete:Property>
```

型参照の場合，器の構造たる型のみを相手辞書から参照利用することとなる.

第 11 図　名前空間が呼び出される側に属する場合と呼び出す側に属する場合の例

表現とならないように管理を徹底する必要がある．

(8) 特定の値の表現

台風や低気圧の移動速度や移動方向を表すときに，通常は「1 時間におよそ 20 キロの速さで西北西へ進んでいます」といった表現をする．これを XML で表現すると例えば次のように記述できる．

```
<Direction> 西北西 </Direction>
<Speed unit="km/h">20</Speed>
```

ところが，台風情報や全般海上警報では，移動速度が 5 ノット以下の時に移動方向が定まる場合は移動速度に「ゆっくり」と表現し，移動方向が定まらない場合は移動速度に「ほとんど停滞」と表現する．上記の移動速度を示す Speed 要素の値は，通常の場合であれば数値のため，型は float 型（浮動小数点型）と定義しておくのが適切であるが，「ゆっくり」や「ほとんど停滞」は float 型ではないため，辞書で Speed 要素の値を float 型で定義してしまうと，「ゆっくり」や「ほとんど停滞」を表現できない(妥当性検証が成功しない)．一方，「ゆっくり」や「ほとんど停滞」を表現するためだけに Speed 要素の値を string 型（文字列型）で定義すると，速度の値としては間違った文字列も妥当性検証で妥当とされてしまうので，適切ではない. 観測データの欠測に関しても同様で，次の例のとおり，欠測について，気象要素については空要素とし，Aqc 要素のように別の要素で状態を表すという案もあった．

```
<Snow>
 <Depth unit = "cm "></Depth>
  <Aqc> 欠測 or 計画休止 </Aqc>
</Snow>
```

このように特定の値の処理については手法も様々にあることから，XML コンソーシアムと担当者により議論を行った．

ある要素値として許容する特殊な値としては空タグ（空要素），“xsi:nil="true"”表記があり，基底型として許容する特殊な値として，“NaN”（非数），“INF”（正の無限大），“-INF”（負の無限大）などがある．これらと W3C XML Schema における各データタイプにおける対応状況表を第 2 表にまとめる．なお，“xsi:nil="true"”を利用した空タグ表記については実装間の差があり，属性値が取れない実装と取れる実装があったことから，属性値を利用する要素では利用しないことが望ましいことが判明した（詳細は第 11.1 節）．

これらの状況を踏まえ，対応方針として以下の 4 案を検討した．具体例については第 12 図，メリットデメリットについては第 3 表にまとめる．

案 1　負の最大値等特殊な値に意味を持たせる．

案 2　親要素を作成し，その下に値の要素と状況の要素と 2 つを用意して利用する．

案 2’親要素を作成せず，単に値の要素と状況の要素と 2 つを用意して利用する．

案 3　実質的な基底型と string 型を union で組み合わせる．

案 4　値を属性値とし，可視化すべき情報や状況を示す文言を要素値とした構造にする．

案 1 については本議論の直前に，当時運用していた電文の処理において利用者のシステムがダミー値を任意の値として読み取ってしまい，社会的に大きな影響を与える誤作動をした事例があったことや“NaN”は計算を成功させない値という強い意味を持つことからも，利用者のシステムにおける「処理の安定性」（プログラムの完全性）に不安があった．案 2，案 2’については，特定の値とする場合に，値を格納してきた要素そのものが省略となり，逆に処理の安定性に欠けるのではないかという意見があった．案 3 は空タグを許容する基底型を XML スキーマ union 構文により自作するというものであるが，java 処理系などが XML の要素値を変数に自動バインディングする際，基底型が string 型になってしまいメリットを生かせないという問題がある．案 4 については一見きれいな格納ができるように見えるものの，

第 2 表　特殊な値とデータタイプ

特殊な値	string	short	int	long	float	double
(空タグ)	○	×	×	×	×	×
(xsi:nil="true"の利用)	○※1	○※1	○※1	○※1	○※1	○※1
NaN	○※2	×	×	×	○	○
INF	○※2	×	×	×	○	○
象徴的な特定の値(負の最大値等)	○※2	○※3	○※3	○※3	○※3	○※3

注）※1属性値が取得できない実装がある.
※2文字列として認識.
※3数字として認識.

案1　負の最大値等特殊な値に意味を持たせる.
（例）
```
<Aelement>-2147483648</Aelement>
```

案2　親要素を作成し，その下に値の要素と状況の要素と2つを用意して利用する.
（例）
```
<PlumeHeight>
  <AvobeSeaLevel unit="FT">5800</AvobeSeaLevel>
  <Condition>雲に入る</Condition>
</PlumeHeight>
```

案2'親要素を作成せず，単に値の要素と状況の要素と2つを用意して利用する.
(<SpeedValue>と<SpeedCondtion>等)
（例）
```
<PlumeHeightAvobeSeaLevel unit="FT">5800</PlumeHeightAvobeSeaLevel>
<PlumeHeightCondition>雲に入る</PlumeHeightCondition>
```

案3　実質的な基底型とstring型をunionで組み合わせる

案4　値を属性値とし，可視化すべき情報や状況を示す文言を要素値とした構造にする.
（例）
```
<Speed unit="m/s" value="10.0">10メートル毎秒</Speed>
<Speed unit="m/s">不明</Speed>
```

第 12 図　特定の値に対する処理案の例

第 3 表　特定の値に対する処理案のメリット・デメリット.

案	方法	メリット	デメリット
案1	負の最大値等特殊な値に意味を持たせる	スキーマ的にはシンプル.	負の最大値という特殊な値を使うことの危険性がある. 特定の値に意味を持たせることの社会性に疑問がある.
案2	親要素を作成し，その下に値の要素と状況の要素と2つを用意して利用する	スキーマ的にはシンプル. 個別の要素値の取得に特殊な処理が不要. 誤った値を使う余地もなく，問題が発生する可能性は低い.	大量のデータがあると表現が冗長的となる. 値の要素が無いことの社会性に疑問がある.
案2'	親要素を作成せず，単に値の要素と状況の要素と2つを用意して利用する	スキーマ的にはシンプル. 個別の要素値の取得に特殊な処理が不要. 誤った値を使う余地もなく，問題が発生する可能性は低い.	値と状況の組み合わせが要素名でしか判別できないことの社会性に疑問 二つの値を併せて一つの状況を説明するなど，情報によっては利用に制限がある. 値の要素が無いことの社会性に疑問がある.
案3	実質的な基底型とstring型をunionで組み合わせる	見た目が非常にすっきりとする. 誤った値を使う余地もなく，問題が発生する可能性は低い	スキーマ的には煩雑となる. 個別の要素値の取得に特殊な処理が必要.
案4	値を属性値とし，可視化すべき情報や状況を示す文言を要素値とした構造にする	スキーマ的にはシンプル. 見た目はすっきりとする. 誤った値を使う余地もなく，問題が発生する可能性は低い.	この形式の社会性に疑問がある. 数値は数値として利用することを前提としてきたが，「可視化された情報」が要素値となることから全体を同じポリシーで再検討しなければいけない.

unit 属性の示す対象が要素値ではなく属性（例で言えば value 属性値）となり，属性間の修飾となることに違和感があるという意見があった．

これらのことを検討した結果，要素の自動バインディング自体は必須の要件でないことと，一度処理を作成すれば安定的に運用できることから，要素辞書（第 7.2 節（2）項参照）を含め，原則として案 3 を採用した．

また，特定の値を表現した場合の意味等の表現に際しては，当初電文により違った属性や構造を使っていた．数値の例外的処理を検討するに当たり，それまでに作成されたサンプル電文に使われていた属性や子要素を分類した結果，第 4 表のように，「例外の状態を示す属性や要素」，「数値の代替となる文字列」にまとめられると考えた．また，基本要素辞書においては，要素ごとに属性が異なるのは避けるべきと考え，統一の仕方について次のとおり 5 つの構造案を検討した．

案 1：
< 要素 type="" unit="" refID="" 状態 ="" 代替文字列 ="">
案 2：
< 要素 refID="">
<Type></Type>
<Unit></Unit>
< 状態 ></ 状態 >
< 代替文字列 ></ 代替文字列 >
</ 要素 >
案 3：
< 要素 type="" unit="" refID="">
< 要素 + 状態 >
< 要素 + 代替文字列 >
案 4：
< 要素 type="" unit="" refID="">
< 要素 + 代替文字列 状態 ="">
案 5：
< 要素 type="" unit="" refID="" 状態 ="" 数値 ="">

検討の結果，案 1 の方法を採用することとし，基本要素は，種別を示す type 属性，単位を示す unit 属性，参照番号を示す refID 属性に加え，状態などを示す condition 属性，文字列表現を示す description 属性からなる次のような基本形が決められ，これらの属性は，要素ごとに必要なものを設定することとなった．

○基本形
<ElementA
type="T" 同一基本要素の種別
unit="U" 単位
refID="R" 時系列の際の参照番号
condition="C" 値の状態
description="D" 文字列表現
> 値 </ElementA>

これらの議論をまとめて，特定の値を表現する場合には要素値は空タグで，その意味等を決められた属性値により示すものとした．すなわち，特定の値の表現は，空タグを許容する integer 型である nullableinteger 型（共通辞書（基本要素）上のグローバル型定義）と，同様の float 型である nullablefloat 型（同）により XML スキーマ上の定義として規定し，その意味等に当たる移動速度の「ゆっくり」や観測データの「欠測」などの表現は，condition 属性や description 属性によりオフライン仕様であるコード表や電文ごとの解説資料上の定義として規定している．この場合の属性値については第 7.2 節（2）項にまとめる．

第 4 表　基本要素の属性や子要素の分類

例外の状態を示す属性や要素	condition 属性，aqc 属性，bound 属性，PlumeHeightCondition 要素
数値の代替となる文字列	altString 属性，remark 属性，bound 属性
要素固有の属性	Coordinate 要素の datum 属性

(9) インデントのための空白文字列

XML インスタンスにおいて，見やすさを向上するためのインデントとして空白文字列を入れることがある. このような空白文字列を一般的には，無視できる空白文字列（Ignorable White Space）と呼んでいる．気象庁 XML においても，提示したサンプル電文では多用しているところであるが，実際の XML 電文中では利用していないことが多い．

実際の XML インスタンスにおいて，処理ソフトウェアが無視していいかどうかはインスタンスだけでは判別できず，XML スキーマによって判断することとなる．

7.2 全体構造

(1) 管理部・ヘッダ部・内容部（仕様 1.2.1 項）

気象庁 XML はルート要素（“Report”）の直下に管理部を示す“Control”要素，ヘッダ部を示す“Head”要素，内容部を示す“Body”要素の 3 要素から成り立つ．3 要素のそれぞれの意味は仕様書に記載されているとおりであるが，そもそもは情報をサマリー的情報としたヘッダ構造と本体構造に分ける 2 分割構造に，気象庁内部の従来の電文通信手順（いわゆる気象庁 TCP/IP ソケット手順・GTS 手順）においてデータ種類コード等で利用していた情報を伝送系の情報として加えて 3 要素としたものである．特に，これまでの気象庁の電文ではコンテンツとして示す時刻・地域情報と伝送情報として示す時刻・地域情報が明確に分離できておらず，情報の運用に困難な状況が発生していた．気象庁 XML では，伝送系における時刻情報・発表官署情報と，コンテンツとしての業務的な各種時刻情報・地域情報を分離した．これにより，業務系情報の充実化を図れると同時に，伝送系という形で情報作成者の主観が入らない信頼性の高い情報とに分離することが可能となった．この効果として，運用指針 2.1.3.4 項にあるとおり，情報の訂正，取消等の運用において，これまでの旧来の方法を整理し，情報運用の明確化，システム化対応が可能となった．また，ヘッダ部については各電文において共通的に取り扱えるように，出現要素や内容を画一化することとした．このため，一部の情報名称においてはヘッダ部と内容部で同一の気象要素等を表現していることがあるが，この点についてはシステム的に矛盾した情報が入ることがないように注意した上で，共通化に伴う許容点とした．なお，XML スキーマ仕様上，内容部を示す要素は要素名も任意となるが，運用上は各名前空間に属する“Body”名の要素とする．

(2) 要素辞書（仕様 1.3.10 項）

名前空間接頭辞を“jmx_eb”（“eb”は element basis の略）として定義している辞書を要素辞書と呼んでいる．要素辞書は各情報において共通的に用いられる地理空間要素や，気圧，気温，風向，風速，湿度，震度，マグニチュードといった物理量について，情報の種類によらずに利用できるよう要素コンポーネントとしてひとまとめにし，共通化したものである．

共通化に当たっては同じ物理量を同じ要素とするだけではなく，各コンポーネント間においても属性名称と意味が共通化するようにしている．共通化している属性名とその意味等を第 5 表のとおりまとめる．

また，時刻・時間表現においては仕様 1.3.4 項にあるとおり，W3C XML Schema にて定義されている dateTime 型，若しくは duration 型を用いることとなっており，要素辞書等における各コンポーネント・各要素が示す時刻・時間表現においても同様となっている．これら時刻・時間表現においても第 6 表のとおり共通化された属性名とその意味が利用されている．

(3) 追加辞書

名前空間接頭辞を“jmx_add”（“add”は addition の略）として定義している辞書を追加辞書と呼んでいる．追加辞書はバージョンアップの際に要素追加の短中期的影響を少なくするため，汎用的に用いる辞書である．利用の仕方については第 10.3 節にまとめる．

第5表　共通化された属性名と意味等

属性名	意味等
type	要素名が示す物理量等の使い分け的な種別を示す．検索キーとなりうる．共通の親要素の下（兄弟要素間）に同一要素名，type属性値が同一若しくは相違の要素値が複数存在する場合があるが，それぞれの関係が並行概念（いずれかを排他的に利用）か直列概念（足し併せて利用）かは情報の種別分類による．
unit	単位を示す．単位の記法については7.1(7)項を参照．検索キーとなりうるかどうかは情報の種別分類により，XML構造の特定部分については必ず対応が固定となる．例えば，同一要素名・同一type属性値で，unit属性値が“m/s”と“ノット”の要素がそれぞれある場合，利用者はいずれかを選択するための検索キーとするが，同一要素名・同一type属性値において要素が単一の場合，unit属性値は値に対する単位として参照利用することになる．この分類は情報の種別分類が同じであれば，構造が同じ所は常に同じ運用となる．
refID	時系列の際の参照番号を示す．検索キーとなりうる．詳細は仕様1.3.11項参照．
condition	要素値に「約」「以上」など，メタ的な情報を与える．また，要素値が無い（空タグ）場合に「不明」等，その意味を与える．
description	人間可読的な文字列表現を与える．一般的に機械処理による数値利用は要素値を，ウェブや印刷物など人間が読むことを前提にしている場合はdescription属性値を利用すると良い．

第6表　時刻・時間表現における共通化された属性名と意味等

属性名	意味等
significant	時刻表現は細かい定義が出来ることから，基底型で詳細を定義すると利用において煩雑となる．このため，基底型はdateTime型のみとし，その業務別の利用打ち切り精度をsignificant属性で示す．属性値のとりうる値とそれの示す時刻精度は要素辞書に書いてあるが，以下に抜粋する． ″yyyy″　有効部分が「″年″まで」であることを示す． ″yyyy-mm″　有効部分が「″月″まで」であることを示す． ″yyyy-mm-dd″　有効部分が「″日″まで」であることを示す． ″yyyy-mm-ddThh″　有効部分が「″時″まで」であることを示す． ″yyyy-mm-ddThh:mm″　有効部分が「″分″まで」であることを示す． ″yyyy-mm-ddThh:mm:ss″　有効部分が「″秒″まで」であることを示す． ″yyyy-mm-ddThh:mm:ss.sss″　有効部分が「″ミリ秒″まで」であることを示す． なお，ここで定義しているISO8601型の様な文字定義についてはこれをフォーマットとしてパースすることを期待するのではなく，単純な7パターンのトークンとして判別し，精度利用することを想定している．
precision	示す時刻に幅がある場合にその幅をduration型で示す．significant属性が打ち切り精度に対して，precision属性はエラー精度に近い．
dubious	時刻の示す曖昧さを文字表現する．基本的に「頃」しか用いない．

8 個別要素の解説*

8.1 管理部の個別要素

管理部の各要素は業務的な利用というよりも，システム的な自動処理系，配信系で必要となる情報を取りまとめている．

(1) DateTime 要素（仕様 1.6.4 項，運用指針 2.1.3 項）

DateTime 要素は発表時刻を示す要素である．旧来の気象庁における情報提供手段では，共通の時刻要素が「観測時刻」という位置づけとなっており，電文の実発信時刻を示すとは限らない．このため，時刻自体が何を示すのかは電文の情報種別によることとなり，また同一時刻表記の電文が複数流通する情報も多々あった．今回，一つの XML 電文の中で種々の時刻表記を複数種類系統立てて利用できるよう用意したことから，この DateTime 要素については純粋にシステム発信時刻という位置づけとなった．これは，いわゆるタイムスタンプとして，電文の順序整理に信用できるという非常に重要な意味を持つ．このため，この部分については人為的な誤びゅうが起きないよう（運用指針 2.1.3.6 項），適切にシステム設計を行う必要がある．

(2) Status 要素（仕様 1.6.1 項）

Status 要素は通常の電文，訓練電文，試験電文を区別する運用種別を示す要素である．従来電文では分野横断的には取り入れることのできなかったもので，気象庁 XML で初めて統一的に導入した運用に関する情報である．このような Status 要素を電文全体に対して意味を持つ管理部に含めることにより，運用利用して良い情報かどうかを平易に判断することが可能となる．また，運用利用に際しても，システム的な動作確認を行うことを目的とする「試験」状態と業務的な確認を行うことを目的とする「訓練」を分けることも行った．

当初，試験状態についてはその影響範囲を更に限定して「部内試験」「部外試験」「特定部外試験」として，配信システムにおける設定を Status 要素によって分けられないかという議論もあったが，試験用の設定では運用状態の確認にならないことから，Status 要素を判断して配信を分けることが適切かどうかという観点もあり，結果として現状のとおり「試験」は一通りのみとした．

なお，旧来の電文を気象庁 XML の形式により配信することを想定し，WMO 形式に則った運用種別情報（いわゆる BBB 部）の記述を含めているが，後方互換のためだけに利用するものであり，運用開始当初から気象庁 XML の各電文を含め，これから気象庁 XML 形式として新たに策定する電文においてこの形式を利用することはない．

(3) EditorialOffice 要素と PublishingOffice 要素（仕様 1.6.1 項）

処理系，配信系の検索キーとしては EditorialOffice 要素を用いる．このため気象庁本庁発信の電文において，EditorialOffice 要素は統一して「気象庁本庁」として発表する．一方，PublishingOffice 要素は業務的な発表官署の定義であることから，気象庁本庁内部部局間であっても統一しない．具体的な記述内容については運用指針別紙 2 を参考のこと．また，PublishingOffice 要素は複数部署の W3C XML Schema の list 型記法を認めている．これは文字列を空白文字により列挙する形式である．気象庁内の複数の部署による共同発表や，気象庁と都道府県や他の機関と共同で発表している防災気象情報では，発表機関を list 型で列挙することにより，これらの発表機関を個別表示等において分類する必要がある場合は空白文字を区切り文字（デリミタ）として値を取得すれば良く，特段分類する必要がない場合は空白文字も含めて 1 連の文字列として取り扱うことができる．

8.2 ヘッダ部の個別要素

ヘッダ部は各業務共通となる管理情報，及び警報事項等を統一的に取り扱うのに必要な情報を取

---

* 第 8 章　第 8.1 節・第 8.2 節（2）項（4）項～（7）項　杉山，第 8.2 節（3）項　横井・中村・竹田，第 8.2 節（1）項・第 8.3 節～第 8.4 節　竹田，第 8.5 節　清本，第 8.6 節　中村

りまとめている.

(1) 各時刻・時間要素（仕様 1.6.4 項）

日付時刻表記をする要素は W3C XML Schema のビルトインデータ型である dateTime 型を用いて定義されていたが，検討を進めると，dateTime 型では表現できない事例が出てきた.

生物季節観測電文の検討において，時刻が情報の性質上意味をなさない「観測日」を記述する必要が生じた．例えば 2008 年 3 月 22 日に観測した場合，「ISO8601:2004」では「2008-03-22」というような表現が可能であるが，dateTime 型では「2008-03-22」のような表現は妥当（valid）とはならないため，日時表現か辞書の基底型のどちらかに修正が必要である．また，火山観測報電文では，火山現象の時刻が第 7 表のように，観測精度により電文に表現する有効桁が変わる情報もある．更に天候情報では，「警戒期間　9 月 6 日頃からの約 1 週間」の「9 月 6 日頃」のような前後 1 日程度の日付時刻の曖昧さがある場合に，「頃」を用いて現象の開始や終了についての「曖昧さ」を考慮した時刻の幅を持たせたいものがある.

当初の検討段階において，生物季節観測電文における観測日については，「2008-03-22」形式の型であるビルトインデータ型の date 型で表記する方法と，観測日時は時刻を 0 時 0 分とした dateTime 型の要素と組となるビルトインデータ型の duration 型をとった要素の値に 1 日を示す「P1D」を入れる方法とが提案された．その後，火山観測報電文のように様々な有効桁で日付時刻を表記する必要が生じたことから，ビルトインデータ型として定義された time 型，date 型，gYearMonth 型等の dateTime 型以外の基底型を用いることについての議論も行った. しかしながら，気象庁 XML を利用する立場から考えると，java では日時を取得する場合に dateTime 型以外の型では実装に依存する可能性があることや，XML コンソーシアムからも，dateTime 型以外を利用することは勧められないとの助言があり，日付時刻を格納する要素の型は dateTime 型のみを用いることとなった.

天候情報等で出てくる「頃」の扱いについても検討を行った．異常天候早期警戒情報では，例えば，「警戒期間　9 月 6 日頃からの約 1 週間」と表現しているとき，「9 月 6 日には，当日を含めた前後 1 日，都合 3 日間程度の幅がある」という

第 7 表　観測精度により時刻の有効桁が変わる火山観測報電文の事例

	情報種別	とりうる値	解説	現在運用中の電文における用法	備考
観測	火山観測報	″10日12時14分″	火山現象の時刻が特定できる.	火 山:北海道駒ケ岳 日 時:2007年01月10日12時14分 (100314UTC) 現 象:噴火	
観測	火山観測報	″10日12時10分頃″	火山現象がその時刻近辺で発生しているが，時刻を特定できない. その幅についても，特定できない場合もある.	火 山:北海道駒ケ岳 日 時:2007年01月10日12時10分 (100310UTC)頃 現 象:噴火したもよう	
観測	火山観測報	″10日12時頃″	火山現象がその時刻近辺で発生しているが，時刻を特定できない. その幅についても，特定できない場合もある.	火 山:北海道駒ケ岳 日 時:2007年01月10日12時00分 (100300UTC)頃 現 象:噴火したもよう	
観測	火山観測報	″10日頃″	火山現象がその時刻近辺で発生しているが，時刻を特定できない. その幅についても，特定できない場合もある.	火 山:北海道駒ケ岳 日 時:2007年01月10日頃 現 象:噴火したもよう	
観測	火山観測報	″時刻不明″	火山現象の時刻が断定できない.	火 山:北海道駒ケ岳 日 時:不明 現 象:噴火したもよう	現象が日界をまたいでいる可能性がある場合を想定. 属性xsi:nilで記述可能.

意味の「曖昧さ」に応じた処理をしてもらうことを想定した DateTimePrecision 要素を導入する案があった.

```
<DateTime significant="yyyy-mm-dd">
  2008-09-06T00:00:00+09:00</DateTime>
<Duration>P7D</Duration>
<DateTimePrecision>P3D</DateTimePrecision>
```

提案された DateTimePrecision 要素は，その後の議論で DateTime 要素の precision 属性の形で基本要素の基本形として取り入れられている.

日付時刻の曖昧さを表現する方法については，基本要素の基本形という意味では上記のように技術的に整理されつつあったが，最後に，ヘッダ部での日付時刻表記の型と内容部の日付時刻表記の型の共通化について議論があった．気象庁 XML は，気象・地震・津波・火山等現象ごとの防災情報についてフォーマットを統一して提供することで，防災情報の利用者がそれらを有効に処理し，活用できる環境を整えることを目指して策定されたフォーマットである．このため，ヘッダ部については，dateTime 型と duration 型の基本的な型を用いることにより利用者が属性値による特殊な処理をする必要のないようにしたいという意見がある一方,内容部の要素における「曖昧な日付時刻」も含め柔軟な表記を認めた上でヘッダ部も同じ形式として日付時刻の表記については統一して共通に扱うべきとの議論があった．結果として，ヘッダ部は体系的な整理を優先して属性値は利用しないこととし，内容部は属性値を用いて簡潔な表記を行うこととした．内容部の日付時刻表記については，全ての情報で共通に利用する型として基本要素辞書に DateTime 型を定義し，情報に応じて「種別」を表す type 属性,「有効部分」を表す significant 属性,「幅（時間）」を表す precision 属性,「あいまいさ」を表す dubious 属性を共通化した上で必要な属性を用いることとなった．これら表記の関係について第 13 図にまとめる.

(2) InfoType 要素（仕様 1.6.2 項，運用指針 2.1.3.4 項）

情報の運用形態については,運用指針2.1.3.4項,及び同別紙 2 に詳しく記述している.

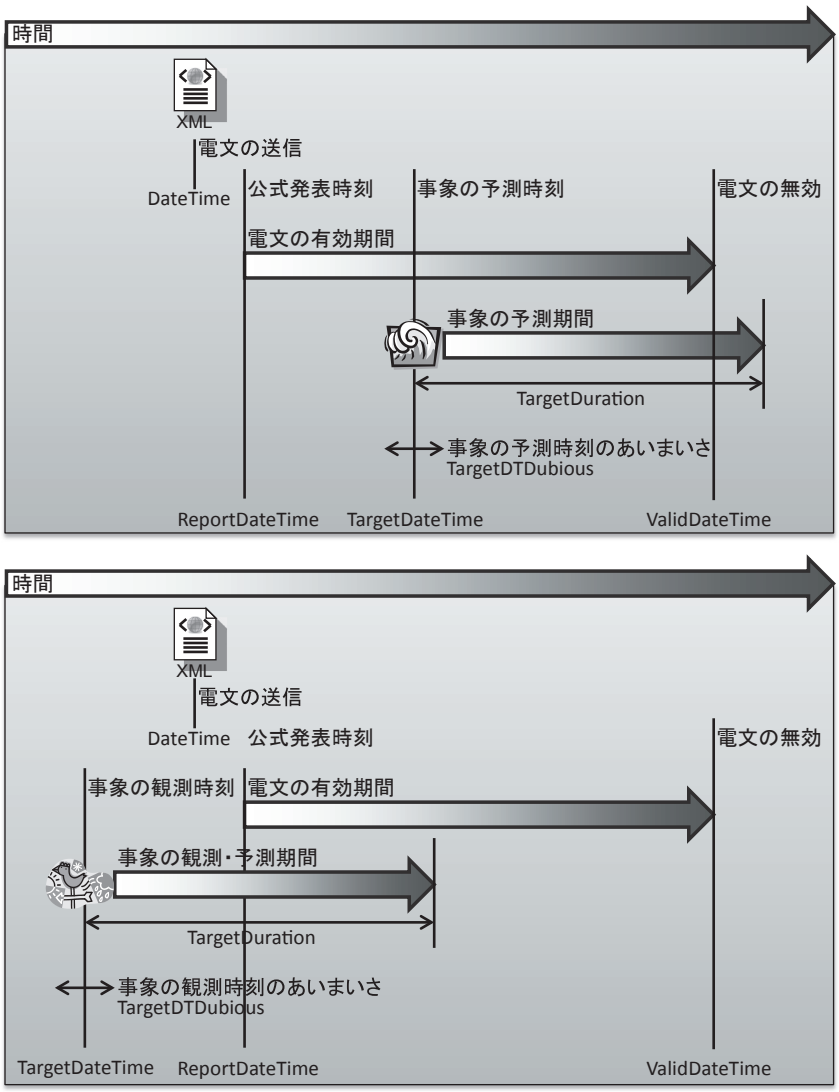
InfoType 要素の運用は，これまで旧来の気象庁の情報提供手段では，どの電文を対象にしているのか，どこを修正しているのか等が体系的に取りまとめられておらず，結局は人間可読的な補足メッセージにて詳細を伝えるような状況であった．気象庁 XML では，これまでシステム利用し辛かった情報の訂正や取消といった運用も整理した上で XML 構造化する改善ができた．運用指針では各電文の複合一意キー制約となる要素の確認や運用定義などを整理しまとめた.

(3) EventID 要素（仕様 1.7 項，運用指針 2.1.3.5 項）

情報の識別情報については，仕様 1.7 項及び運用指針 2.1.3.5 項にて解説している.

特殊気象報や生物季節観測報告気象報において，情報のカテゴリーは，検討当時に国内気象通報式によって規定されていた電文形式のカテゴリーを踏襲し，管理部の Title が同一であっても，異なる観測項目を送信することにしている．このため，EventID 要素に発表日時分と観測項目を付加する対応をとり，どの観測項目を発表しているかを分離できるようにした.

また，特殊気象報（風），特殊気象報（気圧）の場合，XML 電文を発信する観測担当官署が，自官署と担当区域内の特別地域気象観測所の観測データの両方を所掌している関係上，EventID 要素に，更に「現象を観測した気象官署の国際地点番号」を付加することになった．このような対応は土砂災害警戒情報においても必要であり，例えば管理部の EditorialOffice 要素が「札幌管区気象台」となる土砂災害警戒情報は「石狩・空知地方土砂災害警戒情報」と「後志地方土砂災害警戒情報」の 2 種類あるというように，管理部の Title 要素，管理部の Status 要素及び管理部の



第 13 図　時刻・時間要素関係図
　上段が予測に，下段が観測に対する例.
　DateTime は電文の実送信時刻を示す管理部の要素．ReportDateTime, TargetDateTime，TargetDuration，ValidDateTime，TargetDTDubious はヘッダ部の要素.

EditorialOffice 要素の組み合わせで独立した情報単位が決まらないため，ヘッダ部の EventID 要素にヘッダ部の Title 要素と同じ「石狩・空知地方土砂災害警戒情報」や「後志地方土砂災害警戒情報」という文字列を入れることで対応している．

一方，火山関連情報における独立した情報の運用として，EventID 要素の設定には課題があった．情報の利用者にとっては，独立した情報として「ある火山の一連の噴火に関連する情報」を示すことが有用と考えられる．しかし，「一連の噴火」については必ずしも明確に区別できるものではなく，噴火しない場合や終了しないまま次のステージを迎える場合，複数年にまたがる場合もある．このことから，「一連の噴火」による区別は行わず，独立した情報としては，「ある火山の噴火に関連する情報」を単位とすることとし，EventID 要素に「火山コード」を実装して識別することとした．また，噴火に関する火山観測報では，現在も個々の噴火の発生が観測された場合の情報として発表していることから，火山コードに加えて，この個々の「噴火」の識別が必要である．本来は ID として「噴火時刻」を付加するところであるが，その時刻自身の訂正もありうることから，観測後に最初に発表した電文の発信時刻を EventID 要素の値として付加することとした．なお，桜島等では，ある爆発的な噴火について第 1 報，第 2 報で発表したり，噴火開始から継続する「一連の噴火活動」として発表している火山観測報の情報もあるが，一連の事象としてシステム上で識別できるだけのアルゴリズムが構築できていないことから，現時点ではこれまでと同様に訂正報，取消報を除いて電文ごとに EventID 要素値を割り振ることとした．

(4) InfoKind 要素，InfoKindVersion 要素（仕様 2.3 項，運用指針 1 章）

InfoKind 要素は管理部 Title 要素と異なり，フォーマット形式の運用単位での整理を行う情報である．例えば，管理部 Title 要素で「府県気象情報」と「地方潮位情報」とは，地理的・気象学的な概念が違うのみでフォーマット形式は全く同じである．この同じというのは，XML スキーマとして同じというだけではなく，各要素の出現回数やとりうる値が両電文間で同じ事，つまりフォーマット形式の運用が同じ方針であるということを示している．このように情報の形質が近いものについてはフォーマット形式の運用単位が同じとなることから，そのようなものを取りまとめるため，InfoKind 要素にてその運用型を示している．これは電文処理系として，InfoKind 要素値が同じものは同じ処理系で解析・加工処理が可能であり，出力時の分類において管理部 Title 要素を活用することを想定している．

InfoKindVersion 要素の値について，数字がマイナーバージョンアップによる増加となっている限り，前方互換する．「気象庁防災情報 XML フォーマットバージョン管理表（フォーマット別）」によって，InfoKind 要素別の最新バージョンと更新履歴が判別できるが，各電文単位で利用する場合は最新バージョンでなくても，「気象庁防災情報 XML フォーマットバージョン管理表(資料別)」の情報名称別で最新の InfoKindVersion 要素の値よりも上位のバージョン番号に対応しているソフトウェアであれば，当該電文の処理が可能となる．

バージョンアップに関しては第 10 章にまとめる．

(5) Information 要素

Information 要素は防災気象情報の個別事項を大きな粒度でひとくくりにする要素である．

Information 要素の type 属性は防災気象情報事項の種別を示す．Information 要素同士は情報利用としては排他的観点に立つことが多い．例えば気象警報・注意報の情報を利用する場合，利用者は大きな細分区単位としてみるのか，市町村単位としてみるのかのいずれかでのみ利用し，同時にその情報を利用したり重ね合わせたりする人は少ない．つまり，type 属性値が“気象警報･注意報（一次細分区域等)”として検索・取得するか，“気象警報・注意報（市町村等)”として検索・取得するかは利用時・検索時の段階で決まっており，両方を利用する人は少ないということである．逆にいえば，XML 構造の設計として Information 要素で情報を分離することを検討する場合，その際の

粒度は，使用時にこのような排他的に利用することを想定する程度の粒度と考えればよい．

なお，設計当初，警報事項を示すために本要素名は“Warning”としていたが，防災気象情報として汎用的に利用することを考えて，現在の名称へと変更した．

(6) Item 要素

Item 要素は防災気象情報要素である Kind 要素と，その対象地域・地点全体要素である Areas 要素をまとめた親要素である．ここで重要なのは Kind 要素と Areas 要素の関係が兄弟要素であり，親子関係を築いていないことである．これは，XML の構造上，防災気象情報要素の下位概念として対象地域・地点全体要素があるわけでも，またその逆でもないことから，この 2 つを親子関係とするのではなく，兄弟関係とした方が適切であるとの考えからである．これにより，防災気象情報要素と対象地域・地点全体要素との間で，1:1，1:N，N:1，N:N の複数の関係を築くことができることから，各情報の特性に合わせてこれらの運用パターンを定めていく．なお，一般的には利用が煩雑になるので，N:N は利用していない．

(7) Area 要素

Area 要素は対象地域や地点を示す要素である．子要素として対象地域・地点名称を示す Name 要素，同コード値を示す Code 要素，地理空間情報（第 8.3 節（1）項参照）で表現される jmx_eb:Circle 要素，jmx_eb:Coordinate 要素，jmx_eb:Line 要素及び jmx_eb:Polygon 要素がある．XML では数字の取得と文字の取得の間でプログラム的な処理量が変わらないことから，人間可読的な Name 要素を長期継続的に利用する方の要素として推奨することとし，そのために気象庁 XML においては当該 Name 要素の出現回数を必須 1 回とし，地理空間情報で対象地域や地点を表現する場合はコードを示すことができないため Code 要素については省略可とした．

8.3 共通要素（基本要素辞書）

(1) 地理空間情報の拡張

地理空間情報の表記について，気象庁 XML 仕様ドラフト版の段階では，地震分野辞書の Coordinate 要素に，その値は国際標準規格である「ISO6709」を用いることとなっていた．

一方，全般海上警報の電文を検討する過程で，低気圧の位置など地理空間情報を利用する事項が出てきた．ドラフト版で地理空間情報を利用するものは，地震の震源など「点」を表すものだけであったが，全般海上警報では前線のように複数の点を結んでできる「曲線や直線」，海上濃霧警報のように座標で指定した「閉曲線，多角形，領域」といった，点ではない地理空間情報を利用する事項があり，拡張が必要となった．

ISO6709 には複数の地点の表記方法も規定されていたので，「点」を表す Coordinate 要素に加え，スキーマとしては同じ型ではあるが，「曲線」「直線」を表す Line 要素，「閉曲線」「多角形」「領域」を表す Polygon 要素を加えた．また，全般海上警報における警報の対象区域は重要な警報事項であるため，ヘッダ部で対象領域を示す Area 要素において記述する必要が出てきた．このため，ドラフト版では地震分野辞書で定義されていた Coordinate 型を気象関連の電文でも利用できるよう，共通辞書へと移行した．その後，全般海上警報の対象領域に対応するため，Area 要素の Code 要素の出現回数は「0 回か 1 回出現」に変更するとともに，Coordinate 要素，Line 要素，Polygon 要素や予報円を示す Circle 要素もとれるように拡張した．

8.4 内容部の個別要素（気象分野個別辞書）

(1) 気象分野個別辞書の構造

気象分野辞書では，Body 要素の子要素として TargetArea 要素，Notice 要素，Warning 要素，MeteorologicalInfos 要素，Comment 要素，OfficeInfo 要素，AdditionalInfo 要素をとることができる．この中で，TargetArea 要素は対象地域，Notice 要素はお知らせの文章，Comment 要素は文章，OfficeInfo 要素は担当部署に関する事項，AdditionalInfo 要素は電文の個別付加事項という

ように個別の事項について利用する要素である．

一方，Warning 要素，MeteorologicalInfos 要素はいずれも防災気象情報の個別事項を大きな粒度でひとくくりにする要素である．警報事項を示す Warning 要素の構造はヘッダ部の Information 要素と同じように Item 要素を子要素としてとり，type 属性による排他的な利用についても同様の考え方である．

MeteorologicalInfos 要素は予報や観測等の警報事項以外の防災気象情報全般について利用する大きな粒度でひとくくりにする要素で，子要素としてある時刻の予報事項や観測事項を示す MeteorologicalInfo 要素と，予報事項や観測事項を時系列で示す TimeSeriesInfo 要素をとる．MeteorologicalInfo 要素の構造は，ほぼ Warning 要素と同じであるが，予報や観測の時刻を特定する子要素が追加されている．MeteorologicalInfos 要素の type 属性は予報・観測の種別を示し，ヘッダ部の Information 要素や Warning 要素のように領域の粒度といった排他的な観点のみで分けられているとはいえないため，必要な気象情報に応じて検索することとなる．TimeSeriesInfo 要素については次項で解説する．

(2) 時系列表現

気象分野の防災気象情報では気象要素の観測や予想を時系列で提供しているものがある．例えば，府県天気予報は，次のカテゴリー予報のように「今夜」「明日」「明後日」といった時間順に同じ種類の予報を並べて時系列で記述している（第 14 図）．

今日　北の風 雨 夜 くもり
明日　北の風 後 やや強く 晴れ 昼前 から くもり
明後日 北の風 後 東の風 くもり 時々 晴れ

府県天気予報の気象庁 XML 形式は，既に部外提供を行っていた旧形式（府県天気予報の独自形式）の XML 電文（第 15 図）を参考にした．旧形式の府県天気予報は，天気，風，波のカテゴリー予報の部分は「今日」「明日」「明後日」という順序で，降水確率等は「今夜の 18 時から 24 時」「明日の 0 時から 6 時」「明日の 6 時から 12 時」

```
ﾖﾎｳ 1 ｷｼｮｳ
東京都　１４日１１時

東京地方
今日　　北の風　雨　夜　くもり                          （３１３）
明日　　北の風　後　やや強く　晴れ　昼前　から　くもり（１１１）
明後日　北の風　後　東の風　くもり　時々　晴れ          （２０１）
海
今日　　波　０．５メートル
明日　　波　０．５メートル　後　１メートル
明後日　波　０．５メートル

伊豆諸島北部
今日　　北東の風　雨　夜　くもり                        （３１３）
明日　　北東の風　後　やや強く　くもり　夕方　から
　　夜のはじめ頃　雨                                    （２１２）
明後日　北東の風　やや強く　くもり                      （２００）
海
今日　　波　２メートル　うねり　を伴う
明日　　波　１．５メートル　後　２メートル　うねり　を伴う
明後日　波　２．５メートル　後　２メートル　うねり　を伴う

伊豆諸島南部
今日　　北東の風　後　北西の風　雨　時々　くもり       （３０２）
明日　　北西の風　三宅島　では　後　北東の風　やや強く
　　くもり　夕方　から　雨                              （２１４）
明後日　北東の風　やや強く　後　北西の風　くもり　一時
　　雨                                                  （２０２）
海
今日　　波　３メートル　後　２．５メートル　うねり　を伴う　ただし
　　三宅島　では　２．５メートル　後　２メートル　うねり　を伴う
明日　　波　２メートル　うねり　を伴う　ただし　三宅島　では
　　１．５メートル　後　２メートル　うねり　を伴う
明後日　波　２．５メートル　後　２メートル　うねり　を伴う

小笠原諸島
今日　　南の風　晴れ　時々　くもり                      （１０１）
明日　　南西の風　晴れ　昼過ぎ　から　くもり           （１１１）
明後日　南西の風　やや強く　後　北西の風　くもり　時々
　　晴れ                                                （２０１）
海
今日　　波　１．５メートル
明日　　波　１．５メートル　後　２メートル　うねり　を伴う
明後日　波　２メートル　うねり　を伴う

気温　今日日中の最高　　６度　（東京　　７度）
　　８度　（大島　　８度）
　　１５度　（八丈島　　１７度）
　　２２度　（父島　　２２度）
　　明日朝の最低　　５度　（東京）
　　８度　（大島）
　　１１度　（八丈島）
　　１７度　（父島）
　　明日日中の最高　　１３度　（東京）
　　１２度　（大島）
　　１４度　（八丈島）
　　２３度　（父島）

降水確率　（１２－１８）　　００６０　（東京地方）
　　００６０　（伊豆諸島北部）
　　００６０　（伊豆諸島南部）
　　００１０　（小笠原諸島）
　　（１８－００）　　００４０　（東京地方）
　　００４０　（伊豆諸島北部）
　　００６０　（伊豆諸島南部）
　　００１０　（小笠原諸島）
　　（００－０６）　　００００　（東京地方）
　　００２０　（伊豆諸島北部）
　　００１０　（伊豆諸島南部）
　　００１０　（小笠原諸島）
　　（０６－１２）　　００００　（東京地方）
　　００３０　（伊豆諸島北部）
　　００１０　（伊豆諸島南部）
　　００２０　（小笠原諸島）
```

第 14 図　府県天気予報かな漢字電文例

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<report xmlns="http://adess.kishou.go.jp/xml10" lang="ja">
<head>
 <title>府県天気予報(11 時形式)</title>
 <dateTime>2012-02-14T02:00:00Z</dateTime>
 <type>発表</type>
 <editorialOffice>気象庁予報部</editorialOffice>
 <publishingOffice>気象庁</publishingOffice>
 <additionalInfo>
  <v k="発表和暦日時">平成２４年２月１４日１１時００分</v>
 </additionalInfo>
</head>
<body>
 <feature name="カテゴリー予報" isTimeSeries="true"
         isSpaceSeries="false">
  <propertySuffix>
   <property name="波浪予報文">
    <v k="波高 1">m</v>
    <v k="波高 2">m</v>
    <v k="特定地域波高 1">m</v>
    <v k="特定地域波高 2">m</v>
   </property>
  </propertySuffix>
  <dateTime value="2012-02-14T02:00:00Z">
   <time>
    <t>PT0H PT13H</t>
    <t>PT13H PT37H</t>
    <t>PT37H PT61H</t>
   </time>
   <location name="東京都/東京地方">
    <property name="天気テロップ">
     <t>313</t>
     <t>111</t>
     <t>201</t>
    </property>
    <property name="天気予報文">
     <t>
      <v k="天気 1">雨</v>
      <v k="時要素 1">夜</v>
      <v k="天気 2">くもり</v>
      <v k="時要素 2" />
      <v k="天気 3" />
      <v k="特定地域名" />
      <v k="特定地域時要素" />
      <v k="特定地域天気" />
     </t>
     <t>
      <v k="天気 1">晴れ</v>
      <v k="時要素 1">昼前 から</v>
      <v k="天気 2">くもり</v>
      <v k="時要素 2" />
      <v k="天気 3" />
      <v k="特定地域名" />
      <v k="特定地域時要素" />
      <v k="特定地域天気" />
     </t>
     <t>
      <v k="天気 1">くもり</v>
      <v k="時要素 1">時々</v>
      <v k="天気 2">晴れ</v>
      <v k="時要素 2" />
      <v k="天気 3" />
      <v k="特定地域名" />
      <v k="特定地域時要素" />
      <v k="特定地域天気" />
     </t>
    </property>
    <property name="風予報文">
     <t>
      <v k="風向 1">北</v>
      <v k="風強さ 1" />
      <v k="時要素" />
      <v k="風向 2" />
      <v k="風強さ 2" />
      <v k="特定地域名" />
      <v k="特定地域風向 1" />
      <v k="特定地域風強さ 1" />
      <v k="特定地域時要素" />
      <v k="特定地域風向 2" />
      <v k="特定地域風強さ 2" />
     </t>
```

第 15 図　旧形式府県天気予報 XML の例

「明日の 12 時から 18 時」「明日の 18 時から 24 時」の順序でという具合に，t 要素の順序を用いて時系列的表現を記述していた．

府県天気予報の気象庁 XML 形式では，旧形式の時系列表現を準用し，更に要素と日時の対応を明確化するために TimeDefine 要素と timeId 属性による時系列表現を検討した．TimeDefine 要素は対象時間を定義する要素で，天気などの気象要素について時間軸に沿って列挙されている値について，timeId 属性を介して対象時間と結びつける方法である（第 16 図）．

気象要素の値を格納する部分について，その後の議論では，時間の参照だけでない可能性があるので属性名を“timeId”ではなく“refID”にするなどの変更が加えられ，気象庁 XML の仕様に採用されている．

気象情報は時系列で表現できるものが多いため，この表現方法は他の電文においても有効に活用されている．また，まだ時系列的表現は他の個別辞書にて利用されていないが，新たに利用する場合には，本構造に準ずることが望ましい．

```
<TimeDefine>
    <Time timeId="1">
        <TargetDateTime>2008-06-25T08:00:00+09:00</TargetDateTime>
        <TargetDuration>PT7H</TargetDuration>
        <Name>今夜</Name>
    </Time>
    <Time timeId="2">
        <TargetDateTime>2008-06-25T15:00:00+09:00</TargetDateTime>
        <TargetDuration>PT24H</TargetDuration>
        <Name>明日</Name>
    </Time>
    <Time timeId="3">
        <TargetDateTime>2008-06-26T15:00:00+09:00</TargetDateTime>
        <TargetDuration>PT24H</TargetDuration>
        <Name>明後日</Name>
    </Time>
</TimeDefine>
<Item>
    <Area>
        <Name>東京地方</Name>
        <Code>4410</Code>
    </Area>
    <WeatherSentence>
        <Value timeId="1">くもり 所により 夜遅く 雨</Value>
        <Value timeId="2">雨 朝晩 くもり</Value>
        <Value timeId="3">くもり 一時 雨</Value>
    </WeatherSentence>
</Item>
```

第 16 図　旧形式府県天気予報 XML を参考にした気象庁 XML の時系列表現案

(3) 気象予報情報の時間的変化の表現

気象現象には時間とともに状態が変化するものがある．例えば，府県天気予報では，「くもり夜遅く雨」や「北東の風 後 北の風」のように気象要素の時間的変化について表現する必要がある．「地方天気分布予報，府県天気予報及び地域時系列予報の実施細目について（通知）」（平成22.5.27 気業第 64 号．以下本項において「通知」という．）に，例えば，「天気」について次のように定められている．

・府県天気予報の表現は，「基本天気＋地域天気」として表す．
・基本天気は，予報区内で卓越天気（ある時間帯で予報区内の多数を占める天気）の経過を簡潔に表す．基本天気は「晴れ」，「くもり」，「雨」，「雪」，「雨か雪」，「雪か雨」の天気予報用語と現象が発生する期間を示す「一時」，「時々」，「後」や時間細分の用語等とを組み合わせて表す．
・地域天気は，「地域名，時間指定，地域天気の種類」の順で表現する．

このような気象状況が変化する状況の表記についても，気象分野辞書を用いる電文で共通化する必要が出てきた．

「変化前」，「変化後」の記載がある要素の記述方法について第 8 表の案を示し検討を行った．案 1, 案 2, 案 3, 案 5 については，「変化前」，「変化後」などの変化傾向を属性や要素のとりうる値として示す方法をとった．一方，案 4 ではより通知の定めていることに従うよう，変化前若しくは卓越した現象を示すものとして「Primary」を気象要素名の前に付加した要素を，変化後を示すものとして「Change」を気象要素名の前に付加した要素を用いる方法を検討した．具体的には，「くもり夜遅く雨」の「くもり」のように，天気における基本天気の最初の天気予報用語は PrimaryWeather

第 8 表 「変化前」「変化後」の記載がある要素の記述方法

案1）	<Content> <WindDirection> <Type>風向(変化前)</Type> <Value unit="8方位漢字">北東</Value> </WindDirection> <WindDirection> <Type>風向(変化後)</Type> <Value unit="8方位漢字">南</Value> </WindDirection> </Content>
案2）	<Content type="風向(変化前)"> <WindDirection unit="8方位漢字">北東</WindDirection> </Content> <Content type="風向(変化後)"> <WindDirection unit="8方位漢字">南</WindDirection> </Content>
案3）	<Content> <Type>風向(変化前)</Type> <WindDirection unit="8方位漢字">北東</WindDirection> </Content> <Content> <Type>風向(変化後)</Type> <WindDirection unit="8方位漢字">南</WindDirection> </Content>
案4）	<Content> <PrimaryWindDirection> <KanjiBase8WindDirection>北東</KanjiBase8WindDirection> </PrimaryWindDirection> <ChangeWindDirection> <KanjiBase8WindDirection>南</KanjiBase8WindDirection> </ChangeWindDirection> </Content>
案5）	<WindDirection target="風向(変化前)" unit="8方位漢字">北東</WindDirection> <WindDirection target="風向(変化後)" unit="8方位漢字">南</WindDirection>

要素で，「くもり夜遅く雨」の「夜遅く雨」のように，現象が発生する期間を示す「夜遅く」と組み合わされて使われる天気予報用語の「雨」はChangeWeather要素で表現するというものである．「くもり夜遅く雨」の案4での表記を次に示す．

```
<Content>
 <PrimaryWeather>
  <BaseWeather> くもり </BaseWeather>
 </PrimaryWeather>
 <ChangeWeather>
  <BaseWeather> 夜遅く 雨 </BaseWeather>
 </ChangeWeather>
</Content>
```

XML コンソーシアムとの議論を受けて，基本要素の構造は案5を基本として採用し，「変化前」，「変化後」の記載がある要素の記述方法は基本要素の要素名や属性に「変化前」や「変化後」の記述をするのではなく，案4の構造の考え方をするという方向性が示されたことから，再度次のように気象要素の記述方法を検討した

- 変化前，変化後を表す項目はそれぞれPrimary 要素，Change 要素でくくる．
- 地域を分けて記載する場合はLocal 要素でくくる．
- Primary 要素，Change 要素，Local 要素を用いる項目は“Complex[要素名]”要素で，用いない項目は“Simple[要素名]”要素でくくる．

この案で「くもり夜遅く雨」を表現すると次のようになる．

```
<ComplexWeather>
 <Sentence> くもり 夜遅く 雨 </Sentence>
 <Primary>
  <Weather> くもり </Weather>
 </Primary>
 <Change>
  <TimeModifier> 夜遅く </TimeModifier>
  <Weather> 雨 </Weather>
 </Change>
</ComplexWeather>
```

この案に対して，Complex や Simple を冠した要素名は気象要素ではなく技術的なもので要素名の原則から外れることとなるため導入はやめるべき，要素名は気象要素を示したものにすべき，Primary 要素や Change 要素については，航空気象通報式の変化傾向を示す用語である BECMG（Becoming）や TEMPO（Temporary）を利用してはどうかとの提案があり，結果として，卓越や変化前には Base 要素，変化後は Becoming 要素，一時的な現象には Temporary 要素を利用するといった次の修正を行った．

- 同一の要素は ElementAPart 要素でくくる．
- 変化状況は，卓越若しくは変化前を Base 要素，断続現象を Temporary 要素，変化後を Becoming 要素でくくって表現する．
- 区域の一部の状況を表現する場合は，Base 要素，Temporary 要素，Becoming 要素の中では Local 要素でくくる．
- 一部領域で領域全体（Item 要素下の Area 要素で示される領域）と同じ予報表現をする必要がある場合は ElementAPart 要素の子要素として SubArea 要素でくくる．

この修正を反映した「くもり夜遅く雨」の記述は次のようになる．

```
<WeatherPart>
 <Sentence> くもり 夜遅く 雨 </Sentence>
 <Base>
  <Weather> くもり </Weather>
 </Base>
 <Becoming>
  <TimeModifier> 夜遅く </TimeModifier>
  <Weather> 雨 </Weather>
 </Becoming>
</WeatherPart>
```

（4）辞書策定における全体調整

当初，気象分野個別辞書の作成は予報課所掌の電文における共通化を中心に行っていたが，その後，予報課以外の所掌の電文との調整を行った．生物季節観測電文との調整では，平年差，昨年差について，IndividualAddition 要素（後に

AdditionalInfo に名称を変える）の子要素に生物季節観測の独自要素を追加する方法とした．また，Item 要素の対象地域は Area 要素で定義されていたが，気象分野個別辞書に定義されていた Area 要素は，面的な領域の表記には適していたが観測所のような様々な属性情報のある地点の表記には適していなかったため，Area 要素のかわりに観測所の情報を記述する Station 要素を Item 要素の子要素に追加するなどの調整を行った．紫外線情報電文では，この時点で環境分野個別辞書として検討されていたが，気象分野個別辞書との共通化も視野に入れるため，同じ型でありながら Observation 要素と Forecast 要素に分けて観測と予報を区別していたものを，同じ気象情報という意味の MeteorologicalInfos 要素に統一するなど，比較的大きな辞書の変更を行った．

その後，季節予報電文と気象分野個別要素の調整を行った．季節予報の例として，1 か月予報におけるかな漢字電文では，特に注意を要する事項，気象要素の予測，次回の発表予定等の順で記述されており，当初の XML 電文でもかな漢字電文とほぼ同じ順序で，「特に注意を要する事項」はヘッダ部に，「週ごとの気温経過」を除く予測については内容部の MeteorologicalInfos 要素に，「週ごとの気温経過」は TimeSeriesInfo 要素に，「次回発表予定等」は AdditionalInfo 要素に記述されていた（第 17 図）．一方，同じ時期に府県天気予報と府県週間天気予報の構造を統一できないかを検討しており，TimeSeriesInfo 要素の後に MeteorologicalInfos 要素を記述したいという要望があった．気象庁 XML は要素の出現順序は固定という方針であることから，1 か月予報等季節予報の要素の順序との調整が必要になった．また，電文の内容をいくつかの部分の集合として分けて記述する場合には，MeteorologicalInfo 要素や TimeSeriesInfo 要素を複数使って記述することができるが，MeteorologicalInfo 要素は MeteorologicalInfos 要素の子要素のため問題はなくとも，TimeSeriesInfo 要素は Body 要素の直接の子要素であるため，複数の TimeSeriesInfo 要素を記述すると Body 要素に子要素が増えてしまうことが気がかりとなっていた．そこで，構造の大きな変更となってしまうが，TimeSeriesInfo 要素を MeteorologicalInfos 要素の子要素へと変更することとした．MeteorologicalInfos 要素を複数利用すれば，時系列的な記述とある時刻の気象情報の記述の順序を XML 化対象電文と同様の順序で記述できる．

このようにして，気象分野の電文について，関係部課と協力しながら，全ての気象分野の電文が共通化された構造で記述できるように調整した．

```
<?xml version="1.0" encoding="utf-8" ?>
<Report xmlns="http://xml.kishou.go.jp/jmaxml/"
 xmlns:jmx="http://xml.kishou.go.jp/jmaxml/">
  <Control>
    <Title>地方１か月予報</Title>
    <DateTime>2008-03-07T05:25:00Z</DateTime>
    <Type>通常</Type>
    <EditorialOffice>札幌管区気象台</EditorialOffice>
    <PublishingOffice>札幌管区気象台</PublishingOffice>
                〔〕
  </Control>
  <Head xmlns="http://xml.kishou.go.jp/jmaxml/informationBasis/">
    <Title>北海道地方１か月予報</Title>

<ReportDateTime>2008-03-07T14:30:00+09:00</ReportDateTime>

<TargetDateTime>2008-03-08T00:00:00+09:00</TargetDateTime>
    <TargetDuration>P1M</TargetDuration>
    <ID></ID>
    <InfoStatus>発表</InfoStatus>
    <Serial></Serial>
    <Condition>更新</Condition>
    <InfoKind>季節予報</InfoKind>
    <InfoKindVersion>1.0</InfoKindVersion>
    <Headline>
      <Text>期間を通して気温が高い見込みです。雪の多い傾斜地ではなだれに注意して下さい。</Text>
    </Headline>
  </Head>
  <Body xmlns="http://xml.kishou.go.jp/jmaxml/body/meteorology/"
    xmlns:jmx_eb="http://xml.kishou.go.jp/jmaxml/elementBasis/">
    <TargetArea codeType="季節予報／地域">
      <Name>北海道地方</Name>
      <Code>11</Code>
    </TargetArea>
    <Notice>お知らせ　これは検討中の案です。</Notice>
    <MeteorologicalInfos type="１か月予報">
      <MeteorologicalInfo>
        <DateTime significant="yyyy-mm-dd"
          >2008-03-08T00:00:00+09:00</DateTime>
        <Duration>P1M</Duration>
        <Name>向こう１か月</Name>
        <Item>
          <Kind>
            <Name>季節予報</Name>
            <Property>
              <Type>出現の可能性が最も大きい天候と、特徴のある気温、降水量、日照時間等の確率</Type>
              <ClimateFeaturePart>
                <jmx_eb:ClimateFeature>
                  <jmx_eb:GeneralSituationText>天気は数日の周期で変わるでしょう。平年に比べて日本海側では晴れの日が多く、太平洋側では雪または雨の日が多いでしょう。
</jmx_eb:GeneralSituationText>
                  <jmx_eb:SignificantCliamteElement kind="気温">
                    <jmx_eb:Text>気温は、平年より高い確率が６０％です。</jmx_eb:Text>
                    <jmx_eb:ProbabilityOfAboveNormal unit="%">60</jmx_eb:ProbabilityOfAboveNormal>
                  </jmx_eb:SignificantCliamteElement>
                </jmx_eb:ClimateFeature>
              </ClimateFeaturePart>
            </Property>
          </Kind>
            〔〕
          <Area codeType="季節予報／地域">
            <Name>北海道太平洋側</Name>
            <Code>14</Code>
          </Area>
        </Item>
      </MeteorologicalInfo>
    </MeteorologicalInfos>
    <TimeSeriesInfo type="週毎の気温経過">
      <TimeDefines>
        <TimeDefine timeId="1">
          <DateTime>2008-03-08T00:00:00+09:00</DateTime>
          <Duration>P7D</Duration>
        </TimeDefine>
        <TimeDefine timeId="2">
          <DateTime>2008-03-15T00:00:00+09:00</DateTime>
          <Duration>P7D</Duration>
        </TimeDefine>
        <TimeDefine timeId="3">
          <DateTime>2008-03-22T00:00:00+09:00</DateTime>
          <Duration>P14D</Duration>
        </TimeDefine>
      </TimeDefines>
      <Item>
        <Kind>
          <Name>季節予報</Name>
          <Property>
            <Type>地域・期間平均平年偏差各階級の確率</Type>
            <ProbabilityValuesPart>
              <jmx_eb:ProbabilityValues refID="1"
               kind="気温">
                <jmx_eb:ProbabilityOfBelowNormal unit="%"
                 >10</jmx_eb:ProbabilityOfBelowNormal>
                <jmx_eb:ProbabilityOfNormal     unit="%"
                 >10</jmx_eb:ProbabilityOfNormal>
                <jmx_eb:ProbabilityOfAboveNormal unit="%"
                 >80</jmx_eb:ProbabilityOfAboveNormal>
              </jmx_eb:ProbabilityValues>
              <jmx_eb:ProbabilityValues refID="2" kind="気温">
                <jmx_eb:ProbabilityOfBelowNormal unit="%"
                >20</jmx_eb:ProbabilityOfBelowNormal>
                <jmx_eb:ProbabilityOfNormal     unit="%"
                >30</jmx_eb:ProbabilityOfNormal>
                <jmx_eb:ProbabilityOfAboveNormal unit="%"
                >50</jmx_eb:ProbabilityOfAboveNormal>
              </jmx_eb:ProbabilityValues>
              <jmx_eb:ProbabilityValues refID="3" kind="気温">
                <jmx_eb:ProbabilityOfBelowNormal unit="%"
                >20</jmx_eb:ProbabilityOfBelowNormal>
                <jmx_eb:ProbabilityOfNormal     unit="%"
                >30</jmx_eb:ProbabilityOfNormal>
                <jmx_eb:ProbabilityOfAboveNormal unit="%"
                >50</jmx_eb:ProbabilityOfAboveNormal>
              </jmx_eb:ProbabilityValues>
            </ProbabilityValuesPart>
          </Property>
        </Kind>
        <Area codeType="季節予報／地域">
          <Name>北海道地方</Name>
          <Code>11</Code>
        </Area>
      </Item>
    </TimeSeriesInfo>
    <AdditionalInfo>
      <ClimateForecastAddition>
        <NextForecastSchedule target="１か月予報">
          <Text>毎週金曜日　１４時３０分　次回は３月１４日</Text>
          <DateTime>2008-03-14T14:30:00+09:00</DateTime>
        </NextForecastSchedule>
        <NextForecastSchedule target="３か月予報">
          <Text>３月２５日（火）　１４時</Text>
          <DateTime>2008-03-25T14:00:00+09:00</DateTime>
        </NextForecastSchedule>
        <NoticeOfSchedule/>
        <AdditionalNotice/>
      </ClimateForecastAddition>
    </AdditionalInfo>
  </Body>
</Report>
```

第 17 図　当初作成していた季節予報の気象庁 XML 案

8.5 内容部の個別要素（地震・津波分野個別辞書）

(1) 地震津波情報のコード電文と運用の課題

地震火山分野では，平成 11 年 3 月から開始された量的津波予報導入に伴う津波予報区の細分化，平成 8 年以降からの震度観測点の増加，地方公共団体や関係機関の震度観測点の取り込みなどに伴う情報発表量の大幅な増加に適切に対応するため，いわゆる人間可読式のかな漢字電文だけでなく，機械可読式の電文をコード電文として作成し，平成 7 年 4 月から運用を行ってきた．

コード電文は各情報について定型のフォーマットを作り，各要素については英数字や記号を用いて内容を示すものである．例えば，津波注意報・警報（以下「津波注警報」という．）の本文を示す接頭辞は“T FR”であり，北海道太平洋沿岸中部の津波予報区は“101”のような 3 桁の数字で示される，というものである．

コード電文は XML 電文と同様に，定められた要素を取得する際には便利であり，かつかな漢字電文と比べて伝送量が小さく抑えられるため，規模の大きな地震や津波など大量の情報を一度に処理する上では便利である．しかし，その反面，震源要素の緯度経度の桁数は 0.1 度単位でしか表記できないといったように定型のフォーマットから外れる表現をとることはできない．また，情報内容の変更や追加などが発生する場合には，コード電文のフォーマット形式も変更することになるが，XML 電文とは異なり同情報を利用する全ユーザに対してソフト改修が必要となってしまうという問題点があった．コード電文では，情報内容の変更に際して，全ユーザに大幅な処理改修が発生しないように，予備として定義していたフラグの値に意味を持たせることや，ヘッダを別にした新規電文を準備して追加変更したフォーマットを提供することなどの回避策を用いて運用を行ってきた．

例としては，津波注警報発表の迅速化を平成 18 年 10 月 2 日より開始した際のデータ種類コード追加が挙げられる．この迅速化は，緊急地震速報を活用して地震発生後最速 2 分程度で津波注警報を発表できるようにするというものである．従来の津波注警報電文の第 1 報に用いられるコード電文では震源要素は記載されなかったが，津波注警報発表前にはおおむね震度速報などが発表されるため，地震津波情報の利用者からすると他の情報を用いることで震源に近い領域がおおむね把握できていた．しかし，この津波注警報発表の迅速化では，事前情報が何もない中でいきなり津波注警報が発表されるため，テロップなどの誤発表を行わないよう，震源の位置を津波注警報に加えてほしいという強い要望が報道機関から行われた．しかし，これにこたえて従前からの電文に震源要素を追加すると，津波注警報のコード電文を利用している全ユーザに対して改修作業が必要になってしまうため，従前の電文内容には変更を加えず，震源要素を付加した津波注警報電文（データ種類コード：ツナミヨホウ 6）を新たに作成し，配信することで要望に対応することとした．これにより，津波注警報電文のデータ種類コードは 6 種類となり，このように電文数を増加させると電文作成及び配信にタイムラグが生じるおそれがあることや，またどの機関にどの電文を配信すべきか，という配信管理の負荷が上昇することになった．

(2) XML 形式への検討

地震火山部では，緊急地震速報の技術開発及び実用化検討に際して，一部関係機関から XML 形式による情報配信の対応を強く要請されたこともあり，平成 14 年頃から XML 形式の検討を行ってきた．平成 16 年 7 月 8 日から 9 日に掛けて，本庁内の緊急地震速報及び地震津波情報の担当者が半ば自主的に集まり「緊急地震速報の XML 化に関する打ち合わせ」を行い，XML フォーマットの検討を行った．同打ち合わせでは，緊急地震速報の XML フォーマットを検討するのが主目的であったが，それだけでなく地震津波情報や地震火山部の全情報まで拡張しても耐えられるようなフォーマットの策定について，平成 15（2003）年十勝沖地震で発表された地震津波情報を素材として様々な検討がなされた．現在の XML 形式による地震津波情報の内容部の構造は，おおむね同打ち合わせで行われた議論の結果を反映したものである．

(3) XML 形式による地震津波情報の特徴

ア　情報の一連番号による識別管理

かな漢字電文及びコード電文の利用者からは，従来から「発表される情報がどの地震に対応しているものか分からないので対処してほしい」という意見・要望が度々行われていた．有感地震がほとんど発生しないような通常時においては，地震発生の順番に各種地震情報が発表されるため，情報と地震の整合性を判別するのは容易である．しかし，例えば平成 23（2011）年東北地方太平洋沖地震発生後などのように大小様々な規模の余震や他の地域の地震が多発するような状況下では，今受信した地震津波情報はどの地震に対するものなのかを判別することは容易でない．特に，下記のような状況下においては，情報と対応する地震の判別が困難になると考えられる．

- 余震の地震情報を発表した後に規模の大きな地震（本震）の地震情報の更新を行うような，発表する地震情報の震源要素が時間順に並ばない場合．
- 震度 3 程度の地震が発生してその 2,3 分後に例えば震度 5 弱以上の規模の大きな地震が発生するような場合．この時，自動処理で生成される震度速報は両方の地震で発表されるが，その後の地震情報は情報伝達の混乱を避ける目的で，規模の大きな地震だけしか発表されないことがある．
- 津波注警報，津波情報が発表される場合．地震情報の発表はおおむね地震発生後 30 分程度以内で一旦終了するが，津波注警報，情報の発表は短くても 30 分から 1 時間程度は掛かるため，いつまでも過去の地震情報の震源要素が情報に入ってくる．また，規模の大きな余震や別領域で発生する規模の大きな地震の発生により，津波注警報がグレードアップし，また該当予報区が拡大することも考えられる．このような場合は津波を発生させる要因となる地震が複数個存在することになる．
- 地震津波情報では即時的に決定した震源要素を速報値として用いているが，規模の大きな地震などの場合は地震発生後数時間程度で震源要素を精査して暫定値に修正することがある．津波注警報や津波情報は震源要素が途中から変更されることになり，それまで発表された地震情報との紐付けができなくなる．

かな漢字電文及びコード電文では地震を一意に識別する情報が盛り込まれていないために，利用者が地震を識別する場合には地震発現時（地震が検知された時刻），震源位置やマグニチュードなどで識別することができると個別に説明を行っていた．しかし，上記の識別処理を行ったとしても，地震発現時は 1 分単位でしか表記されないため，似たような場所で 1 分以内に 2 つ以上有感地震が発生すると地震の識別ができなくなるという点で問題がある．また，上記の識別は利用者ソフトウェアによるものとなるため，利用者が個々に識別を行うよりも情報発信者である気象庁が識別を行った上で情報発信を行うことが望ましい．以上のような問題があったため，最近作成された新しい情報である緊急地震速報では，地震発現時刻（秒単位）から生成される地震 ID と呼ばれるデータを付して地震イベントの識別に使用していた．気象庁 XML においても，EventID 要素にこの地震 ID を付して全地震津波情報に適用し，上記の課題解決を図った．

しかし，自動処理で生成される緊急地震速報及び震度速報では，例えば本来は複数個に分離されるべき地震を一つに統合することや，若しくはその逆に一つの地震を複数個に分離して処理することもある．人手を介して作成される地震津波情報の EventID 要素は一致するように処理を行っているが，システムで自動的に作成され，発表までの時間の猶予が短い緊急地震速報や震度速報と地震 ID が一致しないことがありうる．また，地震津波情報の処理は東京，大阪の 2 中枢で独立並行して処理を行っている．この 2 中枢では同じ地震波形を用いてほぼ同じ処理を行っているが，地震波形の伝送やシステムの処理サイクルなどの数十ミリ秒や数百ミリ秒といった僅かな差異が秒単位で生成される地震 ID に影響を及ぼすことがあり，東京と大阪で同じ地震を処理しても地震 ID が異なる場合が少なからず存在する．現状で，地震 ID の一意性について上記のような問題点はあり，システム改修だけでなかなか解決できる問題では

ないものの，より一意性が確保できるようにするために今後も処理を検討する必要があるだろう．

イ　種別

例えば，震源・震度に関する情報では，震源要素（震源の位置，マグニチュードなど），震度（都道府県，細分区域，市町村，データ未入電と予想される市町村），付加文（地震，津波，震度，防災に関する事項など）で構成されている．各情報はこのような様々な要素を集められて作成される．しかし，地震に関する情報の要素として，例えば震源要素，津波，震度などと大きな項目で区分することができる．このように，地震津波情報は独立性のある情報区分（これを種別と呼んでいる）にまとめ，これらを組み合わせることで情報を作成している．内容部直下の要素が主な種別を意味しており，上記で挙げた Tsunami 要素（津波），Intensity 要素（震度），Earthquake 要素（震源要素），Comments 要素（付加文）のほかに，単独で用いられることが多い EarthquakeCount 要素（地震回数），NextAdvisory 要素（次回の情報発表時刻），社会的影響度の大きな Naming 要素（命名）などを配置している．

ウ　構造化

コード電文では，震度の情報部分は細分区域の震度，市町村の震度，観測点の震度をそれぞれ列挙する形式となっている．この形式では，全国の震度分布を知ることはできるが，例えば茨城県南部の各観測点の震度分布やつくば市など特定の市区町村の震度を検索する，といったような，細分区域や市区町村，各観測点の連携を必要とする検索を電文だけで行うことはできない．コード電文で上記の検索を可能にするためには，関係する細分区域，市区町村，観測点の各コードの関連性を全て把握しておく必要がある．

しかし，XML 形式では，ツリー状に表記する構造化テキストを利用することができる．震度データを細分区域，市区町村，観測点とツリー状に構造化したため，各コード表の関連性を知らなくとも，XSLT で簡単な記述を行うだけで上記の検索は実現することができる．

津波情報や震度情報では，多数の予報区や観測点のデータが記載されるため，データ量が多くなりがちである．検索性を高めるなどの利便性を向上させるためにも，津波や震度において構造化を積極的に取り入れている．

また，津波注警報等で表現される予報値と観測値については，例えば予測震度と観測震度や予想される津波の高さと最大波の高さなどといったようにほぼ同じ値や形式をとることから，予報値と観測結果を表現する観測値の構造を同じ表現形式にすることで，比較を容易に行うことができる．

エ　ヘッダ部と内容部の活用について

地震津波情報では，ヘッダ部と内容部で予報区や細分区域を表現する方法が異なる．ヘッダ部では，予報区や細分区域が注警報階級や震度順に記述されており，内容部では，その逆に都道府県や予報区，細分区域などの地域別に記述されている．

気象警報・注意報と異なり，地震津波情報は原則として全国の津波や震度に関わる要素が含まれており，利用者の利用用途も様々であると思われるが，上記のようにヘッダ部と内容部の構造の記述方法が異なるため，用途に応じて使い分けることが可能となっている．例えば中央省庁や民放キー局など全国の状況が必要な利用者にとっては，ヘッダ部を見れば細かな予報区単位ではないもののどの地域にどのような津波注警報が発表されているかを大まかに把握することができる．一方で，各都道府県や市区町村単位の報道機関，防災機関などが自らの当該地域の状況を把握する場合は，内容部の該当する要素配下を取得すれば細かな情報まで含めて必要な要素全てを取得することができる．

このように考えると，ヘッダ部は全国を俯瞰的に把握するために利用し，内容部は都道府県や予報区単位などで細部まで含めた内容を抽出するために用いるのが望ましい活用法であると考えられる．

オ　電文種別の統合

近年のネットワークの帯域の増大，低価格化は目覚ましいものがあるが，コード電文の運用をは

じめた頃は十分な帯域を確保することが困難であり，大量のデータが入った電文を配信すると相応の大きな遅延が発生する危険性があった．一方で地震津波情報についてはその情報の性質から常に緊急性が求められるため，情報内容によっては似通った内容の情報を分割してデータ量を削減するなどの工夫を行っていた．例えば，震源・震度に関する情報と各地の震度に関する情報（前者は震度 3 以上が観測された際に観測された地域を示し，後者は各震度観測点の震度を全て示す）や津波到達予想時刻・予想される津波の高さに関する情報と各地の満潮時刻・津波到達予想時刻に関する情報（前者は津波注警報と各予報区の津波の高さ及び各予報区への到達予想時刻を示し，後者は津波注警報と予報区及び各潮位観測点への到達予想時刻と満潮時刻を示す）がそうである．

気象庁 XML では，データ伝送の環境は十分整っているという前提のもと，これらの重複した情報を統合して一つの情報として整理した．

カ　修正情報

規模の大きな地震が発生すると，震源近傍の自治体などの震度観測点においては回線輻輳等の影響により，観測点から気象庁への震度データの伝送に相当の時間を要することがある．そのため，地震情報を発表した後に，より大きな震度が入電することとなり，震度観測点の更新という形で地震情報を複数回発表することがある．コード電文では，情報として分かりやすさや取得の容易さを別にすれば情報として必要な要素はおおむね網羅されていたが，観測値の追加や修正といった修正情報については表現されていなかった．情報の受け手側からすると，修正情報がないためにどの観測点が追加されたか，上方又は下方に修正されたかを把握するためには，前に発表された情報と比較を行う必要がある．しかし，本項アで示したような問題もあるため，前に発表された情報との比較を情報の受け手側が行うことは大変困難であった．

上記のような事情から，震度観測点の追加などを示す情報付加はかねてより防災機関，報道機関等から強く要望されていた．気象庁 XML では，修正情報の要素を用意することで更新状況が把握できるようにした．これにより，以前に発表された情報を取得できなかったとしても，修正情報を取得することが可能となった．

この修正情報は，震度観測点以外にも都道府県，細分区域，市町村の最大震度や津波の高さについても記載できるようにした．

（4）個別要素の説明

第（3）項イに記載したとおり，内容部の要素は種別ごとに独立している．各種の注警報，情報はこれらの種別を組み合わせることで作成される．

ア　震度（Intensity）

緊急地震速報で震度の予測を行い，その後の震度速報及び地震情報において，観測された震度の値を発表するために主に用いられる．Intensity 要素内では，まず Forecast 要素（予報，警報）と Observation 要素（観測）に区別されている．Forecast 要素と Observation 要素の構造及び用いられるコード表は同一としている．

Forecast 要素は緊急地震速報で使用されることから現在は地方，都道府県，細分区域までの分解能しか持たない．ただし，気象庁 XML の仕様上は市区町村や観測点まで持っているため，将来的に市区町村単位で予測震度を発表するような場合でも，仕様を大きく変更する必要がない．一方，Observation 要素は地震情報で各観測点の震度まで発表するため，都道府県，細分区域，市区町村，観測点というようにより細分化された構造を持っており，これらはツリー構造で表記されている．そのため，通常は下位の領域のデータは上位の領域で括られる要素の中に全て記述されている．

ただし，飛び地などで同一市区町村であっても複数の細分区域にまたがるような場合は例外となる．例えば，長崎県佐世保市は長崎県南西部と長崎県五島列島の 2 つの細分区域にそれぞれの佐世保市の震度が分かれて記述されることになる．

規模の大きな地震が発生して震源近傍の観測点の震度の入電が遅れると，当該市区町村及び観測点は「震度 5 弱以上と推定されるが入電していない」（いわゆる震度 5 弱以上未入電）という扱い

になる．従来のコード電文では，市区町村の観測最大震度が 4 であっても，震度が入電していない震度観測点が「震度 5 弱以上未入電」と判定されると，最大震度「震度 5 弱以上未入電」という値のカテゴリーに当該市区町村が記述される．気象庁 XML でもこの点はコード電文と同じ運用となるが，震度情報は都道府県や細分区域等のツリー構造となることから，当該市区町村の最大震度に「震度 5 弱以上未入電」の値が記述される．

イ　津波（Tsunami）

津波注警報・津波予報で各予報区の区分（カテゴリー）を決定し，各種津波情報で予想される津波の高さ，到達予想時刻，各種潮位観測点や津波計の観測値を示している．

津波は震度と異なり，GPS 波浪計や沖合に整備された海底津波計の観測値を元に該当する沿岸部で予測される津波の高さを情報として発表することがある（例えば，気仙沼広田湾沖などの名称）．このような名称に該当する沿岸部は津波予報区のごく一部にとどまること，若しくは複数の津波予報区にまたがるなど，津波予報区とは直接の関連がないことが多い．そのため，予報区を示す要素を用いた構造化表記とせずに別の要素で明瞭に分離することにした．以上より，津波は Forecast 要素（予報，注警報，予想される津波の高さ，第一波到達予想時刻），Estimation 要素（GPS 波浪計や沖合津波計などによる推定値），Observation 要素（第一波到達時刻，最大波到達時刻，津波の高さなどの津波の各種観測値）の 3 つに区別される．また，震度と同様，Forecast 要素と Observation 要素の構造及び用いられるコード表は同一である．

津波についての細分化単位は全国を 66 に区分した津波予報区のみである．津波予報区は例えば東京湾内湾のように複数の都道府県境にまたがることもあるため，震度のような枝の多いツリー構造ではない．

ウ　震源（Earthquake）

震源位置（緯度，経度，深さ）やマグニチュード，震央地名などを記載している．震源位置やマグニチュードは通常，数値が入るが，場合によっては“不明”や“（震源の深さが）600km より深い”などとなることもある．このような不定形の表現については condition 属性値で表現している．

震源の位置を示す震央地名については，津波注警報や地震情報で用いられる通常の震央地名以外に緊急地震速報（警報）で用いられる短縮表記された震央地名（例えば北海道道東），海外で発生した地震の詳細な震央地名（例えば「詳しい震源の位置はケルマデック諸島です。」）の 3 種類が運用されている．これらの 3 つの震央地名は一つの震源位置に対して複数の震央地名を返すものであるため，統合して一つのコード体系に整理することは困難である．そのため，従来通り 3 つの震央地名に要素をそれぞれあてて使い分けしている．

エ　付加文（Comments）

付加文は発表した注警報，情報に付属して注意喚起や補足を行うために用いられ，地震津波情報では頻度の高い表現を定型付加文と呼び，コード化して運用してきた．気象庁 XML では，これまで用いられてきた付加文を整理し，注警報のための付加文要素，予報のための付加文要素，観測値のための付加文要素，用途限定しない付加文要素，不定形な付加文要素と区分することとした．

なお，かな漢字電文では表示位置などの工夫がなされてきたが，気象庁 XML は従来のコード電文同様に視覚的な表示情報は含んでいない．そのため，望ましい利用方法として表示例を示すことが必要と思われる．

オ　東海地震情報（Tokai）

東海地震に関連する情報に特化するための要素を設けている．

カ　地震回数（EarthquakeCount）

地震が多発するなどして，地震情報の発表が追いつかないような場合，毎時間若しくは数時間おきに有感地震や無感地震の回数を情報として発表することがある．EarthquakeCount 要素で時間ごとの地震回数を記載できるようにする．

キ　余震確率（Aftershock）

規模の大きな地震が発生した後に余震が多発して，その発生様式がいわゆる「本震－余震型」で推移するような場合には余震確率を発表することがある．余震確率の確率表現や期間などを定型として要素化し，情報文本文や解説，補足などをテキスト文で記述できるようにしている．

ク　その他の種別

重要度が高く，データの抜き出しができた方が良いと思われる要素について，種別を作成している．「平成 23（2011）年東北地方太平洋沖地震」などのような命名（Naming 要素）や次回の情報発表予定時刻（NextAdvisory 要素）などがある．

8．6　内容部の個別要素（火山分野個別辞書）

(1) 従来の火山関連電文の問題点とその対応方針

平成 19 年 12 月から噴火警報・予報及び噴火警戒レベルの運用が始まったが，これらの旧形式電文における対応については，準備期間が短かったこともあり，噴火警戒レベルについてコード部に組み込んだ点を除き，従来の火山情報の形式を踏襲することとなった．このため，警戒事項や予報区としての対象市町村は本文中に記述することとなり，これら警戒事項や対象市町村をシステムで自動化処理として読み取らせるには本文中の文字列解析を行う必要があった．また，噴火に関する火山観測報や火山現象に関する海上警報・予報の電文についてもコード部がなかったことから，自動化処理には同様の対応を行う必要があった．このことから，火山関連電文の気象庁 XML 対応にあたっては，これらの問題点を踏まえた上での仕様策定を目指した．

噴火警報・予報と火山の状況に関する解説情報については，従来の電文においてコード部を付けて発表していたが，気象庁 XML としての検討にあたっては，このコード部の内容を整理しつつ，前述のような問題点を対処できるようにした．また，火山現象に関する海上警報については，船舶向けの送信という制限からその電文の大きさが限定される特殊な性質を有することに留意が必要であるが，伝達するべき内容は警戒事項等が中心であることから，発表する要素としては噴火警報・予報と共通化を目指した．

噴火に関する火山観測報については，観測した内容を端的に伝えるという他の火山関連電文とはやや異なる性質を有する．ただ，旧形式電文においても，本文中ではあるものの，噴煙高度等主な観測項目について項目別に記述していたことから，これらの項目を主な要素としつつ，他の気象観測報と整合をとりながら，将来的に追加される可能性のある項目も考慮して，その必要とする要素を検討した．

なお，火山関連電文において，主たる防災気象要素については，ヘッダ部と内容部の両方に記述することとした．

(2) 火山関連電文における個々の要素について

火山関連電文における管理部やヘッダ部の特徴については，第 8.2 節においてまとめている．火山分野個別辞書における個々の要素の特徴については，第 9 表にまとめる．

第9表　火山分野個別辞書の個々の要素の特徴

要素名	特徴・解説
Condition（状況）	・防災気象要素Kindについては，同じ構成で前回の状態を示すLastKind要素を設けたが，さらに，電文の発表によって防災気象要素がどう変化したかをわかり易くするために本要素を設けた． ・対象市町村に発表する警報の変化，対象火山に対する噴火警戒レベル等の変化がわかるよう，それぞれの種類に応じた値を設定できるようにした．
EventDateTimeComment （現象の時刻の解説）	・噴火に関する火山観測報で発表する現象の発生時刻（EventDateTime要素）に関連する補足説明事項を記述できるようEventDateTimeComment要素を設けた．
FormalName （防災気象情報要素名（正式名称））	・噴火警報の名称を気象庁における正式名称で発表する場合に使用するために設けた．
Coordinate（対象火山の位置）	・旧形式電文には含まれていなかった情報だが，基本要素で定義されるCoordinate要素を用いて対象火山の位置を発表することとした． ・要素値は基本要素辞書を利用して，緯度，経度，高度を記述し，description属性で文字列としての表示形式を記述できるようにしている． ・文字列での表示形式は地震・津波電文で用いる記述とあわせ，全角文字を基本とする形式とした．
CraterName（対象火口名称）， CraterCoordinate（対象火口の位置）	・情報を火口別に発表する場合に使用する要素として設けた． ・それぞれName要素，Coordinete要素と同じ形式とした．
AreaFromMark（位置補助情報）	・旧形式電文では設けていないが，噴火の位置等の情報を発表するために設けた．
NextAdvisory（次の情報発表日時の予告）	・火山の状況に関する解説情報において，旧形式電文でも記述している次の発表予定日時を記述するために設けた要素である． ・東海地震の観測情報においても用いることから地震・津波分野個別辞書（jmx_seis）でも設けている．
OtherInfo（その他必要と認める事項）	・新たな要素として追加するまでもないが他の要素には含まれない事項を記述する場合に利用するものとして設けた．
Appendix（補足）	・発表する情報全体に関する補足事項を記述するために設けた．
ColorPlume（有色噴煙）， WhitePlume（白色噴煙）	・噴火に関する火山観測報の旧形式電文においても，本文中ではあるが必須項目として記述してきた観測項目である． ・辞書の作成においては基本的な要素を基本要素辞書に記述し，属性の使い方を気象等の他の観測要素と共通化した．

9 コード表*

9.1 コード表の公開

気象庁 XML に関する技術情報については気象庁 XML の HP にて公開しているが，今回，気象庁 XML の解析に必要なコード表についても同様に公開した．これは，とりうる値（enumeration）の妥当性確認方法として，XML スキーマによりバリデーションを行う方法もあるが，このことによる XML スキーマの長大化によるデメリットを考えると，必ずしも XML スキーマによるものだけである必要はなく，オフライン仕様やテーブル（コード表）の提供による方法なども手段として検討対象とする必要がある．また，いずれの方法でも妥当性を確認できる環境が整えられていれば良いという XML コンソーシアムの助言があった．このことに，「気象庁 XML の解析に必要な情報は全て公開すべき」という全体的な考え方も加えて，コード表を公開することとした．

なお，コード表の公開にあたっては，コード表に関連する情報の特性に留意する．具体的には，公開を前提としていない詳細な住所情報などがある．ただし，地点情報に位置情報は必要であることから，粒度を下げる等により，必要最小限の情報は付加できるように努力する．

9.2 コード表の管理（運用指針 1.2 項）

コード値は運用指針 1.2 項にあるとおり，辞書・XML スキーマとは独立して管理される．これは，辞書・XML スキーマの変更は基本的に気象庁の業務変更によるものであるのに対して，コード値には地点を示すコードなど，他機関に依存するものや定常業務として定期的に変更する運用上の値という場合が多く，辞書や XML フォーマットの変更とコード値の変更とでは変更頻度や変更に伴うシステムへの影響度などが全く異なることから，同等に扱わないことが望ましいことによる．

コード表はコード値そのものの持つ意味や関連するメタ情報との関連を示すテーブルであるが，コード表公開の趣旨である妥当性確認の観点からすれば，XML スキーマによらない妥当性確認方法の一つの資料でもある．このため，どの要素値・属性値のとりうる値にどのコード表が利用され，またその際のバージョンがどのようになっているかという情報は，妥当性確認における重要な情報ともいえる．このことから，各コード表と，コード表を利用する要素・属性などコード表関連のメタ情報を整理して提示するため，コード管理表を作成した．コード管理表にはコード値に対するメタ情報の提供のほか，辞書・XML スキーマの運用とコード表の運用を結びつける目的も持ち合わせている．

コード表の管理において，1 つのコード表を複数部局が担当している場合は，バージョン管理上のデグレード（バージョンダウン）が起きないように関係者間で協力して行う．基本的にはコード値を業務上管理している管理者よりは，その値を利用している利用者側で管理することを基本とする．これは，利用者側でメタ情報を付加する可能性が高いことからである．いずれにしても，気象庁 XML のコード表のためだけの別管理体系とならないようにするため，業務整理を行い，各業務における管理体系との整合性を図れるようにする．

9.3 コード表の統合

気象庁 XML における統一的な利用に当たっては，コード表についても例外とすべきではなく，情報の利用範囲や業務などに応じて，コード表の統合に際して検討を行ってきた．結果として，統合されたコード表は多くないが，その成果の一つとして市町村を示すコード表がある．本来，市町村単位を示すコード表として，全国地方公共団体コード（JIS X 0402）があるが，火山関連情報で利用する八丈支庁などの行政区分や島単位の地域，気象警報・注意報で現地の防災上の理由による同一市町村内での地域分割などには対応できていない．このことから，全国地方公共団体コードの完全互換利用はできず，準拠した上で一部のみ拡張した気象庁独自のコード表を作成して，気象警報・注意報から，地震や火山関係情報まで共通で利用している．

* 第 9 章　杉山

10 バージョン管理*

10.1 要素の追加に関する議論

(1) 要素の追加と UPA 制約

当初，気象庁 XML では，将来の拡張のために W3C XML Schema の any 要素を用いる方法（第 4.3 節（4）項参照）を広く利用することとしていたが，Unique Particle Attribution（UPA）制約（第 4.3 節（5）項参照）により問題となることが判明したため，将来の拡張について精査する必要が生じてきた．

そこで，Temperature 要素が既に定義されているときに，未定義の Humidity 要素を兄弟要素として追加しなければいけなくなったときを事例として，第 10 表のとおり，any 要素を属する名前空間のみに制限する方法（スキーマ例 1），any 要素を別名前空間のみに制限する方法（スキーマ例 2），any 要素の出現場所を一段階掘り下げる方法（スキーマ例 3），any 要素をやめる方法（スキーマ例 4）の 4 種類の対応について UPA 制約に関する技術的検討を行った．

共通辞書（基本要素）で定義している基本要素については原則として要素参照（第 7.1 節（6）項参照）により利用することとしているが，この場合，共通辞書（基本要素）のバージョンアップを行うと，全ての電文が影響を受ける．このため，any 要素を使って未定項目を設定する場合，共通辞書との関連も含めて整理する必要がある．しかしながら，共通辞書（基本要素）については，基本的な辞書なので，any 要素を用いて要素を増やすようなことはせず，名前空間の変更により追加要素を表現する方針となった．このため，4 種類の事例のうち，スキーマ例 2 が他の事例よりも良い候補となりそうであった．

一方，第 8.4 節（3）項で検討した気象要素の型に従い，どのような気象要素にもなれる「部品の any」として anyElementPart 要素の導入につい

第 10 表　要素の追加における any 要素の利用方法による影響

スキーマ例	スキーマ例の概要	バージョンアップ後の電文	影響
理想		<jmx_mete:〇〇Part> <jmx_eb:Temperature>14.5</jmx_eb:Temperature> <jmx_eb:Humidity>50</jmx_eb:Humidity > </jmx_mete:〇〇Part>	W3C XML Schemaの枠組みでは不可能である.
スキーマ例1	any要素を属する名前空間のみに制限		jmx_eb:Temperature要素の出現回数が0回以上であるため, UPA制約違反になる.
スキーマ例2	any要素を別名前空間のみに制限	対応例① <jmx_mete:〇〇Part> <jmx_eb:Temperature>14.5</jmx_eb:Temperature> <jmx_mete:Humidity>50</jmx_mete:Humidity > </jmx_mete:〇〇Part> 対応例② <jmx_mete_V11:〇〇Part> <jmx_eb:Temperature>14.5</jmx_eb:Temperature> <jmx_eb:Humidity>50</jmx_eb:Humidity > </jmx_mete_V11:〇〇Part>	対応例①ではHumidity要素が既にjmx_ebで定義してあるのにjmx_meteでも新たに定義する必要がある. 対応例②では親要素を別空間のanyで定義するする必要がある.
スキーマ例3	any要素の出現場所の階層を一段階深くする	<jmx_mete:〇〇Part> <jmx_eb:Temperature>14.5</jmx_eb:Temperature> <jmx_mete:Addition> <jmx_mete:Humidity>50</ jmx_mete:Humidity > </jmx_mete:Addition> </jmx_mete:〇〇Part>	Humidity要素が既にjmx_ebで定義してあるのにjmx_meteでも新たに定義する必要がある.
スキーマ例4	any要素の利用をやめる	<jmx_mete_V2:〇〇Part> <jmx_eb:Temperature>14.5</jmx_eb:Temperature> <jmx_eb:Humidity>50</jmx_eb:Humidity > </jmx_mete_V2:〇〇Part>	jmx_meteとは名前空間が変わってしまう.

* 第 10 章　第 10.1 節・第 10.7 節　竹田，　第 10.2 節～第 10.6 節　杉山

て検討した（第 18 図）．anyElementPart 要素による対応は，スキーマの any 要素を利用しないという意味でスキーマ例 4 の対応に当たる．この案について検討したところ，anyElementPart 要素の type 属性のとり得る値は検索のキーとなる事項であるため，辞書において列挙して記載すべきであること，又，各要素を示すのは属性値ではなく要素名として表現すべきという原則からも外れることが意見として出され，anyElementPart 要素のような属性値を利用して何にでもなれる要素を用意した未定項目の定義はできないこととした．

議論の過程ではこのような anyElementPart 要素の提案もあったが，最終的には W3C XML Schema の any 要素を利用してバージョンアップを実現することが適切と判断した．そこで，any 要素を利用してバージョンアップするとき，バージョンアップ前後で妥当性検証が想定通りできるかどうかの検討を行い，スキーマ例 2 の事例について，バージョンアップに伴う妥当性検証への影響について議論を重ねた．この結果，辞書のバージョンと妥当性検証のための名前空間の関連について更なる技術的検討が必要なことが分かった．

（2）追加要素辞書

前項の議論は既存の辞書を変更してバージョンアップする方法であったが，any 要素を別名前空間のみに制限するスキーマ例 2 の方法では，既存の辞書ではない辞書に新しい要素の定義をすることができる．このため，個別辞書の any 要素の利用と併せて，バージョンアップに伴うスキーマと電文の妥当性検証に関する互換性（以下本項において「互換性」という．）を満たす方法を検討し，更に第 19 図にまとめたように共通辞書（追加要素）を用いた辞書のバージョンアップに関する検討を行った．この方法は，共通辞書（追加要素）以外の辞書については名前空間と辞書のバージョンを一致させ，共通辞書（追加要素）は名前空間と辞書のバージョンを別に管理し，未定項目として想定していた箇所に要素を追加する場合は名前空間の変わらない共通辞書（追加要素）に対して新たに定義を加えることでバージョンアップ前後の互換性を確保するというものである．また，XMLDic2Schema ツール（第 12.1 節参照）を利用して辞書から作成したスキーマによりバージョンアップ前後の電文の妥当性検証も行った．更に，気象庁 XML 電文の利用者には，辞書のバージョンの変更があってもシステムの対応を極力抑えたい利用者もいれば，より厳密に運用したい利用者もいると考え，前者は any 要素の processContents 属性を“lax”として，後者は“strict”として妥当性検証を行うことを新たに想定して互換性の検証を行った．

このように，共通辞書（追加要素）の検討をきっかけとして，名前空間，スキーマ，インスタンス（電文）がバージョンアップ前後でどのようになるか，妥当性検証の結果はどのようになるべきかなど，運用と互換性の検討を行った（第 20 図）．

また，第 19 図の案と同様に気温だけの電文から構造や湿度が追加されていくというバージョンアップについて，前項のスキーマ例 2 の方法（以下「案 A」とする．）でバージョンアップするときの妥当性検証を行った（第 21 図）．このバージョンアップの方法は，バージョンアップ前の辞書の内容は全て含み，要素を追加した新しい辞書を，バージョンアップ前の辞書と違う名前空間で新たに作成するというバージョンアップの

33	type.AnyElementPart					
34			type	xs:string	1	分類
35		Sentence		xs:token	?	文書形式の表現
36		Base		type.BaseAnyElement	?	卓越もしくは変化前
37		Temporary		type.BaseAnyElement	*	断続現象
38		Becoming		type.BaseAnyElement	*	変化後
39		SubArea		type.SubAreaAnyElement	*	地域
40		jmx_eb:AnyElement		jmx_eb:type.AnyElement	*	任意の要素
41		Time		xs:dateTime	?	起時
42		Remark		xs:string	?	注意事項・付加事項

第 18 図　anyElementPart 型の案

方法である．この方法について検証を行ったところ，any 要素の processContents 属性を“lax”として処理すればメジャーバージョンが同じであれば前方互換，後方互換が保たれるようである．しかし，“strict”として処理する場合は，スキーマの import 要素において指定されるファイルが実際に存在しないとエラーとなる．このことは，スキーマが整理されるメジャーバージョンアップまでの将来に渡り必要なスキーマファイルについて，そのファイルを対象とした全ての import 要素を想定しなければならないことから，結果としてこのバージョンアップの方法が難しいことが確認された．

一方，第 19 図の案では，辞書の Ver1.1 と Ver1.2 で processContents 属性を“lax”として妥当性検証しても電文の Ver1.4 がエラーとなってしまい，後方互換が保たれないという問題があった．このことから，スキーマ例 2 をヒントに，第 19 図の案の方法において，共通辞書（追加要素）の運用を厳密化することで後方互換性を確保できる方法（以下「案 B」とする．）が提案された（第 22 図）．共通辞書（追加要素）で運用を厳密化した事項は次の部分である．

> 「追加要素辞書」の前バージョンの定義には子要素や属性の追加・削除などの変更を行わず，新たな要素を追加する．

この制限を加えることで，電文に構造のない湿度を記述するため HumidityPart 要素を共通辞書（追加要素）に追加（第 22 図の Ver1.2）した後，新たに構造のある湿度を記述（第 22 図の Ver1.3）する必要が生じた場合，同じ湿度を表す要素にも関わらず HumidityPart 要素とは別に HumidityPart2 要素を定義しなければならないものの，any 要素の processContents 属性を“lax”として処理すれば前方互換，後方互換が保たれ，また，processContents 属性を“strict”として処理すればマイナーバージョンアップで要素が追加された電文は妥当性検証でエラーとなり，システムが想定していない要素による悪影響を妥当性検証で防ぐことができることから，想定通りの互換性となる．また，未定項目として any 要素を使うが，とり得る名前空間は共通辞書（追加要素）の名前空間に限定した．このことで，共通辞書（追加要素）以外の辞書において UPA 制約の影響を受けずに未定項目をより柔軟に配置できるようになるとともに，要素の追加においては共通辞書（追加要素）以外の辞書のバージョンアップが不要となり，メジャーバージョンアップとなる事態を抑えられる．更に，案 B によるバージョンアップで作成した共通辞書（追加要素）（第 11 表）を見ると分かるが，マイナーバージョンアップを行う場合には，これまでに定義している行は変更を行わず，必ず最後の行に新しい定義を行うことにより，共通辞書（追加要素）で運用を厳密化した事項を満たせることから，マイナーバージョンアップに伴う管理も容易である．

この検討を踏まえると，案 A におけるスキーマの import 要素の問題については，Ver1.1 の import 要素を Ver1.0 に，Ver1.2 の import 要素を Ver1.1 にと，新規に作成するスキーマには必ず次のバージョンのスキーマファイルを記載すれば解決できるとの意見があった．これについて再検証した結果，any 要素の processContents 属性を“lax”として処理すれば前方互換，後方互換が保たれ，processContents 属性を“strict”として処理すればマイナーバージョンアップで要素が追加された電文はエラーとなり，案 B と同様の結果となった．

これら案 A と案 B について，メリットとデメリットをまとめたものが第 12 表である．

結果として，案 A も案 B も互換性では同様の結果となったが，案 A は名前空間で厳密に管理するのでバージョン管理が明確になるというメリットがあるものの，若干高度なスキーマの知識が必要となることから，今後の運用・管理を考慮して，W3C XML Schema の知識のない担当者でも管理が容易な案 B を採用することとした．なお，二つの案の詳細な検討によりスキーマのバージョン管理に関して明確かつ深い知見が得られたことで，辞書やスキーマの運用管理に関する各規定においてもこれら知見が反映されている．

（1）作成したサンプル電文

jmx_mete においては，Property 要素の子要素と，AdditionalInfo 要素の子要素に将来追加される要素（any）を考慮しておけば，ほとんどの新しい電文を記述できると考えた．

そこで，Property 要素の子孫に「気温」のみが記述してある電文（電文 Ver1.0）から要素が増えていく場合について，辞書や電文がどのように変化していき，それぞれの電文のバージョンと辞書のバージョンによって妥当性検証がどのようになるか調べた．（表 1）

表 1　調査した電文の内容

	気温 列挙あり	気温 列挙なし	湿度 構造なし	湿度 構造あり	付加事項
電文 Ver1.0	○				
電文 Ver1.1	○	○			
電文 Ver1.2	○	○	○		
電文 Ver1.3	○	○	○		○
電文 Ver1.4	○	○	○	○	○
電文 Ver2.0	○	○	○	○	○

調査に利用した電文は別添のとおりだが，変更部分の概要は次のとおりである．

（ア）電文 Ver1.0

電文のバージョン

```
<InfoKindVersion>1.0</InfoKindVersion>
```

Property 要素

```
<Property>
    <TemperaturePart>
        <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
    </TemperaturePart>
</Property>
```

付加事項

```
<AdditionalInfo>
</AdditionalInfo>
```

（イ）電文 Ver1.1

電文のバージョン

```
<InfoKindVersion>1.1</InfoKindVersion>
```

Property 要素に type 属性で列挙にない「最低気温」を拡張（*）を利用して追加

```
<Property>
    <TemperaturePart>
        <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
        <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
    </TemperaturePart>
</Property>
```

付加事項

```
<AdditionalInfo>
</AdditionalInfo>
```

（ウ）電文 Ver1.2

電文のバージョン

```
<InfoKindVersion>1.2</InfoKindVersion>
```

Property 要素に湿度を追加

```
<Property>
    <TemperaturePart>
        <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
        <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
    </TemperaturePart>
    <HumidityPart>
        <jmx_eb:Humidity type="平均湿度" unit="%">50</jmx_eb:Humidity>
```

第 19 図　バージョンアップに伴うスキーマと電文の妥当性検証に関する互換性の確認資料
any 要素を利用した未定項目の絞り込みと併せたもの．（別添資料は省略）

```
        </HumidityPart>
    </Property>
付加事項
    <AdditionalInfo>
    </AdditionalInfo>
```

（エ）電文 Ver1.3

```
電文のバージョン
    <InfoKindVersion>1.3</InfoKindVersion>
Property 要素
    <Property>
        <TemperaturePart>
            <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
            <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
        </TemperaturePart>
        <HumidityPart>
            <jmx_eb:Humidity type="平均湿度" unit="%">50</jmx_eb:Humidity>
        </HumidityPart>
    </Property>
付加事項に TestAddition 要素を追加
    <AdditionalInfo>
        <TestAddition>電文固有の要素追加</TestAddition>
    </AdditionalInfo>
```

（オ）電文 Ver1.4

```
電文のバージョン
    <InfoKindVersion>1.4</InfoKindVersion>
Property 要素の湿度に Base 要素と Temporary 要素の構造を追加
    <Property>
        <TemperaturePart>
            <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
            <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
        </TemperaturePart>
        <HumidityPart>
            <Base>
                <jmx_eb:Humidity type="平均湿度" unit="%">50</jmx_eb:Humidity>
            </Base>
            <Temporary>
                <jmx_eb:Humidity type="最小湿度" unit="%">20</jmx_eb:Humidity>
            </Temporary>
        </HumidityPart>
    </Property>
付加事項
    <AdditionalInfo>
        <TestAddition>電文固有の要素追加</TestAddition>
    </AdditionalInfo>
```

（カ）電文 Ver2.0

```
ネームスペースの変更
    <Report xmlns="http://xml.kishou.go.jp/jmaxml2/" xmlns:jmx="http://xml.kishou.go.jp/jmaxml2/">

    <Schema>
      <Head>
        <Name>防災気象情報型</Name>
        <Version>2.0_0</Version>
        <URI>http://xml.kishou.go.jp/jmaxml2/informationBasis/</URI>
      </Head>
      <Body>
        <Name>気象情報型</Name>
        <Version>2.0_0</Version>
        <URI>http://xml.kishou.go.jp/jmaxml2/body/meteorology/</URI>
      </Body>
    </Schema>

    <Head xmlns="http://xml.kishou.go.jp/jmaxml2/informationBasis/"
      xmlns:jmx_eb="http://xml.kishou.go.jp/jmaxml2/elementBasis/">
```

第 19 図　続き

```
    <Body xmlns="http://xml.kishou.go.jp/jmaxml2/body/meteorology/"
     xmlns:jmx_eb="http://xml.kishou.go.jp/jmaxml2/elementBasis/"
     xmlns:jmx_add="http://xml.kishou.go.jp/jmaxml2/addition/">
電文のバージョン
    <InfoKindVersion>2.0</InfoKindVersion>
Property 要素に追加された要素は jmx_mete と jmx_eb で記述
    <Property>
        <TemperaturePart>
            <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
            <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
        </TemperaturePart>
        <HumidityPart>
            <Base>
                <jmx_eb:Humidity type="平均湿度" unit="%">50</jmx_eb:Humidity>
            </Base>
            <Temporary>
                <jmx_eb:Humidity type="最小湿度" unit="%">20</jmx_eb:Humidity>
            </Temporary>
        </HumidityPart>
    </Property>
AdditionalInfo 要素に追加された要素は jmx_mete で記述
    <AdditionalInfo>
        <TestAddition>電文固有の要素追加</TestAddition>
    </AdditionalInfo>
```

（2）作成した辞書

これらの電文を定義するために，現辞書については次のような変更を行った．

（ア）jmx

「追加要素辞書」のバージョンを電文中に記述するために，Schema 要素の子要素に Addition 要素を追加した．

（イ）jmx_mete

Property 要素と AdditionalInfo 要素の子要素に「##other」をネームスペースに指定した any 要素を追加した．

（ウ）jmx_add

電文のバージョンアップに応じて必要な要素を追加した．電文のバージョンと「追加要素辞書」のバージョンの対応は次のとおり．追加要素辞書の変更を伴わない電文のバージョンアップがあるので，マイナーバージョンの違いに注意が必要．

電文 Ver1.0　→　追加要素辞書 Ver1.0
電文 Ver1.1　→　追加要素辞書 Ver1.0
電文 Ver1.2　→　追加要素辞書 Ver1.1
電文 Ver1.3　→　追加要素辞書 Ver1.2
電文 Ver1.4　→　追加要素辞書 Ver1.3
電文 Ver2.0　→　追加要素辞書 Ver2.0

（3）スキーマによる妥当性検証

エクセルの辞書からスキーマの作成は「XMLDic2Schema」を用いたが，そのままでは jmx_add のインポートがされないので，jmx_mete は次の赤字の部分をテキストエディタで追加した．

```
<?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/jmaxml/elementBasis/"
xmlns:jmx_add="http://xml.kishou.go.jp/jmaxml/addition/"
xmlns:jmx_mete="http://xml.kishou.go.jp/jmaxml/body/meteorology/" elementFormDefault="qualified"
targetNamespace="http://xml.kishou.go.jp/jmaxml/body/meteorology/">
  <xs:import namespace="http://xml.kishou.go.jp/jmaxml/elementBasis/" schemaLocation="jmx_eb.xsd"/>
  <xs:import namespace="http://xml.kishou.go.jp/jmaxml/addition/" schemaLocation="jmx_add.xsd"/>
```

第 19 図　続き

このようにして作成したスキーマを使って妥当性検証を行った．また，「XMLDic2Schema」ではprocessContents 属性が「lax」であるが，この属性を「strict」に変更したスキーまでも妥当性検証を行った．その結果が表２である．

```
<xs:any maxOccurs="unbounded" minOccurs="0" namespace="##other" processContents="lax"/>
<xs:any maxOccurs="unbounded" minOccurs="0" namespace="##other" processContents="strict"/>
```

表２　妥当性検証結果

	辞書 Ver1.0 lax	辞書 Ver1.0 strict	辞書 Ver1.1 lax	辞書 Ver1.1 strict	辞書 Ver1.2 lax	辞書 Ver1.2 strict	辞書 Ver1.3 lax	辞書 Ver1.3 strict	辞書 Ver2.0 lax	辞書 Ver2.0 strict
電文 Ver1.0	○	○	○	○	○	○	○	○	×	×
電文 Ver1.1	○	○	○	○	○	○	○	○	×	×
電文 Ver1.2	○	×	○	○	○	○	○	○	×	×
電文 Ver1.3	○	×	○	×	○	○	○	○	×	×
電文 Ver1.4	○	×	×	×	×	×	○	○	×	×
電文 Ver2.0	×	×	×	×	×	×	×	×	○	○

水色の背景が対応するバージョン

第 19 図　続き

ベース

運用開始後に要素を加えることを考える

名前空間	ts1（本体）	ts1_p1（部品）
スキーマ	【Ver.1.00】 <xs:element name="Doc" type="ts1:type.doc"/> <xs:complexType name="type.doc"> <xs:sequence> <xs:element minOccurs="0" maxOccurs="1" ref="ts1_p1:AP"/> </xs:sequence> </xs:complexType> </xs:element>	【Ver.1.00】 <xs:complexType name="type.P"> <xs:simpleContent> <xs:extension base="xs:float"> <xs:attribute name="p" type="xs:string"/> </xs:extension> </xs:simpleContent> </xs:complexType> <xs:element name="AP" type="ts1_p1:type.P"/> <xs:element name="BP" type="ts1_p1:type.P"/>
インスタンス	【Ver.1.00】 <Doc> <ts1_p1:AP>12.5</ts1_p1:AP>←ここに要素を追加する </Doc>	

名前空間を変えずにバージョンアップ

既存要素追加、“any”を用いない場合　⇒　運用：？、互換：×

名前空間	ts1, ts1_p1	
スキーマ	【Ver.1.00】 Ots1 <xs:sequence> <xs:element minO...="0" maxO...="1" ref="ts1_p1:AP"/> </xs:sequence> Ots1_p1 <xs:element name="AP" type="ts1_p1:type.P"/> <xs:element name="BP" type="ts1_p1:type.P"/>	【Ver.1.01】 Ots1 <xs:sequence> <xs:element minO...="0" maxO...="1" ref="ts1_p1:AP"/> <xs:element minO...="0" maxO...="1" ref="ts1_p1:BP"/> </xs:sequence> Ots1_p1 同左
インスタンス	【Ver.1.00】 <Doc> <ts1_p1:AP>12.5</ts1_p1:AP> </Doc>	【Ver.1.01】 <Doc> <ts1_p1:AP>12.5</ts1_p1:AP> <ts1_p1:BP>25.0</ts1_p1:BP> </Doc>

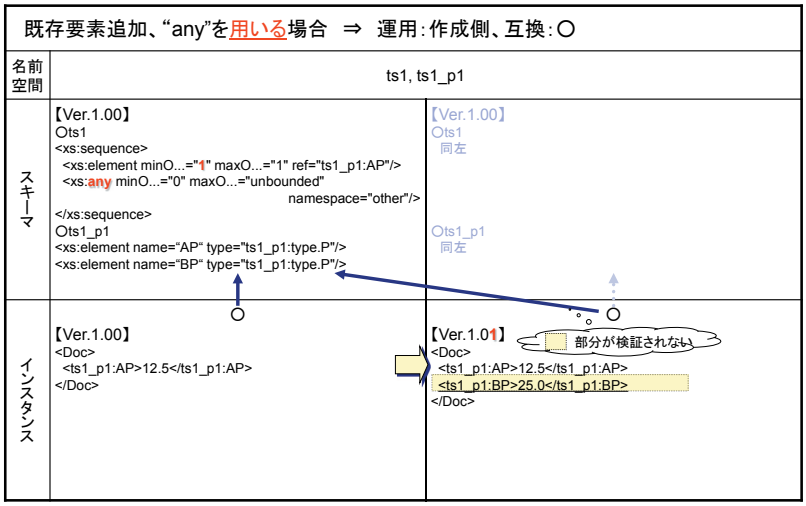
名前空間を変えてバージョンアップ

既存要素追加、“any”を用いない場合　⇒　運用：作成側、互換：×

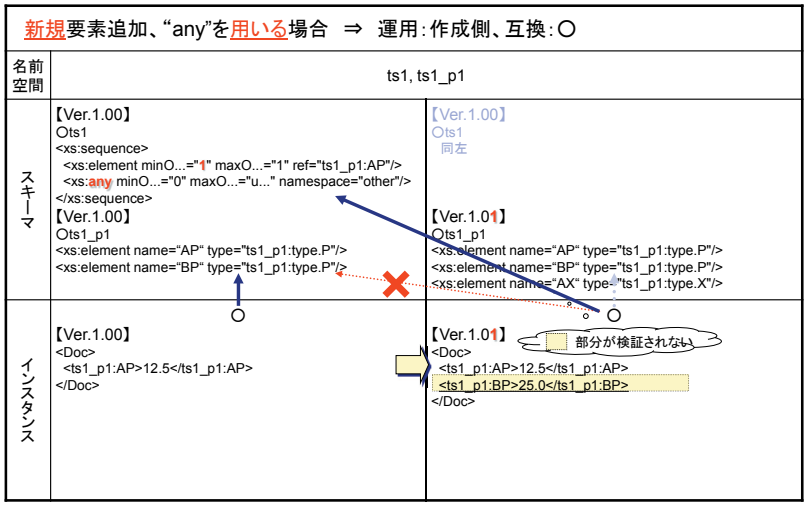
名前空間	ts1 ts1_p1	ts2 ts1_p1
スキーマ	【Ver.1.00】 Ots1 <xs:sequence> <xs:element minO...="0" maxO...="1" ref="ts1_p1:AP"/> </xs:sequence> Ots1_p1 <xs:element name="AP" type="ts1_p1:type.P"/> <xs:element name="BP" type="ts1_p1:type.P"/>	【Ver.2.00】…便宜的 Ots2 <xs:sequence> <xs:element minO...="0" maxO...="1" ref="ts1_p1:AP"/> <xs:element minO...="0" maxO...="1" ref="ts1_p1:BP"/> </xs:sequence> Ots1_p1 同左
インスタンス	【Ver.1.00】 <Doc> <ts1_p1:AP>12.5</ts1_p1:AP> </Doc>	【Ver.2.00】 <Doc> <ts1_p1:AP>12.5</ts1_p1:AP> <ts1_p1:BP>25.0</ts1_p1:BP> </Doc>

第 20 図　バージョンアップにおける運用と互換性のまとめ

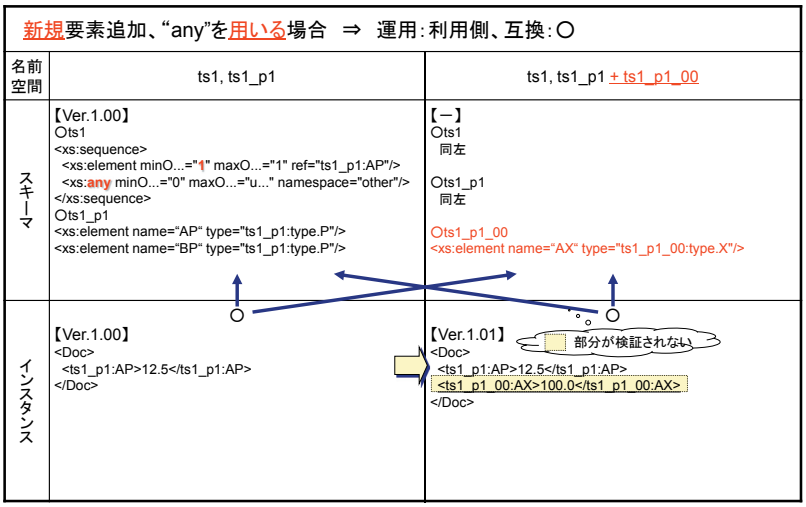
名前空間を変えずにバージョンアップ



名前空間を変えずにバージョンアップ



名前空間を加えてバージョンアップ



第 20 図　続き

（1）作成したサンプル電文

jmx_mete においては，Property 要素の子要素と，AdditionalInfo 要素の子要素に将来追加される要素（any）を考慮しておけば，ほとんどの新しい電文を記述できると考えた．

そこで，Property 要素の子孫に「気温」のみが記述してある電文（電文 Ver1.0）から要素が増えていく場合について，辞書や電文がどのように変化していき，それぞれの電文のバージョンと辞書のバージョンによって妥当性検証がどのようになるか調べた．（表 1）

表 1　調査した電文の内容

	気温 列挙あり	気温 列挙なし	湿度 構造なし	湿度 構造あり	付加事項
電文 Ver1.0	○				
電文 Ver1.1	○	○			
電文 Ver1.2	○	○	○		
電文 Ver1.3	○	○	○		○
電文 Ver1.4	○	○	○	○	○
電文 Ver2.0	○	○	○	○	○

調査に利用した電文は別添のとおりだが，変更部分の概要は次のとおりである．

（ア）電文 Ver1.0

電文のバージョン

```
<InfoKindVersion>1.0</InfoKindVersion>
```

Property 要素

```
<Property>
    <TemperaturePart>
        <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
    </TemperaturePart>
</Property>
```

付加事項

```
<AdditionalInfo>
</AdditionalInfo>
```

（イ）電文 Ver1.1

電文のバージョン

```
<InfoKindVersion>1.1</InfoKindVersion>
```

Property 要素に type 属性で列挙にない「最低気温」を拡張（*）を利用して追加

```
<Property>
    <TemperaturePart>
        <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
        <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
    </TemperaturePart>
</Property>
```

付加事項

```
<AdditionalInfo>
</AdditionalInfo>
```

（ウ）電文 Ver1.2

電文のバージョン

```
<InfoKindVersion>1.2</InfoKindVersion>
```

Property 要素に湿度を追加

```
<Property>
    <TemperaturePart>
        <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
        <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
    </TemperaturePart>
    <jmx_mete11:HumidityPart>
        <jmx_eb11:Humidity type="平均湿度" unit="%">50</jmx_eb11:Humidity>
```

第 21 図　バージョンアップに伴うスキーマと電文の妥当性検証に関する互換性の確認資料
any 要素を別名前空間のみに制限する方法．（別添資料は省略）

```
            </jmx_mete11:HumidityPart>
        </Property>
    付加事項
        <AdditionalInfo>
        </AdditionalInfo>

（エ）電文 Ver1.3
電文のバージョン
        <InfoKindVersion>1.3</InfoKindVersion>
Property 要素
        <Property>
            <TemperaturePart>
                <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
                <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
            </TemperaturePart>
            <jmx_mete11:HumidityPart>
                <jmx_eb11:Humidity type="平均湿度" unit="%">50</jmx_eb11:Humidity>
            </jmx_mete11:HumidityPart>
        </Property>
付加事項に TestAddition 要素を追加
        <AdditionalInfo>
            <jmx_mete12:TestAddition>電文固有の要素追加</jmx_mete12:TestAddition>
        </AdditionalInfo>

（オ）電文 Ver1.4
電文のバージョン
        <InfoKindVersion>1.4</InfoKindVersion>
Property 要素の湿度に Base 要素と Temporary 要素の構造のある HumidityPart2 要素を追加
        <Property>
            <TemperaturePart>
                <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
                <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
            </TemperaturePart>
            <jmx_mete11:HumidityPart>
                <jmx_eb11:Humidity type="平均湿度" unit="%">50</jmx_eb11:Humidity>
            </jmx_add:HumidityPart>
            <jmx_mete13:HumidityPart2>
                <jmx_mete13:Base>
                    <jmx_eb11:Humidity type="平均湿度" unit="%">50</jmx_eb11:Humidity>
                </jmx_mete13:Base>
                <jmx_mete13:Temporary>
                    <jmx_eb11:Humidity type="最小湿度" unit="%">20</jmx_eb11:Humidity>
                </jmx_mete13:Temporary>
            </jmx_mete13:HumidityPart2>
        </Property>
    付加事項
        <AdditionalInfo>
            <jmx_mete12:TestAddition>電文固有の要素追加</jmx_mete12:TestAddition>
        </AdditionalInfo>

（カ）電文 Ver2.0
ネームスペースの変更
        <Report xmlns="http://xml.kishou.go.jp/jmaxml2/" xmlns:jmx="http://xml.kishou.go.jp/jmaxml2/">

        <Schema>
          <Head>
            <Name>防災気象情報型</Name>
            <Version>2.0_0</Version>
            <URI>http://xml.kishou.go.jp/jmaxml2/informationBasis/</URI>
          </Head>
          <Body>
            <Name>気象情報型</Name>
            <Version>2.0_0</Version>
            <URI>http://xml.kishou.go.jp/jmaxml2/body/meteorology/</URI>
          </Body>
        </Schema>
```

第 21 図　続き

```
<Head xmlns="http://xml.kishou.go.jp/jmaxml2/informationBasis/"
xmlns:jmx_eb="http://xml.kishou.go.jp/jmaxml2/elementBasis/">

<Body xmlns="http://xml.kishou.go.jp/jmaxml2/body/meteorology/"
xmlns:jmx_eb="http://xml.kishou.go.jp/jmaxml2/elementBasis/"
xmlns:jmx_add="http://xml.kishou.go.jp/jmaxml2/addition/">
```

電文のバージョン

```
<InfoKindVersion>2.0</InfoKindVersion>
```

Property 要素に追加された要素は jmx_mete と jmx_eb で記述．湿度は HumidityPart2 要素を HumidityPart に変更し，互換性のため重複した内容を記述していた Ver1 系の HumidityPart は削除

```
<Property>
    <TemperaturePart>
        <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
        <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
    </TemperaturePart>
    <HumidityPart>
        <Base>
            <jmx_eb:Humidity type="平均湿度" unit="%">50</jmx_eb:Humidity>
        </Base>
        <Temporary>
            <jmx_eb:Humidity type="最小湿度" unit="%">20</jmx_eb:Humidity>
        </Temporary>
    </HumidityPart>
</Property>
```

AdditionalInfo 要素に追加された要素は jmx_mete で記述

```
<AdditionalInfo>
    <TestAddition>電文固有の要素追加</TestAddition>
</AdditionalInfo>
```

（2）作成した辞書

これらの電文を定義するために，現辞書については次のような変更を行った．

（ア）jmx_mete

Property 要素と AdditionalInfo 要素の子要素に「##other」を名前空間（namespace）に指定した any 要素を追加した．

電文 Ver1.0　→　逐次変更辞書 Ver1.0
電文 Ver1.1　→　逐次変更辞書 Ver1.0
電文 Ver1.2　→　逐次変更辞書 Ver1.1
電文 Ver1.3　→　未作成
電文 Ver1.4　→　未作成
電文 Ver2.0　→　未作成

（3）スキーマによる妥当性検証

スキーマの作成は「XMLDic2Schema_alfa2.0_fat.jar」を利用し，オプションは「-a normal -e normal」（スキーマの出力で，xs:any 要素の namespace 属性値については記述どおり出力，enumeration は各指定内容を union 結合して出力）を指定した．また，「XMLDic2Schema」では processContents 属性が「lax」であるが，この属性を「strict」に変更したスキーマでも妥当性検証を行った．その結果が表 2 である．

```
<xs:any maxOccurs="unbounded" minOccurs="0" namespace="##other" processContents="lax"/>
<xs:any maxOccurs="unbounded" minOccurs="0" namespace="##other" processContents="strict"/>
```

また，processContents を strict にして検証する場合，対応するバージョンの電文でも追加した新しい辞書について，jmx_mete もしくは jmx の次の赤字のように追加して記述しないとエラーとなったため，手書きで書き加えた．

```
<?xml version="1.0" encoding="UTF-8"?><xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:jmx="http://xml.kishou.go.jp/jmaxml/"
 xmlns:jmx_ib="http://xml.kishou.go.jp/jmaxml/informationBasis/"
 xmlns:jmx_mete="http://xml.kishou.go.jp/jmaxml/body/meteorology/"
```

第 21 図　続き

```
xmlns:jmx_mete11="http://xml.kishou.go.jp/jmaxml/body/meteorology/1.1/"
elementFormDefault="qualified" targetNamespace="http://xml.kishou.go.jp/jmaxml/">
  <xs:import namespace="http://xml.kishou.go.jp/jmaxml/body/meteorology/" schemaLocation="jmx_mete.xsd"/>
  <xs:import namespace="http://xml.kishou.go.jp/jmaxml/informationBasis/" schemaLocation="jmx_ib.xsd"/>
  <xs:import namespace="http://xml.kishou.go.jp/jmaxml/body/meteorology/1.1/" schemaLocation="jmx_mete11.xsd"/>
```

表２　妥当性検証結果

	辞書 Ver1.0 lax	辞書 Ver1.0 strict	辞書 Ver1.1 lax	辞書 Ver1.1 strict	辞書 Ver1.2 lax	辞書 Ver1.2 strict	辞書 Ver1.3 lax	辞書 Ver1.3 strict	辞書 Ver2.0 lax	辞書 Ver2.0 strict
電文 Ver1.0	○	○	○	○						
電文 Ver1.1	○	○	○	○						
電文 Ver1.2	○	×	○	○						
電文 Ver1.3	○	×	○	×						
電文 Ver1.4	○	×	○	×						
電文 Ver2.0	×	×	×	×						

水色の背景が対応するバージョン

第 21 図　続き

（1）作成したサンプル電文

jmx_mete においては，Property 要素の子要素と，AdditionalInfo 要素の子要素に将来追加される要素（any）を考慮しておけば，ほとんどの新しい電文を記述できると考えた．

そこで，Property 要素の子孫に「気温」のみが記述してある電文（電文 Ver1.0）から要素が増えていく場合について，辞書や電文がどのように変化していき，それぞれの電文のバージョンと辞書のバージョンによって妥当性検証がどのようになるか調べた．（表 1）

表 1　調査した電文の内容

	気温 列挙あり	気温 列挙なし	湿度 構造なし	湿度 構造あり	付加事項
電文 Ver1.0	○				
電文 Ver1.1	○	○			
電文 Ver1.2	○	○	○		
電文 Ver1.3	○	○	○		○
電文 Ver1.4	○	○	○	○	○
電文 Ver2.0	○	○	○	○	○

調査に利用した電文は別添のとおりだが，変更部分の概要は次のとおりである．

（ア）電文 Ver1.0

電文のバージョン

```
<InfoKindVersion>1.0</InfoKindVersion>
```

Property 要素

```
<Property>
    <TemperaturePart>
        <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
    </TemperaturePart>
</Property>
```

付加事項

```
<AdditionalInfo>
</AdditionalInfo>
```

（イ）電文 Ver1.1

電文のバージョン

```
<InfoKindVersion>1.1</InfoKindVersion>
```

Property 要素に type 属性で列挙にない「最低気温」を拡張（*）を利用して追加

```
<Property>
    <TemperaturePart>
        <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
        <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
    </TemperaturePart>
</Property>
```

付加事項

```
<AdditionalInfo>
</AdditionalInfo>
```

（ウ）電文 Ver1.2

電文のバージョン

```
<InfoKindVersion>1.2</InfoKindVersion>
```

Property 要素に湿度を追加

```
<Property>
    <TemperaturePart>
        <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
        <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
    </TemperaturePart>
    <jmx_add:HumidityPart>
        <jmx_add:Humidity type="平均湿度" unit="%">50</jmx_add:Humidity>
```

第 22 図　バージョンアップに伴うスキーマと電文の妥当性検証に関する互換性の確認資料
共通辞書（追加要素）の運用を厳格化する方法．（別添資料は省略）

```
        </jmx_add:HumidityPart>
    </Property>
付加事項
    <AdditionalInfo>
    </AdditionalInfo>
```

（エ）電文 Ver1.3

```
電文のバージョン
    <InfoKindVersion>1.3</InfoKindVersion>
Property 要素
    <Property>
        <TemperaturePart>
            <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
            <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
        </TemperaturePart>
        <jmx_add:HumidityPart>
            <jmx_add:Humidity type="平均湿度" unit="%">50</jmx_add:Humidity>
        </jmx_add:HumidityPart>
    </Property>
付加事項に TestAddition 要素を追加
    <AdditionalInfo>
        <jmx_add:TestAddition>電文固有の要素追加</jmx_add:TestAddition>
    </AdditionalInfo>
```

（オ）電文 Ver1.4

```
電文のバージョン
    <InfoKindVersion>1.4</InfoKindVersion>
Property 要素の湿度に Base 要素と Temporary 要素の構造のある HumidityPart2 要素を追加
    <Property>
        <TemperaturePart>
            <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
            <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
        </TemperaturePart>
        <jmx_add:HumidityPart>
            <jmx_add:Humidity type="平均湿度" unit="%">50</jmx_add:Humidity>
        </jmx_add:HumidityPart>
        <jmx_add:HumidityPart2>
            <jmx_add:Base>
                <jmx_add:Humidity type="平均湿度" unit="%">50</jmx_add:Humidity>
            </jmx_add:Base>
            <jmx_add:Temporary>
                <jmx_add:Humidity type="最小湿度" unit="%">20</jmx_add:Humidity>
            </jmx_add:Temporary>
        </jmx_add:HumidityPart2>
    </Property>
付加事項
    <AdditionalInfo>
        <jmx_add:TestAddition>電文固有の要素追加</jmx_add:TestAddition>
    </AdditionalInfo>
```

（カ）電文 Ver2.0

```
ネームスペースの変更
    <Report xmlns="http://xml.kishou.go.jp/jmaxml2/" xmlns:jmx="http://xml.kishou.go.jp/jmaxml2/">

    <Schema>
      <Head>
        <Name>防災気象情報型</Name>
        <Version>2.0_0</Version>
        <URI>http://xml.kishou.go.jp/jmaxml2/informationBasis/</URI>
      </Head>
      <Body>
        <Name>気象情報型</Name>
        <Version>2.0_0</Version>
        <URI>http://xml.kishou.go.jp/jmaxml2/body/meteorology/</URI>
      </Body>
    </Schema>
```

第 22 図　続き

```
<Head xmlns="http://xml.kishou.go.jp/jmaxml2/informationBasis/"
xmlns:jmx_eb="http://xml.kishou.go.jp/jmaxml2/elementBasis/">

<Body xmlns="http://xml.kishou.go.jp/jmaxml2/body/meteorology/"
xmlns:jmx_eb="http://xml.kishou.go.jp/jmaxml2/elementBasis/"
xmlns:jmx_add="http://xml.kishou.go.jp/jmaxml2/addition/">
```

電文のバージョン

```
<InfoKindVersion>2.0</InfoKindVersion>
```

Property 要素に追加された要素は jmx_mete と jmx_eb で記述．湿度は HumidityPart2 要素を HumidityPart に変更し，互換性のため重複した内容を記述していた Ver1 系の HumidityPart は削除

```
<Property>
    <TemperaturePart>
        <jmx_eb:Temperature type="気温" unit="度">14.5</jmx_eb:Temperature>
        <jmx_eb:Temperature type="最低気温" unit="度">10.5</jmx_eb:Temperature>
    </TemperaturePart>
    <HumidityPart>
        <Base>
            <jmx_eb:Humidity type="平均湿度" unit="%">50</jmx_eb:Humidity>
        </Base>
        <Temporary>
            <jmx_eb:Humidity type="最小湿度" unit="%">20</jmx_eb:Humidity>
        </Temporary>
    </HumidityPart>
</Property>
```

AdditionalInfo 要素に追加された要素は jmx_mete で記述

```
<AdditionalInfo>
    <TestAddition>電文固有の要素追加</TestAddition>
</AdditionalInfo>
```

（2）作成した辞書

これらの電文を定義するために，現辞書については次のような変更を行った．

（ア）jmx

「追加要素辞書」のバージョンを電文中に記述するために，Schema 要素の子要素に Addition 要素を追加した．若干テクニカルであるが，自動で次の import 文が作成されるように Body 部の any で名前空間 http://xml.kishou.go.jp/jmaxml/body/meteorology/ と http://xml.kishou.go.jp/jmaxml/addition/ をとれるような記述にした．

```
<xs:import namespace="http://xml.kishou.go.jp/jmaxml/addition/" schemaLocation="jmx_add.xsd"/>
```

（イ）jmx_mete

Property 要素と AdditionalInfo 要素の子要素に「http://xml.kishou.go.jp/jmaxml/addition/」をネームスペースに指定した any 要素を追加した．

（ウ）jmx_add

電文のバージョンアップに応じて必要な要素を追加した．電文のバージョンと「追加要素辞書」のバージョンの対応は次のとおり．追加要素辞書の変更を伴わない電文のバージョンアップがあるので，マイナーバージョンの違いに注意が必要．

電文 Ver1.0 → 追加要素辞書 Ver1.0
電文 Ver1.1 → 追加要素辞書 Ver1.0
電文 Ver1.2 → 追加要素辞書 Ver1.1
電文 Ver1.3 → 追加要素辞書 Ver1.2
電文 Ver1.4 → 追加要素辞書 Ver1.3
電文 Ver2.0 → 追加要素辞書 Ver2.0

（3）スキーマによる妥当性検証

このようにして作成したスキーマを使って妥当性検証を行った．スキーマの作成は「XMLDic2Schema_alfa2.0_fat.jar」を利用し，オプションは「-a uri -e normal」を指定した．また，

第 22 図 続き

「XMLDic2Schema」では processContents 属性が「lax」であるが，この属性を「strict」に変更したスキーまでも妥当性検証を行った．その結果が表２である．

```
<xs:any maxOccurs="unbounded" minOccurs="0" namespace="http://xml.kishou.go.jp/jmaxml/addition/" processContents="lax"/>
<xs:any maxOccurs="unbounded" minOccurs="0" namespace="http://xml.kishou.go.jp/jmaxml/addition/" processContents="strict"/>
```

表２　妥当性検証結果

	辞書 Ver1.0	辞書 Ver1.0	辞書 Ver1.1	辞書 Ver1.1	辞書 Ver1.2	辞書 Ver1.2	辞書 Ver1.3	辞書 Ver1.3	辞書 Ver2.0	辞書 Ver2.0
	lax	strict	lax	strict	lax	strict	lax	strict	lax	strict
電文 Ver1.0	○	○	○	○	○	○	○	○	×	×
電文 Ver1.1	○	○	○	○	○	○	○	○	×	×
電文 Ver1.2	○	×	○	○	○	○	○	○	×	×
電文 Ver1.3	○	×	○	×	○	○	○	○	×	×
電文 Ver1.4	○	×	○	×	○	×	○	○	×	×
電文 Ver2.0	×	×	×	×	×	×	×	×	○	○

水色の背景が対応するバージョン

第 22 図　続き

第 11 表　案 B によるバージョンアップで作成した共通辞書（追加要素）

接頭辞	jmx_add	URI	http://xml.kishou.go.jp/jmaxml/addition/				

項番	親要素	子要素	属性	基底型	出現回数	意味	解説
1	type.HumidityPart						Ver1.1で追加
2		Humidity		type.Humidity	*	湿度	Ver1.1で追加
3	type.Humidity			xs:float			Ver1.1で追加
4			type	xs:string	1	分類	Ver1.1で追加
5			unit	xs:string	?	単位	Ver1.1で追加
6			refID	xs:unsignedByte	?	時系列での参照番号	Ver1.1で追加
7			condition	xs:string	?		Ver1.1で追加
8	(element)	HumidityPart		type.HumidityPart	1		Ver1.1で追加
9	(element)	TestAddition		xs:string	1	テスト用の付加要素	Ver1.2で追加
10	type.HumidityPart2						Ver1.3で追加
11		Sentence		xs:token	?	文章形式の表現	Ver1.3で追加
12		Base		type.BaseHumidity	?	卓越もしくは変化前	Ver1.3で追加
13		Temporary		type.BaseHumidity	*	断続現象	Ver1.3で追加
14		Becoming		type.BaseHumidity	*	変化後	Ver1.3で追加
15		SubArea		type.SubAreaHumidity	*	地域	Ver1.3で追加
16		Humidity		type.Humidity	*	湿度	Ver1.3で追加
17		Time		xs:dateTime	?	起時	Ver1.3で追加
18		Remark		xs:string	?	注意事項・付加事項	Ver1.3で追加
19	type.SubAreaHumidity						Ver1.3で追加
20		AreaName		xs:token	?	地域の名称	Ver1.3で追加
21		Sentence		xs:token	?	文章形式の表現	Ver1.3で追加
22		Base		type.BaseHumidity	?	卓越もしくは変化前	Ver1.3で追加
23		Temporary		type.BaseHumidity	*	断続現象	Ver1.3で追加
24		Becoming		type.BaseHumidity	*	変化後	Ver1.3で追加
25		Local		type.LocalHumidity	*	地域	Ver1.3で追加
26		Humidity		type.Humidity	*	湿度	Ver1.3で追加
27		Time		xs:dateTime	?	起時	Ver1.3で追加
28		Remark		xs:string	?	注意事項・付加事項	Ver1.3で追加
29	type.BaseHumidity						Ver1.3で追加
30		TimeModifier		xs:token	?	変化を表す時要素	Ver1.3で追加
31		Humidity		type.Humidity	*	湿度	Ver1.3で追加
32		Local		type.LocalHumidity	*	地域	Ver1.3で追加
33		Time		xs:dateTime	?	起時	Ver1.3で追加
34		Remark		xs:string	?	注意事項・付加事項	Ver1.3で追加
35	type.LocalHumidity						Ver1.3で追加
36		AreaName		xs:token	?	地域の名称	Ver1.3で追加
37		Sentence		xs:token	?	文章形式の表現	Ver1.3で追加
38		Humidity		type.Humidity	*	湿度	Ver1.3で追加
39		Time		xs:dateTime	?	起時	Ver1.3で追加
40		Remark		xs:string	?	注意事項・付加事項	Ver1.3で追加
41	(element)	HumidityPart2		type.HumidityPart2	1		Ver1.3で追加
42	(end)						

第 12 表　バージョンアップに伴うスキーマと電文の妥当性検証に関する互換性の確認
案 A と案 B のまとめ.

案	メリット	デメリット
案A	・名前空間で厳密に管理するのでバージョン管理が明確.	・importをどう対応していくか（案Bでも同じ問題があるが1回のみなので問題とならない. ただしこれを許容するならば案Aでも対応方法がある）. ・大量の名前空間が出来ることによる管理の問題.
案B	・import問題は1回だけ追加するという方針であることから事前に対策が取れるので問題とならない. ・管理すべき名前空間は少なくて済む.	・importをどう対応していくか（案Aと同じ問題だが1回のみなので平易に対応可能）. ・マイナーバージョンの管理と運用が徹底できるか. ・"lax"の対応部分で若干きれいではない判定がされている.

10.2　辞書･XML スキーマのバージョン管理（仕様 2.3 項，運用指針 1.1 項）

前節における検討が行われる前の当初において，バージョン番号は XML 電文として配信される電文（XML インスタンス）に対応する XML スキーマのバージョン番号として記述することを想定していた．しかしながら，前項における議論等を経た結果，バージョン番号は XML インスタンスの運用に対して管理番号を振ることとした．これら辞書・XML スキーマのバージョン管理については，仕様 2.3 項においてバージョンアップとバージョン番号の考え方，及び共通辞書（追加要素）による対応について概要を，運用指針 1.1 項でバージョンアップの内容に応じたバージョン番号や業務的運用について，具体的に解説をしている．

また，バージョン番号の付与ルールについては，当初から「上位バージョン番号」+ “.” +「下位バージョン番号」+ “_” +「リビジョン番号」の「A.B_C」形式を案として取り入れていたが，これをそのまま踏襲して，

- 上位バージョン番号は，メジャー番号のうち名前空間を変えるような大きな変更の際に利用する．
- 下位バージョン番号は，メジャー番号のうち新しい XML スキーマが古い XML インスタンスに対して互換するような名前空間を変更しない XML スキーマの変更の際に利用する．
- リビジョン番号は，XML スキーマの互換性が必ず維持されるような変更及びインスタンスの運用のみの変更の際に利用する．

とすることとした．特に，リビジョン番号のみの変更となるような XML スキーマの互換性が維持される変更においては，「XML スキーマのバージョンは変更されない」という認識で整理がされた．また，いずれの変更においても，XML スキーマの注釈行にて変更履歴を記述することとした．なお，運用開始後，実際にリビジョン番号を変更するバージョンアップを行った際に，何も変更が無い XML スキーマを注釈行の追記だけで変更してよいかという議論を行い，追記しないことに変更した．このため，変更履歴の記述については再検討を行っている．

10.3　共通辞書（追加要素）の使い方（仕様 2.3.1 項，運用指針 1.1.2.3 項）

共通辞書（追加要素）を用いたバージョンアップについては，運用指針 1.1.2.3 項にてマイナーバージョンアップ 3 として解説している．マイナーバージョンアップ 3 は，主として以下のような状況を想定いる．

- 特定の電文のみにおいて，全く新しい要素を追加する．
- 他の電文には影響を与えない．

本バージョンアップにおいて，辞書・XML スキーマを具体的に以下のとおり対応する．

- 共通辞書（追加要素）の XML スキーマ（jmx_add.xsd）に改変を行う．
- 同 XML スキーマの注釈行においてバージョン番号の増加（マイナー番号）と変更内容の解説を付記する．
- 共通辞書（追加要素）を取り込む側の XML

スキーマに改変はない．

・対象とする特定電文については，“jmx_ib:InfoKindVersion”のバージョン番号の増加（マイナー番号）を行う．

・それ以外の電文について改変はない．

この対応の結果，利用者における電文の運用は以下のとおりとなる．

・対象となる特定の電文の利用者で追加要素を利用せずにすませる利用者については，基本的に特段の対応をしなくても利用可能（“processContents="lax"”の定義により正誤が判定できなくてもエラーとならない対応が可能）．

・対象となる特定電文の利用者で追加要素を利用する利用者については，共通辞書（追加要素）の XML スキーマ（jmx_add.xsd）の差し替えをするとともに，当該要素に関わる処理プログラムを改修する．なお，一部のシステムでは XML スキーマの差し替えなしで動作する可能性がある．

・対象となる特定電文以外の利用については，対応不要．

10.4 メジャーバージョンアップ 1 の XML インスタンス運用（運用指針 1.1.2.4 項）

メジャーバージョンアップ 1 において，対応する XML スキーマは名前空間を維持したまま改変される．一方で運用指針 1.1.2 項のとおり，変更で影響を受けない電文については，そのままのバージョン番号を維持する．このため，変更電文以外のその他の電文利用者にとっては新 XML スキーマファイルを更新する必要は原則としてない．ただし，バージョンの管理は“jmx_ib:InfoKind”単位で実施していることから，後日その他の電文における変更があった場合に，以前の変更部分も含めて対応が必要となる可能性はある．

10.5 メジャーバージョンアップ 2 について（運用指針 1.1.2.5 項）

メジャーバージョンアップ 2 は名前空間の変更が発生することから，互換性が一切維持されない．これは内容的には変更が発生されない管理部においても，名前空間定義の変更が発生することから，これまでとは互換性がなくなる．このため，メジャーバージョンアップ 2 の実施については，事案がある電文関係に限らず，“jmx_mete”，“jmx_seis”，“jmx_volc”，“jmx_eb”等全てにおいて要素の拡充・改変等を検討し，一斉整理する機会として期間を掛けて実施する．

10.6 XML インスタンスと XML スキーマのライフサイクル

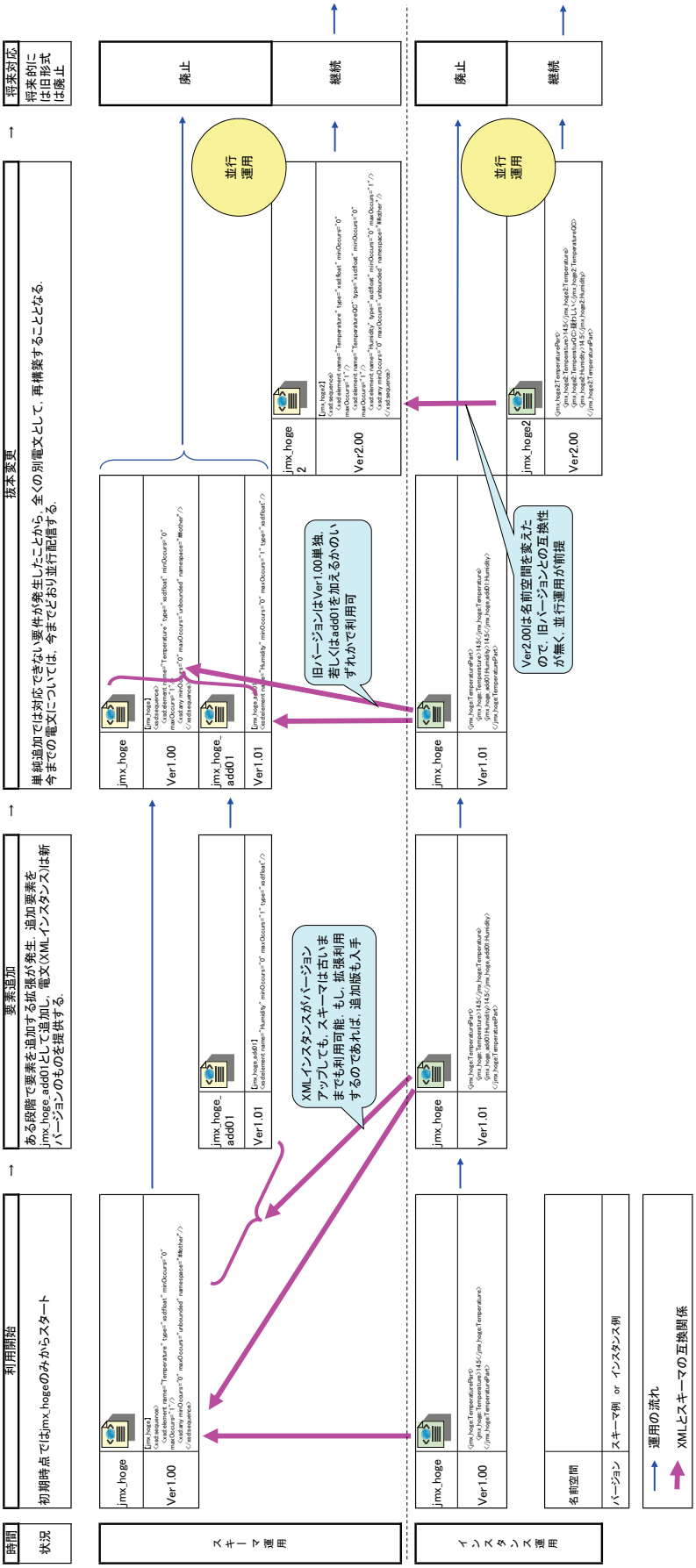
バージョン管理として，XML インスタンスと XML スキーマにおけるバージョン番号の変移を第 23 図にまとめる．

10.7 バージョンアップ対応の事例

初めての気象庁 XML のバージョンアップは，指定河川洪水予報の電文の様式変更に伴い平成 22 年 8 月 6 日に実施した．

指定河川洪水予報は，国土交通省及び都道府県と共同で実施していることから，その改善には気象庁内の調整のほかに，国土交通省等の部外機関との調整が必要であるため，気象庁 XML の仕様検討とは別に検討・調整がされた．新たな洪水予報の様式については，平成 22 年 2 月頃に調整がまとまりつつあった．そこで，指定河川洪水予報の改善内容を気象庁 XML に反映するよう，辞書の一部追加・変更等を行うこととしたいとの要望が予報課から上がった．これまで議論してきたメジャーバージョンアップの影響度や重要性から勘案すると，現時点でメジャーバージョンアップを行うことは極力避けるべきである．このため，どうしても現在の辞書で記述できない事項については，共通辞書（追加要素）に定義してマイナーバージョンアップができるかどうか検討し，それでも対応できない場合は，個別辞書（気象分野）の Body 要素の子要素の AdditionalInfo 要素の中に記述する方法をとるべきであった．

そこで，とりうる値の変更のみで辞書の変更を伴わない案 1（第 24 図）と辞書の変更を伴う案 2（第 25 図）を予報課にて作成し，関係者間での検討を行った．この結果，本件バージョンアップ時点で気象庁 XML は正式運用開始前であること



第 23 図　バージョンアップにおける XML インスタンスと XML スキーマの番号管理

```
<MeteorologicalInfos type="はん濫水の予報">
 <MeteorologicalInfo>
  <DateTime significant="yyyy-mm-ddThh"
     >2010-02-27T09:00:00+09:00</DateTime>
   <Item>
    <Kind>
     <Property>
      <Type>はん濫水</Type>
      <Text type="想定到達時刻">○時間後(○月○日○時頃)</Text>
      <Text type="浸水最深時刻">○時間後(○月○日○時頃)</Text>
      <Text type="想定最大浸水深">0～0.5m 未満</Text>
     </Property>
    </Kind>
    <Area>
     <Name>○市市役所</Name>
    </Area>
   </Item>
   <Item>
    <Kind>
     <Property>
      <Type>はん濫水</Type>
      <Text type="想定到達時刻">○時間後(○月○日○時頃)</Text>
      <Text type="浸水最深時刻">○時間後(○月○日○時頃)</Text>
      <Text type="想定最大浸水深">0.5～1.0m 未満</Text>
     </Property>
    </Kind>
    <Area>
     <Name>○市郵便局</Name>
    </Area>
   </Item>


<MeteorologicalInfos type="水位情報">
 <MeteorologicalInfo>
  <DateTime significant="yyyy-mm-ddThh:mm"
     >2010-02-27T09:00:00+09:00</DateTime>
   <Item>
    <Kind>
     <Status>上昇</Status>
     <DateTime type="現況"
        >2010-02-27T09:00:00+09:00</DateTime>
     <Property>
      <Type>水位</Type>
      <WaterLevelPart>
       <jmx_eb:WaterLevel type="水位" unit="m"
       condition="正常">143.00</jmx_eb:WaterLevel>
       <jmx_eb:WaterLevel
       type="レベル">2</jmx_eb:WaterLevel>
      </WaterLevelPart>
     </Property>
    </Kind>
    <Kind>
     <DateTime type="1 時間後
        ">2010-02-27T10:00:00+09:00</DateTime>
     <Property>
      <Type>水位</Type>
      <WaterLevelPart>
       <jmx_eb:WaterLevel type="水位" unit="m"
       condition="正常">144.80</jmx_eb:WaterLevel>
       <jmx_eb:WaterLevel
       type="レベル">3</jmx_eb:WaterLevel>
      </WaterLevelPart>
     </Property>
    </Kind>
    <Kind>
     <DateTime type="2 時間後"
        >2010-02-27T11:00:00+09:00</DateTime>
     <Property>
      <Type>水位</Type>
      <WaterLevelPart>
       <jmx_eb:WaterLevel type="水位" unit="m"
       condition="正常">145.00</jmx_eb:WaterLevel>
       <jmx_eb:WaterLevel
       type="レベル">4</jmx_eb:WaterLevel>
      </WaterLevelPart>
     </Property>
    </Kind>
    <Kind>
     <DateTime type="3 時間後
        ">2010-02-27T12:00:00+09:00</DateTime>
     <Property>
      <Type>水位</Type>
      <WaterLevelPart>
       <jmx_eb:WaterLevel type="水位" unit="m"
       condition="正常">144.80</jmx_eb:WaterLevel>
       <jmx_eb:WaterLevel
       type="レベル">3</jmx_eb:WaterLevel>
      </WaterLevelPart>
     </Property>
    </Kind>
    <Station>
     <Name>○○水位観測所</Name>
     <Code type="水位観測所">00000</Code>
     <Location>○○市○○</Location>
    </Station>
   </Item>
```

第 24 図　指定河川洪水予報のバージョンアップにおける変更案 1
とりうる値の変更のみで辞書の変更を伴わない方法.

から，影響が少ないのであれば，電文の様式は可能な限り理想に近い形式が望ましいとの方針により，辞書の変更を伴うメジャーバージョンアップとして対応することとなった.

この方針に基づき，共通辞書（基本要素）と個別辞書（気象分野）の改変を含むバージョンアップの方法について検討を行った．この中で，指定河川洪水予報だけでなく，今後のバージョンアップの際にも留意しなければならない一般的な辞書の管理に関する留意事項として，次の 2 点が挙げられた.

一点目は, 予報課が示した案 2 で, 共通辞書（基本要素）の type.WaterLevel 型（第 25 図の A 部分）には，他の基本要素にない level 属性と difference 属性が追加されているが，共通辞書（基本要素）は汎用的に使うことを目的とするため，属性値に関しては原則として他の要素と共通とするという方針である.

```
<MeteorologicalInfos type="はん濫水の予報">
 <MeteorologicalInfo>
  <DateTime significant="yyyy-mm-ddThh"
     >2010-02-27T09:00:00+09:00</DateTime>
   <Item>
    <Kind>
     <Property>
      <Type>はん濫水</Type>
      <FloodAssumptionTable>
       <Area>
        <Name>○○川</Name>
        <Code>0000000000</Code>
       </Area>
       <FloodAssumptionPart>
        <FloodAssumptionArea
          >○市市役所</FloodAssumptionArea>
        <AttainmentTime>○時間後(○月○日○時頃)
        </AttainmentTime>
        <AssumptionFlood>0～0.5m未満</AssumptionFlood>
        <AttainmentDeepestTime
          >○時間後(○月○日○時頃)</AttainmentDeepestTime>
       </FloodAssumptionPart>
       <FloodAssumptionPart>
        <FloodAssumptionArea
          >○市郵便局</FloodAssumptionArea>
        <AttainmentTime
          >○時間後(○月○日○時頃)</AttainmentTime>
        <AssumptionFlood
          >0.5～1.0m未満</AssumptionFlood>
        <AttainmentDeepestTime
          >○時間後(○月○日○時頃)</AttainmentDeepestTime>
       </FloodAssumptionPart>


<MeteorologicalInfos type="水位情報">
 <TimeSeriesInfo>
  <TimeDefines>
   <TimeDefine timeId="1">
    <DateTime>2010-02-27T09:00:00+09:00</DateTime>
    <Duration>PT1H</Duration>
    <Name>現況</Name>
   </TimeDefine>
   <TimeDefine timeId="3">
    <DateTime>2010-02-27T11:00:00+09:00</DateTime>
    <Duration>PT1H</Duration>
    <Name>２時間後</Name>
   </TimeDefine>
   <TimeDefine timeId="2">
    <DateTime>2010-02-27T10:00:00+09:00</DateTime>
    <Duration>PT1H</Duration>
    <Name>１時間後</Name>
   </TimeDefine>
   <TimeDefine timeId="4">
    <DateTime>2010-02-27T12:00:00+09:00</DateTime>
    <Duration>PT1H</Duration>
    <Name>３時間後</Name>
   </TimeDefine>
  </TimeDefines>
  <Item>
   <Kind>
    <Property>
     <Type>水位</Type>
     <WaterLevelPart>                                    A
      <jmx_eb:WaterLevel type="水位" unit="m" refID="1"
       level="2" condition="正常"
       difference="上昇">143.00</jmx_eb:WaterLevel>
      <jmx_eb:WaterLevel type="水位" unit="m" refID="2"
       level="3" condition="正常
       ">144.80</jmx_eb:WaterLevel>
      <jmx_eb:WaterLevel type="水位" unit="m" refID="3"
       level="4" condition="正常
       ">145.00</jmx_eb:WaterLevel>
      <jmx_eb:WaterLevel type="水位" unit="m" refID="4"
       level="3" condition="正常
       ">144.80</jmx_eb:WaterLevel>
     </WaterLevelPart>
    </Property>
   </Kind>
   <Station>
    <Name>○○水位観測所</Name>
    <Code type="水位観測所">00000</Code>
    <Location>○○市○○</Location>
   </Station>
  </Item>
```

第 25 図　指定河川洪水予報のバージョンアップにおける変更案 2
辞書の変更を伴う方法.

二点目は，独自の拡張を行っていることに対する留意事項である.「以上」や「未満」の表現は異常天候早期警戒情報を参考にできること，○時間後というのは W3C XML Schema のビルトインデータ型における datetime 型や duration 型で示すべき情報であるなど，既に同様の内容の表記を行っている情報がある場合は，共通の構造や基底型を使うべきとの意見である.

上記のような議論を経て，指定河川洪水予報の変更の内容が決められ，部会においても運用指針の「メジャーバージョンアップ 1（名前空間を継続）」で対応することが承認された．この結果，共通辞書（基本要素）と個別辞書（気象分野）のバージョンは「1.1」とし，これに併せて共通辞書（ヘッダ部）のとりうる値の追加も行われ，こちらのバージョンは「1.0c」とした.

11 その他の議論*

11.1 xsi:nil の取り扱い

“xsi:nil="true"”の取り扱いにおいては実装間の差があり，同記述がある要素の属性値が取得できる実装とできない実装がある．W3C XML Schema の仕様 [7] の Primer（入門書）編 2.9 項によると「この nil の仕組みは要素値のみに適用され，属性値には適用されない．“xsi:nil="true"”の要素は要素内容を持つことができず，属性値は持つことができる.」（原文は英語）となっている．しかしながら，.NET Framework 2.0 では「XML ドキュメントをオブジェクトに逆シリアル化するとき : xsi:nil="true" と指定された XML 要素が存在すると，XmlSerializer クラスでは，対応するオブジェクトに null 参照が代入され，他のすべての属性は無視されます.」[14] となっており，また動作試験によっても取得できないことが確認できている．その他，JDK1.6[15] 付属 xjc[16] でも同様に属性値が取得できず，逆に DB2[17] ではできることを動作確認した．このように，仕様では取れることが正しいにもかかわらず，一般的なソフトウェアにおいて実装の動作が異なることが判明したことから，xsi:nil を利用する要素においては他の属性をとらないこととしている．

11.2 厳密な辞書

気象庁 XML としてまとめた辞書は，構造と XML スキーマの共通化を目的の一つとしていることから，個別の情報種別単位としてみると，自由度の幅が広すぎて業務アプリケーションの設計・実装上は不要となる部分も多い．このため，共通化した「ゆるい辞書」ではなく，個々の情報種別に応じて，いわば共通化した辞書の個別運用を示す「厳密な辞書」の作成について検討を行った．この「厳密な辞書」を公開する利点については次のとおりとなる．

・利用者から見ると各情報種別の運用が一目で判別でき，適切な業務アプリケーションの開発に役立つ．
・作成者から見ると電文の設計図であり，作成するアプリケーションが適切に作成できるかどうかの確認ツールとなる．

なお，利用者に提供するのは辞書ファイルのみとし，XML スキーマについては提供しないことを想定していた．これは XML スキーマが XML インスタンスに一対一対応する原則に対して，「厳密な辞書」に対応する XML スキーマを提供することは，XML インスタンスに対応する XML スキーマが複数存在することとなり，原則が崩れて混乱することを回避するためである．

このように「厳密な辞書」の作成・提供を前提に検討を進めていたところであるが，以下の問題があった．

・「厳密な辞書」の有効的な記法の策定が困難．
「厳密な辞書」は「ゆるい辞書」の部分集合となるように記述する必要があるため，それを確認しながら辞書の作成を行うこととなるが，そのために有効なエクセルシートの記法等について何種類か試作したが，現実的に利用可能なものが作成できなかった．
・相互の辞書のバージョン管理が困難．
「厳密な辞書」の改定に当たっては必ず「ゆるい辞書」の最新バージョンに対して部分集合となるように策定する必要があるが，マスターとなる「ゆるい辞書」のバージョンが上がった場合に，旧バージョンから新バージョンへの置き換えを適切に行うための手法が検討できなかった．
・辞書間，スキーマ間の差分検出が困難．
「厳密な辞書」と「ゆるい辞書」の間で，適切に部分集合となっていることを確認する作業が必要となる．これについては辞書の記法により確認する方法もあるが，同様に XML スキーマ間でも部分集合となっているかを確認できることが望ましい．そのための一つの方法としては XML スキーマ間で差分を検出する方法の開発となる．また，XML スキーマ間での差分検出が可能となれば，辞書の記法に拠らず最終確認を XML スキーマ間で行うという手法もとれる．このため，XML ス

* 第 11 章　杉山，　第 11.4 節　竹田・杉山

キーマ間での差分検出の方法を検討してみたが解決に至らなかった．これは XML スキーマ同士，若しくはオブジェクト化した際の Document オブジェクト同士として，比較する手法が検討できなかったからである．

このことから，「厳密な辞書」の作成を断念し，気象庁 XML における個別情報種別における運用については解説資料の形にて詳細な仕様・運用を提示している．もし，上記の差分の検出方法が可能となれば，「厳密な辞書」による管理手法について改めて検討しても良い．

11.3 タグ付けの分解能

内容部においては各情報の特徴を生かすために，全体的な方針・理念の整合を必ずしも優先していない．このため，各情報や要素のタグ付けの仕方に粒度的な違いが見られる．例えば時刻的な概念の日本語において，開始時刻と終了時刻を別の要素として規定するか，それとも時間情報として一つの要素に「○○から××まで」とまとめてしまうかの差などである．これら粒度的な分解能については，情報を直接検討している担当者によっても認識が異なることから，一貫性をとるのが難しいところであるが，基本的には次のように考える必要がある．

- プリミティブ型，準プリミティブ型として機械的に値が取得できるのであれば，粒度は細かくしても良い．
- そうでない場合（日本語記述等）は利用の仕方によるが，日本語では用途が表示に限定されるので，余り粒度を細かくしない．

XML 設計全体にいえることであるが，構造を検討するにあたっては，必ず利用スタイルを意識し，作成者側の視点よりも利用者側視点で考える必要がある．また，その際にも「自分がレイアウトするのに必要な情報」といった個人的な必要性ではなく，重複を省いて単純化することを意識する．特にデータ構造においてはデータベース設計技法における正規化の考え方も重要となる．

11.4 地域情報の一意性

気象庁 XML 仕様ドラフト (Ver0.1) の作成時は，気象警報・注意報，緊急地震速報，津波警報を中心とした検討であったため，ヘッダ部の「Warning 要素」（現在の Information 要素）において，Kind 要素に対応する警報事項としての「何が」と，Area 要素に対応する対象地域としての「どこに」が明確となっていた．しかしながら，その後ヘッダ部については仕様にあるとおり「防災上重要な警報等においては，有効期間（いつ），警報事項の種別（何が），対象地域（どこに）を入手することが可能」となり，警報等の防災上重要な情報ではない場合はヘッダ部への情報の記述が必須ではなくなった．そこで対象となる地域情報を明確化させる手法として 3 通りの提案がされた．

案 1　ヘッダ部に電文が対象とする地域を示す TargetArea 要素を加える．

案 2　内容部も含めて地域情報が必ず一意となるような形式にて記載する．

案 3　編集官署名等も踏まえたオフライン的情報も含めて利用者側が総合的に判断する（現状のまま）．

案 1 については，単純に不要であるという意見が多く出た一方で，震源要素など地域情報の粒度判断が困難かつ意味をなさないという意見もあった．また，案 2 は旧形式府県天気予報（独自形式）において利用している「広島県 / 府中市」や「千葉県 / 北東部」のような「/」文字を分割記号（デリミタ）とした形式を用いることにより，地域情報として必ず一意に定まるような形式を採用することとして，内容部だけでも全国から地域が一意に定まることを目指したものである．しかしながら，これも冗長的となり不要であるという意見が出た上で，更に XML として要素を直接読み取れるメリットを使わず分割記号により解釈するという理念の逆行性への批判，純粋に作成処理が複雑になるといったデメリットが意見としてかなり多く出た．

これらの議論により，結果として案 3 のとおり，観測情報等の防災上重要ではない一部の情報については，管理部の編集官署名等の情報をオフライン情報として判断した上で，地域情報を判断することとなっている．XML コンソーシアムから，オフラインでの判断が有ること自体は否定されな

いとのコメントがあることからも，決して手法として不適切なわけではないが，この方法についてはXML的ではないこともあり，各電文同志の調整が付くのであれば，将来的に整理・検討がされても良い部分と思われる．

11.5 システム障害時の対応（運用指針4章）

(1) 気象庁XMLの作成障害時の運用について

通常時に気象庁XMLの各電文を作成しているシステムが障害となった際，当該システムによるバックアップ手段で対応できない場合は，他のシステム・手順により気象庁XML電文の作成を行うこととしている．このため，通常時の気象庁XMLと比べて簡略的な形式として発信する．

このような運用を定めるにあたっては3点注意すべき点がある．

一つ目は部外に対して通常時には取得できることを説明している要素が，簡略的形式では存在しない・出現しない場合である．これは主として量的・時間的な見積りをするために必要な値が入手できないこととなり，防災業務等において大きな影響を与えるのみならず，値があることを信用して作成したシステムにおいては，動作不良の原因となりかねない．このため，まずは運用指針において全体方針として簡略形式の発表の可能性を示唆しており，各個別の電文においては実際にどのような簡略化を行うかを詳細に説明することとしている．なお，そもそも省略してよい値であれば必然性も少ないし値があることを信用できないという観点もあることから，利用者における利用度を下げる結果となる懸念もある．

二つ目としてXML形式は機械可読形式であり，事実上のバイナリーデータに近いものであるという認識を持つ必要がある部分である．つまり，XMLはテキストエディターでも編集できることから，安易に書き換えたり作成したりしそうになるが，処理は機械的に実施されることからわずかな手違いが利用者システムの誤動作へとつながり，発信できない被害から多数の利用者システムの誤動作という二次被害を生みかねないことである．このため，XMLの代替作成機能は必ず各電文仕様に従って作成されるように制限や支援機能を付ける必要がある．例えば市町村単位でコード値を入れるところに，細分区単位のコード値が入れられないようにするような機能などである．

最後にこのような簡略形式の作成・発信という代替運用についてはめったに行わないことから作業手順上の問題を起こしやすいということである．防災情報の作成システムとしては必ず地域冗長性も含めてシステム構成を行っていることから，代替発信を行うような事態の想定はソフトウェア不具合程度である．このことからも，代替運用は可能な限り実施しないように，ソフトウェアの動作確認・慣熟運用，ソフトウェアバージョンアップ時の版管理・運用については慎重に実施すべきと考えている．

(2) 利用者システム・伝送システム等の障害時の運用について

利用者システムや伝送システム等の障害時，つまり気象庁側で各電文の作成は問題なくできているが，利用者側にオンラインで配信できない場合，他のオンライン手段をバックアップ手段として利用するか，FAX等非オンライン機能によることになる．このような場合，従前のかな漢字電文では，例えば地火山情報ではコード行とその内容を示すかな漢字本文の両方が付いていたため，障害時等でも電文をそのまま印刷するだけで人間可読形式な情報提供ができていた．また，このことは，平時でも利用者におけるコード行の解釈・処理が適切かどうかの確認を，同じ電文の本文部分で行えていたことを意味する．一方で，XML技術ではスタイルシート（XSLT）を利用することによりXMLから単純なテキスト文書を作成することは平易である．

これらの事情から，全内容をそのまま出力する単純なスタイルシートを作成し，庁内及び利用者に配布することとした．仕様は以下のとおりである．

- ある事項・要素に対してコードと日本語と両方あるならばコードは不要とする．
- 情報量としてXMLに含まれている内容が全て含まれるようにする．これは取扱いが分からない値を作らないように網羅するためであ

る．
・冗長的な修飾等は不要とし，着実な動作を目指す．改行・空白 1 文字等を利用して要素・内容の区切りを行うが，インデントや列位置を併せる必要はない．また HTML の場合はレイアウト情報がなくなっても理解できるようにする．

12 XML 関連ツール*

12.1 XMLDic2Schema

(1) ツールによる管理手順

XMLDic2Schema ツールは，XML 要素辞書を XML スキーマに変換するツールとして作成した．これは，気象庁の各情報担当職員は，その情報をどのように扱うべきか等の検討にあたってはエキスパートであっても，必ずしも IT のエキスパートではなく，今後何年もの XML スキーマの管理を考えると，その時々の担当者に XML スキーマの記述方法を覚えて対応してもらうよりは，管理しやすいエクセル形式での辞書を XML スキーマに変換するツールを作成して担当者に配布した方が，管理上適切であると判断したからである．また，エクセル形式からツールによる作成という手段に基づいた記法の制限を掛けることにより，XML エキスパートによる難解な XML スキーマの作成等を抑止することもでき，今後の運営管理上も平易になるといった利点もある．

このため，電文担当者においてはエクセル形式で記法に従い辞書を作成・管理し，それらはツールにより XML スキーマに変換するものとして，担当者における管理や IT 知識の軽減を目指すとともに，本質的な辞書の構造等の検討・理解に力を注いでもらえるようにした．

(2) ツールの開発

ツールの開発にあたっては，XML コンソーシアムから辞書の書き方の一例を提示していただいたので，これを参考にした上で，XML スキーマ生成のためのプログラム上のアルゴリズムを検討し，これと照らし合わせて辞書の書き方を作成した．また，辞書においてネスティングが多段階となると記述が困難となり，また XML スキーマの自動作成上も利点がないこと，型の再利用・共通化を目指していたこと，辞書の可読性も重視したことから，出力される XML スキーマはベネチアン・ブラインド型（第 4.2 節参照）となるようにした．参考までに開発環境・工数は第 13 表のとおり．

なお，本ツールについては職務上作成したソフトウェアでもあることから著作権等の権利は気象庁に帰属する．その他の利用条件等については明確としておらず，庁外機関・団体等への提供は禁止している．

(3) 開発上の問題点

ツールの開発途中で大きく二つの問題点が発生した．

一つはとりうる値の列記に当たり，辞書上は値を全て見せたいが XML スキーマ上は省略したい場合があることと，そもそもとりうる値に“*”を許容すると値の妥当性確認にならないという問題，もう一つは任意の要素を許容する W3C XML schema の any 要素において，出現する要素の属する名前空間指定（namespace 属性）を「任意」（“##any”）としてそのまま出力するとこれも妥当性確認にならないという問題である（第 4.3 節 (4) 項参照）．

いずれの問題においても，電文担当者等の担当者が動作確認をする際には厳密に妥当性確認をし

第 13 表　XMLDic2Schema ツール開発環境と工数

XMLDic2Schema α1.0版の開発
開発環境
・JDK1.4.2
・Eclipse3.0
・Jakarta poi 3.1
・Apache Xalan-J 2.7.1 (serializer.jar)
工数・完成状況
・設計2人日　作成5人日　試験2.5人日
・コード：2000行弱
・スタンドアローンのjarファイルとしてリリース

* 第 12 章　第 12.1 節　杉山，第 12.2 節　平原，第 12.3 節～第 12.4 節　長谷川，第 12.5 節　竹田

たいが，部外に対して仕様提示する場合には想定した記法に従って出力したいという相反する問題であることから，ツールにスイッチを作り，妥当性確認のために厳しく出力するモードと，部外向けに出力するモードとを切り換えられるようにした．このツール仕様により，部外向けとして想定した記法については，結果として第 5.2 節のとおりに XML スキーマを出力することになる．

なお，このスイッチの切替えにより作成される XML スキーマが異なることから，生成した XML スキーマの利用については確認用，部外提供用等の適切な管理が必要となる．このためにも XML スキーマに出力時のスイッチパラメーター等の出力を行う機能も付けている．

(4) 辞書の書き方

このように，XMLDic2Schema ツールに併せて辞書の書き方は策定された．書き方（辞書記述要領の抜粋版）を付録する．

12.2 防災情報 XML ビューア

気象庁 XML の実用性を検証するため，XSLT を使った防災情報 XML ビューアを作成した．まずは紫外線観測データを対象に，管理部，ヘッダ部，内容部を表形式で表示するものを作り，次にグラフ表示を作った．特別なソフトをインストールする必要がなく，インターネットエクスプローラー等のブラウザさえあれば，視覚的に説得力のある表示が可能であることが検証できた．また，データベースのように自分の知りたい情報を抽出したり，ソートが可能であることも検証できた（第 26 図）．

12.3 XML 出現タグ縮退表示ツール

気象庁 XML 各電文の構造の検討やサンプル電文作成の作業を支援するため，出現要素の構造のみを解析･保持して，要素値は，出現順を問わず，構造上の相当位置に一括して格納・表示するという加工ツールを，Web で利用する CGI の形で用意した（第 27 図）．これは，以下の場面で有用であった．

- 最初に構造を検討する段階では，検証用のスキーマも存在していない．一方，気象庁 XML の実際のインスタンスは観測値などの表現を中心に，同じ構造が繰り返し現れ，全体として大きな量になることが多い．スキーマもない状況で，長大になりがちな実際に近い電文を検討・作成する作業において，構造の概観や簡便なチェックに用いることができた．
- サンプル電文を作成する段階では，電文を生成する処理系は存在していない．また，大量のサンプル電文を人手による作業で作成するにあたっては，必ずしも全ての作業環境にスキーマ検証の処理系を用意することができない．こうした状況において，手軽に利用できるチェックツールとして機能した．

Body部

紫外線観測データ

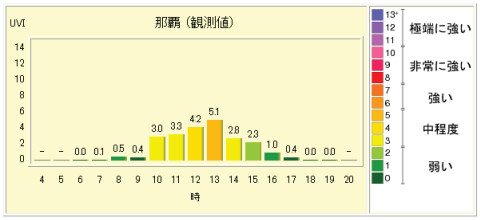
気象庁地球環境・海洋部　2008-04-08T20:26:00+09:00発表

最新時刻:2008-04-08T20:00:00+09:00

観測地点[市町村/WMO/WOUDC]	項　目	04時	05時	06時	07時	08時	09時	10時	11時	12時	13時	14時	15時	16時	17時	18時	19時	20時
那覇 [47201/ 47936/ 190/]	太陽天頂角[1]	-	-	93.7	80.4	66.9	53.5	40.6	28.8	20.3	20.1	28.4	40.1	53.0	66.4	79.8	93.2	-
	UVインデックス[2]	-	-	0.0	0.1	0.5	0.4	3.0	3.3	4.2	5.1	2.8	2.3	1.0	0.4	0.0	0.0	-

0	1	2	3	4	5	6	7	8	9	10	11	12	13+
弱い			中程度			強い		非常に強い			極端に強い		

表形式表示（那覇抽出）

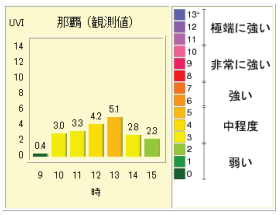


グラフ表示（那覇抽出）

観測地点[市町村/WMO/WOUDC]	項　目	09時	10時	11時	12時	13時	14時	15時
那覇 [47201/ 47936/ 190/]	太陽天頂角[1]	53.5	40.6	28.8	20.3	20.1	28.4	40.1
	UVインデックス[2]	0.4	3.0	3.3	4.2	5.1	2.8	2.3

0	1	2	3	4	5	6	7	8	9	10	11	12	13+
弱い			中程度			強い		非常に強い			極端に強い		

表形式表示（時刻抽出）

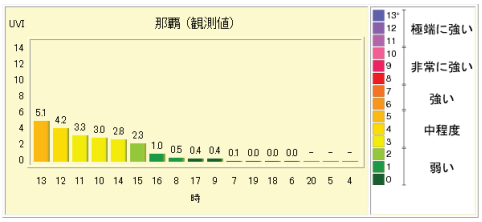


グラフ表示（時刻抽出）

観測地点[市町村/WMO/WOUDC]	項　目	1位	2位	3位	4位	5位	6位	7位	8位	9位	10位	11位	12位	13位	14位	15位	16位	17位
那覇 [47201/ 47936/ 190/]	太陽天頂角[1]	20.1	20.3	28.8	40.6	28.4	40.1	53.0	66.9	66.4	53.5	80.4	93.2	79.8	93.7	-	-	-
	UVインデックス[2]	5.1 (13時)	4.2 (12時)	3.3 (11時)	3.0 (10時)	2.8 (14時)	2.3 (15時)	1.0 (16時)	0.5 (08時)	0.4 (17時)	0.4 (09時)	0.1 (07時)	0.0 (19時)	0.0 (18時)	0.0 (06時)	-	-	-

0	1	2	3	4	5	6	7	8	9	10	11	12	13+
弱い			中程度			強い		非常に強い			極端に強い		

表形式表示（ソート）

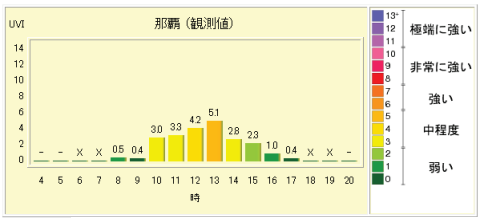


グラフ表示（ソート）

観測地点[市町村/WMO/WOUDC]	項　目	04時	05時	06時	07時	08時	09時	10時	11時	12時	13時	14時	15時	16時	17時	18時	19時	20時
那覇 [47201/ 47936/ 190/]	太陽天頂角[1]	-	-	93.7	80.4	66.9	53.5	40.6	28.8	20.3	20.1	28.4	40.1	53.0	66.4	79.8	93.2	-
	UVインデックス[2]	-	-	XXX	XXX	0.5	0.4	3.0	3.3	4.2	5.1	2.8	2.3	1.0	0.4	XXX	XXX	-

0	1	2	3	4	5	6	7	8	9	10	11	12	13+
弱い			中程度			強い		非常に強い			極端に強い		

表形式表示（最低値カット）

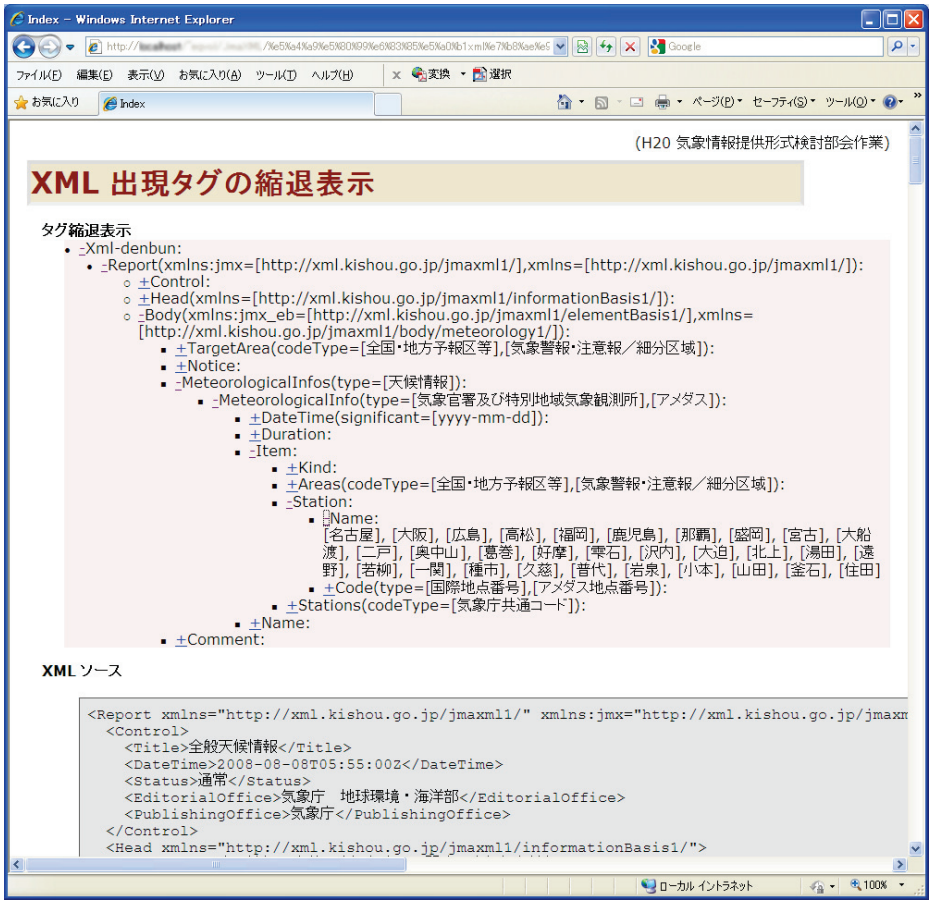


グラフ表示（最低値カット）

地　域	項目	05/25	05/27	05/28
東京地方[4410]	天気	くもり	くもり時々雨	くもり時々晴れ
	天気予報用テロップ番号	200	203	201
	風	東の風海上では北東強く	南西の風後北東の風非常に強く多摩西部では後南西の風非常に強く	西の風後北東の風非常に強く
	波	3メートル	1メートル	0．5メートル
伊豆諸島北部[4420]	天気	くもり	くもり時々晴れ	晴れ一時雨
	天気予報用テロップ番号	200	201	112
	風	東の風新島では後北東非常に強く	南西の風はじめ北東強く	南西の風やや強く後南東の風非常に強く
	波	5メートル	5メートル後3メートル新島では8メートル後5メートル	2．5メートル 新島では 4メートル
伊豆諸島南部[4430]	天気	くもり	晴れ一時雨	晴れ後雨
	天気予報用テロップ番号	200	102	114
	風	東の風	南西の風八丈島では後南西の風非常に強く	南西の風やや強く後北の風
	波	3メートル	5メートル後3メートル八丈島では5メートル後8メートル	2．5メートル 八丈島では 3メート後4メートル
小笠原諸島[4440]	天気	くもり後晴れ	晴れ時々くもり	くもり後雨
	天気予報用テロップ番号	211	101	214
	風	風弱く	南西の風強く後おさまり	南西の風やや強く後北の風
	波	3メートル うねりを伴う	5メートル後3メートル うねりを伴う	2．5メートル うねりを伴う

府県天気予報　区域予報

第 26 図　防災情報 XML ビューアの表示例



第 27 図　XML 出現タグ縮退表示ツールの表示画面

12.4　XSLT サンプル

(1) XSLT サンプルについて

平成 24 年 3 月から提供している「全内容出力スタイルシート」（第 11.5 節（2）項参照）の提供に先立ち，気象庁 XML サンプル電文提供開始後の早い時期から，利用の一例を示す目的で「従来のかな漢字電文に近い表現を再現する」等の XSLT サンプルを一部の電文について用意した．このサンプルの用意にあたっては，利用者の確認作業のために，当時広く実装が普及している仕様，すなわち XSLT 1.0 で用意することが重要と考えた．このため，EXSLT 等の拡張も利用していない．

(2) 実装上の問題

「拡張なしの XSLT 1.0 で従来のかな漢字電文を再現する」という実装を用意するにあたっては，いくつかの困難に直面した．その一つに，日付計算に係る機能が XSLT 1.0 に欠如していたことが挙げられる．例えば「地方 1 か月予報」のかな漢字電文には，以下のような部分がある．

…
＜予報の対象期間＞
1 か月：3 月 8 日（土）～4 月 7 日（月）
…

XML 電文においては，これに相当する情報を以下のように示している．

…
<MeteorologicalInfo>
<DateTime significant="yyyy-mm-dd">2008-03-08T00:00:00+09:00</DateTime>
<Duration>P1M</Duration>
…

「地方 1 か月予報」は当該地方の向こう 1 か月間の予報を伝えるものであって，「予報期間の終わりが何曜日に当たるか」を伝えることはこの電文の目的には含まれておらず，こうした部分は利用の便を想定して付加したものに過ぎない．したがって，XML 電文は予報期間の先頭と予報期間（この場合は「向こう 1 か月」）を持てば十分で，予報期間終わりの曜日を示したい場合には加工によって表現上付加されるような利用形態が望ましい．

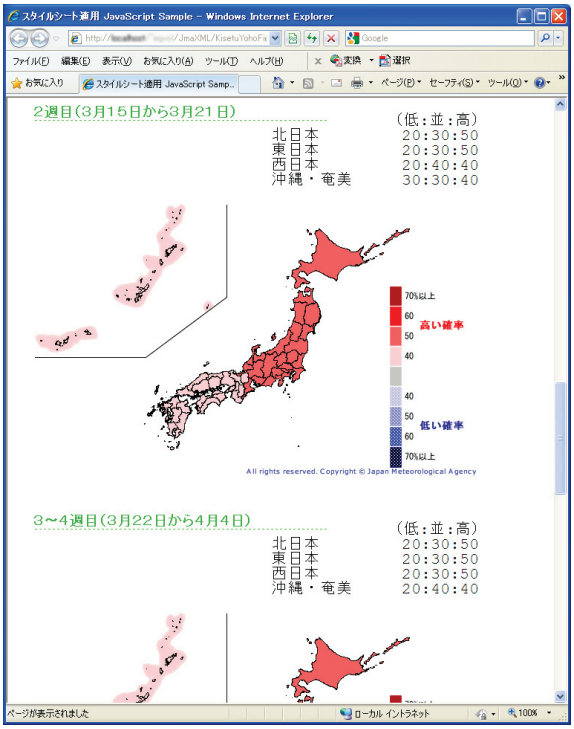
しかし，ある日付から「向こう 1 か月の終わりの日付」を求めることは簡単ではない．「1 か月」の日数は月によって異なるためである．また，期間先頭の日付によっては翌月に対応する日付がないことがあるため，例えば 3 月 30 日，3 月 31 日のそれぞれを起点とした「向こう 1 か月の終わり」はともに 4 月 30 日になる，といったケースもある．更にその「曜日」を求めるためには，特定の日付からの通算日数を扱う日付計算が必要になる．XSLT 1.0 にはこうした機能がなかったため，特に期間終わりの曜日 1 文字のためだけに，300 行程度にわたる日付計算処理を実装する必要があった．

また，こうした処理を記述するプログラミング言語として XSLT を見た場合，いくつかの極めて特徴的な性質がある．例えば「変数」と称するものは，処理の過程で値を変更できない．この特徴は，いわゆるループ変数を利用する繰り返し処理が記述できないことを意味する．手続き呼出しに伴う仮引数への値設定は許されるので，繰り返し処理は手続きの再帰呼出しを利用する以外にない．このように，気象庁において広く利用されているいくつかの手続き型言語とは大きく異なる記述が必要だったことも作業を難しくした要因になった．

(3) 新しい表現方法との比較

従来のかな漢字電文と XML 電文では，前者は利用者の便も視野に入れた完成品として考えられたもの，後者は機械処理による加工による幅広い利用を想定し情報の本質部分を抽出するように考えられたものと，その目的に違いがある．また，前者は元来，機械処理による生成を想定して作られたものではない．XSLT を用いて XML から平文を生成することは簡単ではあるが，上記の例に見られるように，従来のかな漢字電文を再現することには，単に「平文を生成する」こととは全く異なる困難があった．季節予報関係電文について従来のかな漢字電文を再現する XSLT サンプル（「1 か月」「3 か月」など数種類あるが，それらで共通のもの）は，1500 行を超える規模となった．一方，新しい応用例として第 28 図のような図による表示の生成も試みたところ，その XSLT サンプルは従来のかな漢字電文を再現するものよりもむしろ単純で，容易に実装できた．

幸い XSLT 処理系の普及に係る事情はここ数年で大きく変わり，本稿執筆時点（平成 24 年 10 月）では日付計算の可能な XSLT 処理系も広く普及して，ここに示したような例が問題となるような状況ではなくなっているようである．しかしながら，こうした問題に直面したことで従来のかな漢字電文と XML 電文で見られた大きな隔たりの存在が明らかになったことの意味は小さくない．それは，



第 28 図　気象庁 XML から色分けの図を作成する XSLT サンプルの適用例

我々が伝えるべき情報の目的と手段，またそれに応じた表現のあり方について，改めて考えるべき観点があることを示唆するものである．

12.5 妥当性検証等に利用したツール

(1) The Oracle Multi-Schema XML Validator

The Oracle Multi-Schema XML Validator（気象庁防災情報 XML フォーマットの作成を行っていた平成 20 年当時は「The Sun Multi-Schema XML Validator」の名称であった）は，執筆時点（平成 24 年 10 月）で次の URL から入手できるコマンドラインでスキーマによる妥当性検証を行うことができる検証ツールである．

http://msv.java.net/

インストールと実行も簡単で，入手した zip ファイルを任意のフォルダに展開し，コマンドプロンプト上で msv.jar に対して次のように java を実行すれば妥当性検証ができる．

java -jar msv.jar jmx.xsd XML 電文ファイル

XMLDic2Schema と合わせて利用することで，辞書による構造の共通化の作業効率が格段に向上した．

(2) AltovaXML

AltovaXML は執筆時点（平成 24 年 10 月）で次の URL から入手できるスキーマによる妥当性検証やスタイルシート（XSLT）による変換等が行える多機能な XML ツールである．

http://www.altova.com/jp/altovaxml.html

上記 URL から AltovaXML 2013 Community Edition（執筆時点）のインストーラをダウンロードし，インストールすれば，コマンドプロンプト上で次のように妥当性検証を実行できる．

AltovaXML　/validate XML 電文ファイル jmx.xsd

また，スタイルシートによる変換も可能で，コマンドプロンプト上で次のように XSLT による変換ができる．

AltovaXML -xslt1 スタイルシート
-in XML 電文ファイル -out 変換後ファイル

AltovaXML は，XML コンソーシアムから紹介されたツールで，(1) 項で紹介した MSV とともに気象庁 XML の妥当性検証に利用した．また，サンプルスタイルシートの動作確認でも利用した．

(3) XMLEditor.NET

XMLEditor.NET は執筆時点（平成 24 年 10 月）で次の URL から入手できる XML エディタで，スキーマによる妥当性検証やスタイルシート（XSLT）による変換も行える．

http://www.xmleditor.jp/

XMLDic2Schema ができるまでは，個別辞書（気象分野）の XML スキーマをテキストエディターで作成していたが，XMLEditor.NET の XML スキーマ生成機能によりサンプル電文から自動生成される XML スキーマを利用することで，効率的に XML スキーマの作成を図っていた．また，妥当性検証についても，MSV や AltovaXML のほか，XMLEditor.NET でも利用できる．更に，スタイルシートを利用して HTML へ変換する場合は，ブラウザ機能によりすぐに確認できるため，XMLEditor.NET を効果的に利用していた．

13 他仕様との関係について*

13.1 Common Alerting Protocol

Common Alerting Protocol (CAP)[18] とは，防災・治安等の公衆安全に関わる情報を伝達するための XML プロトコル（フォーマット）であり，電子商取引やウェブサービスに関する標準化団体である OASIS(Organization for the Advancement of Structured Information Standards) により仕様が承認されている．公衆安全ということで，気象分野のみならず，テロや原子力事故等も考慮した全ての公衆安全に関する情報交換を想定している．WMO では，全ての危険に対するあらゆるメディア利用した伝達を考慮すると，CAP は望ましいフォーマットであると考えており，各国の取り組み等について，2008 年，2009 年，2011 年に計 3 回ワークショップを開催している（2006 年には国際通信連合（ITU）主催のワークショップに WMO が参加という形式でも実施）．当庁では 2008 年と 2011 年に企画課より気象庁 XML の取り組みと CAP との関係を同ワークショップにて報告している．[19] [20] また，これまでの取り組みをより一層進めるため，2012 年に同じく WMO と ITU，OASIS により緊急警報政策ワークショップが開催され，当庁の緊急警報の伝達に関する取り組みについて，参事官より紹介した．[21]

CAP は気象庁 XML へ取り組み始める段階で既に仕様が策定されていたことから，当初より XML 化にあたっての仕様候補として検討していたが，次の理由により CAP 仕様は不採用とした．

- 既に防災気象情報としての詳細化が進んでおり，また高度なコード化による処理の自動化を進めている中，CAP は平文的文書に防災上のメタ情報を加えた構造となっていることから，日本国内の利用形態としては，むしろ情報量が減ってしまうことになる．
- 量的予想，時系列値の表現ができない．
- CAP の要素には公衆安全の立場から，「発生確率」「深刻度」といった要素が必須となっているが，これらは気象庁単独で決められるものではなく，値の設定には公衆安全関係者も含めた合意が必要となる．
- 防災は主として国内体制によることから，国際規格が万能とはいえない．

しかしながら，CAP は国際規格であり，国内における公衆安全における情報フォーマットとして今後利用される可能性もあるから，CAP との互換性については XML コンソーシアムの協力の下，検討を行い，以下のとおり整理された．

- 気象庁 XML は CAP の各要素に対する変換は容易である．
- 気象庁 XML から CAP への変換により情報が大きく失われることから，元の情報への参照が必要である．

個別の電文を元に確認してみても，構造や考え方が大きく変わることから，変換ルールの策定には情報の専門家による知見が必要とはなるが，変換可能性についてはおおむね問題ないと考えられる．このことから，気象庁 XML は CAP に変換可能であり，互換性の観点から将来的における CAP への対応は問題なく可能であると考えている．

なお，RSMC 東京（Regional Specialized Meteorological Center：地区特別気象センター）における CAP への取り組みとして，台風アドバイザリーの CAP 版について，検討を行っている．

13.2 その他のフォーマット仕様

気象分野における XML フォーマット関連として，公衆安全分野から ISO/TC 223（社会セキュリティ）[22] では，Tactical Situation Object（TSO）という XML フォーマットの制定を進めている．この TSO は，CAP が単純化を目指しているのに対して詳細化を進めており，どちらかといえば気象庁 XML の理念に近い．例を挙げると，コードとして原子力関係の状態を示す定義がされている一方で霧雪なども定義されており，かなり細かい定義がされている．

また，GIS に関する標準化団体である Open Geospatial Consortium（OGC）では観測値や測量値の標準化として，Observations and Measurements

* 第 13 章　杉山

(O&M) [23] を策定しており，これは ISO/DIS 19156 にもなっている．WMO では O&M を推奨することとしており [24]，航空分野から利用を計画している．

このように，気象分野に関するフォーマット仕様としては，国際的に様々な仕様が広まってきていることから，CAP と同様に，引き続き気象庁 XML との関連を整理していく必要がある．

13.3 （財）全国地域情報化推進協会（仕様 1.8 項）

（財）全国地域情報化推進協会（APPLIC）では，「地方公共団体の情報システムの抜本的改革や，地方公共団体内外の地域における多数の情報システムをオープンに連携させるための基盤の構築を推進するとともに，地方公共団体で共通利用が可能な公共アプリケーション（防災，医療，教育等）の整備等の促進」を目的として「地域情報プラットフォーム」を推進しており，これらの共通通信基盤である「プラットフォーム通信標準仕様」[25] や，業務ユニットである「防災業務アプリケーションユニット標準仕様」[26] の策定等を行っている．

政府の高度情報通信ネットワーク社会推進戦略本部（IT戦略本部）において平成 21 年 7 月 6 日に決定した i-Japan 戦略 2015[27] では，電子政府・電子自治体の推進のため「地域情報プラットフォームを活用した国及び地方の連携のための基盤システムの整備等を促進すること.」としており，地方における標準化を推奨する観点からも，気象庁 XML 仕様において「地域情報プラットフォーム標準仕様書」及び「防災業務アプリケーションユニット標準仕様」の利活用を推奨している．また，「防災業務アプリケーションユニット標準仕様」においても『気象情報については，気象庁が策定する「気象庁防災情報 XML フォーマット」を推奨する』こととしている．

2012 年現在，気象庁 XML の仕様そのものと APPLIC の各仕様との間で直接関係のあるものはない．しかしながら，通信仕様においては地方自治体や国における導入実績が多くあることから，気象庁 XML を SOAP 通信するための通信手順においては，実装が相互互換するように通信手順仕様を定めている．

13.4 全国地方公共団体コード

市町村単位の地域を示すコード表として，全国地方公共団体コード（JIS X 0402）がある．しかしながら，第 9.3 節でも示したとおり，一部行政区分や島単位の地域区分が規定されていない．また，全国地方公共団体コードの改定と当庁の業務上のコード表の改定において，日時が異なる場合もある．このため，全国地方公共団体コードをそのまま利用することは行わず，一般的な市町村については全国地方公共団体コードの値をそのまま準拠して利用し，特殊な地域情報としての利用についてはその趣旨にのっとった形で気象庁独自の拡張を行い，全体的なコード表としては気象庁管理の独自のコード表として運営している．

14 涉外活動*

気象庁 XML の策定や普及・利活用推進にあたっては，渉外活動を積極的に行ってきた．この活動履歴について，以下にまとめる．

14.1 報道発表

気象庁 XML の策定にあたっては，活動の要点において報道発表を計 4 回行った．気象庁において，業務の変更ではなく電文形式の変更という情報技術的な立場において，報道発表を行ったのは極めて異例（おそらく初めて）である．

最初の報道発表は，気象庁が XML コンソーシアムと協力して気象庁 XML の策定に当たるという内容で平成 20 年 2 月 1 日に，その後気象庁 XML のドラフト版を公開した上で，これに対する意見募集 1 回目を同年 5 月 22 日に，同意見を踏まえた上でのドラフト版最終案を公開した上で同じく意見募集 2 回目を平成 21 年 1 月 30 日に，気象庁 XML 公開版の策定について同年 5 月 15 日に，それぞれ XML コンソーシアムと共同で報

* 第 14 章　杉山

道発表を行った．これら意見の募集に際しては，1 回目の募集で 21 名，2 回目の募集で 18 名の意見の応募があり，それぞれに対して気象庁 HP 上で回答を行った．意見においては，データ構造等に対する意見のほか，運用上の質問や資料の充実の要望，データ提供に関する質問等があった．このように，情報形式に際して一般からの意見を受け付けるのも異例であるが，更にそれを踏まえて形式の修正や資料の充実を図るなど，気象庁 XML の策定や情報提供はこれまでの進め方と異なり，広く門戸を開いて実施してきた．

14.2 XML コンソーシアム

第 2.3 節に挙げているとおり，XML コンソーシアムとは平成 19 年から非公式・公式な幾多の協力関係の下，気象庁 XML の策定や利活用の推進に協力していただいた．ここでは気象庁 XML の策定そのもの以外も含めた各年度の活動関係についてまとめる．

（1）平成 19 年度～関係模索の年～

平成 19 年度当初は気象庁も電文の新形式を XML 形式とすることがまだ確定しておらず，このような状況から XML に照準を絞りこむため，XML コンソーシアムに連絡をとり，打ち合わせや勉強会を通じて，双方ともに協力関係を模索していく．2 月には共同して気象庁 XML の策定を行うことについて報道発表するまでに至り，その後，XML コンソーシアムの組織内公募に応じた会員会社により結成された第一次気象庁 XML 策定支援チームとの合同会議・勉強会を繰り広げていく．

（2）平成 20 年度～仕様策定の年～

前年度から継続している合同会議・勉強会の成果として，5 月 22 日にドラフト版の発表を行った．これ以降も，第二次気象庁 XML 策定支援チームと継続して合同会議・勉強会を実施し仕様の策定を進めていく．10 月には辞書の作成も完了したことから，改めて XML コンソーシアムの組織内公募に応じた会員会社で構成された気象庁 XML 検証協力メンバーにより，各ソフトウェアにおける動作検証を実施した．この結果も踏まえて 1 月に 2 回目の意見募集を行った．

（3）平成 21 年度～利活用推進の年～

XML コンソーシアムによる 2 回目の動作検証の結果も踏まえて，5 月 15 日に気象庁 XML の仕様公開となった．気象庁では，仕様も固まったことからサンプルや参考資料等の充実を図り，利活用を推進することとして改組も行うが，XML コンソーシアムでもこれまでの活動成果を踏まえ，各部会の活動テーマとして気象庁 XML の利活用を挙げる．XML コンソーシアムの SOA 部会，ビジネス・イノベーション研究部会，関西部会，次世代 Web 活用部会，Web サービス実証部会では気象庁 XML 関連テーマとして第 14 表のとおり活動した．

また，同年度に発足した XML 設計技術検討部会では，気象庁 XML の CAP との互換性について検討したほか，本部会にて実施された XML 設計技術勉強会では気象庁職員も参加させていただき，担当者の技量向上に貢献いただいた．

このように，XML コンソーシアムは，気象庁 XML の策定後もその利活用推進への協力を継続し，そして，平成 22 年 3 月 31 日をもって XML コンソーシアムは資産継承・後継組織設立準備団体である XML コンソーシアムコミュニティに組織を移行しつつ活動を終了した．

（4）平成 22 年度～先端 IT 活用推進コンソーシアム出発の年～

XML コンソーシアムコミュニティを経て，XML コンソーシアムはその軸足を先端 IT 技術に変えて，先端 IT 活用推進コンソーシアム（AITC）へと移り変わった．気象庁でも継続して XML 関係の助言を受けられることを期待しつつ，新たな防災情報の利活用技術についての検討を続ける．

14.3 ソフトウェアジャパン 2010

一般社団法人情報処理学会は IT の実務家のためのシンポジウムとして 2004 年度からシンポジウム「ソフトウェアジャパン」を開催しており，2010 年の IT フォーラムセッションでは，XML

第 14 表　XML コンソーシアムの各部会研究概要

部会名	タイトル	解説
SOA部会／ビジネス・イノベーション研究部会	防災情報XMLの利活用モデル	ゴール指向分析手法（i*法）を活用したサービス・モデリングと実装設計
関西部会	気象庁防災情報XMLを活用した住民情報サービス実証実験	上記部会によるサービスモデルの実装
次世代Web活用部会	雨量情報のセマンティック処理	雨の言い回しに関して地方毎に異なる雨量を対応させる方法としてセマンティックWeb技術を利用した実装の試み
Webサービス実証部会	気象庁防災情報XMLを用いた実証実験	気象庁防災情報XML配信サーバをクラウドを使って実装，気象庁防災情報XMLをGoogle App Engineで大量配信する一手法，大規模災害発生時における安否確認をどう行うか：防災ステーション編・Android編

コンソーシアムが「気象庁防災情報 XML を使った実証実験～災害から住民一人ひとりの命を守るために～」と題して，これまで気象庁 XML に関して研究・実装してきた各種成果を発表した．[28]

14.4　出版物

これまでに気象庁 XML 関連では 2 件の主筆原稿を掲載しているので，以下に紹介する．

気象庁総務部企画課（2009）：気象庁の情報システムとデータ標準化の取り組み．行政 & 情報システム，第 45 巻，第 4 号，pp.82-85.

長田泰典（2011）：防災気象情報を一人一人に届けるために－気象庁防災情報 XML フォーマットの策定－．デジタルプラクティス，Vol.2, No.1, pp.4-11.

14.5　気象庁 XML の HP

気象庁 XML の情報提供においては専用の HP を立ち上げた（http://xml.kishou.go.jp）．これは情報の内容ではなくフォーマットそのものを説明するページとしては異例であるが，これも XML コンソーシアムより提案された「XML を使うのであれば情報はオープンに」という考え方の実施でもある．基本的に気象庁 XML の電文そのものや電文の入手方法以外の全ての情報はこのページにまとめてある．

気象庁 XML の HP にはもう一つ目的がある．気象庁 XML の XML スキーマには，名前空間による URL の示唆と，XML スキーマ内に記載されている XML スキーマの入手先情報が存在する．実際にこれら URL をインターネットに接続されている端末のブラウザ上でアクセスしてみると分かるが，これら URL のアクセスにより気象庁 XML の情報及び XML スキーマを取得できるようにするためにも，専用のページを立ち上げている．

15　まとめ*

15.1　反省点

気象庁 XML の策定にあたって，適切に検討・調整しておくべきであったと考えている反省点を以下に挙げる．これらについては，大規模な改定の際に考慮していただきたい．

(1)“jmx_mete:MeteorologicalInfo”要素の子要素“name”要素値

個別辞書となることから，その値の運用については各電文検討担当者に任せていたが，“jmx_mete:MeteorologicalInfo”の要素の子要素

* 第 15 章　杉山

“name”要素値については，気象予報関係の電文だけでもその値の意味するところが微妙に異なっており，統一感のない値となってしまっている．

(2) 一意キーの導入

当初から一意キーの導入については検討していたが，現状の電文と同じく複数の情報要素を組み合わせることにより対応することで良いと考えており，また，地域冗長化しているシステムにおいては，地域間での疎結合性等から一意キーの連携を考慮しづらい点もあり，この点も含めて情報内容への一意キーの導入は行わなかった．しかしながら，その後検討を進めていくと，CAP では一意キーを導入する必要があるなど，当庁業務としては不要であっても他の情報との連携の観点からはあった方が望ましい場合もあり，互換性を考えると一意キーを導入すべきであったと考える．また，WMO にて検討されている CAP の運用において，OID（ITU-T Rec. X.660 (2008) | ISO/IEC 9834-1:2009）の利用が提案されている [29] ことから，一意キーを導入する場合は情報体系整理の観点からも適切な検討を行う必要がある．

15.2 気象庁 XML の策定と今後の管理

気象庁 XML の策定にあたって，まずは XML コンソーシアムの協力による助言，提言，指摘等により，XML 技術的に適切な仕様となったと考えている．また，庁内的にも事務局，WG，連絡担当者等各関係者が気象庁 XML をどのようにしていくのかという前向きな方向性の下，様々な提案や議論を経て各防災情報や電文の持つ構造を適切に反映した現状の仕様となった．

仕様の策定以降は仕様の管理が必要となる．管理においては，これら策定の経緯や議論を踏まえた上で，仕様の更新等を行っていく必要があることから，特定の電文内での整合性を確認するのみならず，全体的な整合性を確認する必要がある．本文書はそのための経緯をまとめたものである．管理担当者は，これらを踏まえた上で，以下の点に着眼して管理していただきたいと考えている．

・現状の各要素・属性，XML スキーマ上の型等について，新たな利用形態がこれまでの利用の趣旨から逸脱していないか．現状定義の流用が可能だからといって，趣旨の違うものを利用すると本来混ぜてはいけない機能が同居することとなり，利用者側では既存処理の方に例外処理を含めなければいけなくなるなど，動作不具合や処理の複雑化が発生することとなる．趣旨が違うのであれば，拡張・追加を検討すべきである．

・例えば複数の情報間における同一構造部分のある要素・属性を見比べた際に，追加を検討している要素･属性の値が異質とならないか．値の名詞・形容詞等の分類が違う場合，表示値・参照値として利用する場合と検索条件としての条件値として利用する場合など，利用の方向性が違う場合は共存させてはいけない．

・追加する要素・属性は不必要に冗長的なものとなっていないか．例えば作成者の判断で HP 等への掲載の際に便利なだけの値が含まれていないか（日付情報の単純な日本語表記等）．XML は機械処理を前提としていることから，オフライン仕様で処理できるところを不必要に構造化する必要がない場合も多い．

・コード表の改変によって対応できるところを要素･属性の追加により対応としていないか．本来いずれでも対応できる場合も多いが，情報構造上の無理がなければ，影響の少ないコード表の改変・オフライン仕様により対応することが望ましい．

・不必要なコード化がされていないか．XML 化により「文字」でもコード的に適切な取得ができるので，コード化は後方互換，多数のデータ群，文字表現が困難等でなければ不要である．

･必要な部分の一意性が保たれているかどうか．例えば，表示値が自由で可変であるのは当然であっても，検索条件として値を読み取って判別する部分の文字・コード値の定義が曖昧な場合に，利用者は独自の解釈をして誤動作の原因となりかねない．また，コード表はあくまでコード値と実社会における値やメタ情報を結びつけるものであり，コード値を羅列

するのものではない．あるコード値が実社会における何かと一意に結びつくようにコード表から参照できなければならない．

XML は自由度が高い故に，改変に当たってはバランス感覚が重要であると考える．フォーマットの改変は利用者への影響度が大きいことから，特にスキーマの変更を要するようなバージョンアップは慎重にすべきである．とはいえ，不必要な改変は避けつつも，新たな情報構造上必要であれば，影響が最小限となるようにした上で，思い切って追加対応等もすべきである．

16 謝辞*

気象庁 XML の策定にあたっては XML コンソーシアムに多大なる支援を頂きましたことについて，ここに感謝いたします．特に，XML コンソーシアム気象庁 XML 策定支援チーム及び検証協力メンバー，Web サービス実証部会，XML 設計技術部会，SOA 部会，ビジネス・イノベーション研究部会，次世代 Web 活用部会，及び関西部会にて協力頂いた各位には，実際のソフトウェアにおける気象庁 XML の適合性や利活用の実証実験等により，気象庁 XML の実運用・活用に道筋を示していただいたと感じています．策定支援チーム及び検証協力メンバーを第 15 表に掲載いたします．また，田原春美様，遠城秀和様には気象庁と XML コンソーシアムを強く結びつけるためご尽力を注いでいただきました．著者が個人的に相談に乗っていただいた荒本道隆様，松山憲和様，小川直人様，芦田尚人様にも感謝いたします．

また，気象情報提供形式検討部会事務局，気象庁防災情報 XML フォーマット普及・活用推進部会事務局の各事務局員，WG 構成員，連絡調整員，オブザーバ，及び関係者の皆様には，気象庁 XML の策定に当たり多大なる貢献に感謝いたします．

第 15 表　XML コンソーシアム気象庁 XML 策定支援チーム及び検証協力メンバー

第一次策定支援メンバー（会社名 50 音順、活動当時の所属会社名で記載）

アドソル日進株式会社	荒本　道隆
株式会社 NTT データ	遠城　秀和（リーダー）
日本アイ・ビー・エム株式会社	中林　紀彦
日本アイ・ビー・エム株式会社	田原　春美
日本オラクル株式会社	鈴木　俊宏
株式会社日立システムアンドサービス	村垣　委久夫
株式会社日立製作所	畑中　康一
株式会社日立製作所	永尾　雅光
PFU ソフトウェア株式会社	松山　憲和
富士ソフト株式会社	小川　直人
富士通株式会社	渡辺　進

第一次検証協力メンバー（会社名 50 音順）

日本アイ・ビー・エム株式会社	中林　紀彦
日本オラクル株式会社	鈴木　俊宏
日本電気株式会社	高橋　公一
日本マイクロソフト株式会社	田丸　健三郎
日本マイクロソフト株式会社	太田　寛
株式会社日立製作所	矢田部　英次
富士通株式会社	斉藤　一実

* 第 16 章　杉山

第 15 表 続き

第二次策定支援メンバー（会社名 50 音順、活動当時の所属会社名で記載）

アドソル日進株式会社	荒本 道隆
株式会社 NTT データ	遠城 秀和（リーダー）
株式会社時事通信社	川上 貴之
株式会社電通国際情報サービス	松村 志明
日本アイ・ビー・エム株式会社	中林 紀彦
日本アイ・ビー・エム株式会社	田原 春美
日本オラクル株式会社	鈴木 俊宏
株式会社日立システムアンドサービス	村垣 委久夫
株式会社日立製作所	畑中 康一
株式会社日立製作所	永尾 雅光
PFU ソフトウェア株式会社	松山 憲和
富士ソフト株式会社	小川 直人

第二次検証協力メンバー（会社名 50 音順）

日本アイ・ビー・エム株式会社	中林 紀彦
日本オラクル株式会社	鈴木 俊宏
日本電気株式会社	高橋 公一
日本マイクロソフト株式会社	田丸 健三郎
株式会社日立製作所	矢田部 英次
PFU ソフトウェア株式会社	竹森 昭一
富士通株式会社	斉藤 一実

参 考 文 献

[1] デジタル放送地域情報 XML 共通化研究会（2010）：デジタル放送地域情報 XML 共通 XML フォーマット．（http://www.tvcml.jp/tvcml/，2012 年参照）

[2] 屋内恭輔（2003-2005）：たのしい XML．（http://www6.airnet.ne.jp/manyo/xml/index.html，2012 年参照）

[3] ITMedia Inc.（2012）：XML 用語辞典．atmarkIT．（http://www.atmarkit.co.jp/fxml/dictionary/indexpage/xmlindex.html，2012 年参照）

[4] Refsnes Data（1999-2012）：w3schools.com．（http://www.w3schools.com/，2012 年参照）

[5] 川俣晶（2003）：XML における「ボヘミアンと貴族の階級闘争」を読み解く．atmarkIT．（http://www.atmarkit.co.jp/fxml/tanpatsu/24bohem/01.html，2012 年参照）

[6] Bray, T., J. Paoli, C.M. Sperberg-McQueen, E. Maler, and F. Yergeau（2008）：Extensible Markup Language (XML) 1.0．W3C．（http://www.w3.org/TR/xml/，2012 参照）

[7] Fallside, D.C., P. Walmsley, H.S. Thompson, D. Beech, M. Maloney, N. Mendelsohn, P.V. Biron, and A. Malhotra（2004）：XML Schema．W3C．（http://www.w3.org/2001/XMLSchema，2012 年参照）

[8] Murata, M.（2012）：RELAX NG home page．（http://www.relaxng.org/，2012 年参照）

[9] Murata, M., Y. Komachi, A. Kawamata, M. Hiyama, K. Kamimura, Y. Okui. S. Imago, Y. Yamamoto, and K. Kazama (2005) : XML Japanese Profile．W3C．（http://www.w3.org/Submission/japanese-xml/，2012 年参照）

[10] JIS（1999）：XML 日本語プロファイル 解説．JIS TR X 0015:1999（邦訳）．(http://www.y-adagio.com/public/standards/tr_xml_jpf/kaisetsu.htm, 2012 年参照）

[11] 気象庁予報部（2003）：注意報・警報の改善と天気予報の充実等について．配信資料に関する技術情報（気象編），第 153 号

[12] Bray, T., D. Hollander, A. Layman, and R. Tobin (2004) : Namespaces in XML 1.1．W3C．(http://www.w3.org/TR/xml-names11，2012 年参照）

[13] Berners-Lee, T., R. Fielding, and L. Masinter（2005）：RFC 3986: Uniform Resource Identifier (URI): Generic Syntax．IETF．(http://www.ietf.org/rfc/rfc3986.txt, 2012 年参照）

[14] Microsoft : Xsi:nil 属性バインディングのサポー

ト．MSDN．（http://msdn.microsoft.com/ja-jp/library/ybce7f69(VS.80).aspx，2012 年参照）

[15] Oracle：Java Platform. Enterprise Edition (Java EE) Technical Documentation．Oracle Technology Network．（http://download.oracle.com/javaee/，2012 年参照）

[16] Oracle（2009）：JAXB Class Generator の 使 用．Oracle XML Developer's Kit プログラマーズ・ガイド，11g リリース 2．（http://docs.oracle.com/cd/E16338_01/appdev.112/b56264/adx_j_jaxb.htm，2012 年参照）

[17] IBM：データベース (DB2)．（http://www-06.ibm.com/software/jp/data/db2/，2012 年参照）

[18] Jones, E. and A. Botterell（2005）：Common Alerting Protocol, v.1.1. OASIS．（http://www.oasis-open.org/committees/download.php/15135/emergency-CAPv1.1-Corrected_DOM.pdf，2012 年参照）

[19] Yamakoshi, Y.（2008）：XML draft format for warnings and advisories in Japan．WIS Common Alerting Protocol (CAP,X.1303) Implementation Workshop，WMO，Geneva，Switzerland.

[20] Sugiyama, Y. and N. Washitake（2011）："Japan disaster Mitigation and prevention information XML format" (JMX) is in operation!．Common Alerting Protocol (CAP) Implementation Workshop，WMO，Geneva，Switzerland.

[21] Kuma, K.（2012）：Disaster Prevention Information Provided by Japan Meteorological Agency．Emergency Alerting Policy Workshop，Environment Canada, other Canadian agencies, OASIS, WMO and ITU，Montreal，Canada.

[22] ISO：TC223 Societal security．（http://www.iso.org/iso/iso_technical_committee.html?commid=295786，accessed 2012）

[23] Cox, S.（2010）：Geographic Information: Observations and Measurements．OGC．（http://portal.opengeospatial.org/files/?artifact_id=41579，2012 年参照）

[24] WMO（2010）：DECISION FOR THE OPEN PROGRAMME AREA GROUP ON INFORMATION SYSTEMS AND SERVICES, INCLUDING THE WMO INFORMATION SYSTEM, CAPACITY-BUILDING AND THE QUALITY MANAGEMENT FRAMEWORK．CBS-Ext 2010，WMO-No.1070，pp.25.

[25] APPLIC 技術専門委員会（2012）：プラットフォーム通信標準仕様 V2.3．APPLIC．（http://www.applic.or.jp/2012/tech/，2012 年参照）

[26] APPLIC 安心安全ワーキンググループ（2012）：防災業務アプリケーションユニット標準仕様 V1.1．APPLIC．（http://www.applic.or.jp/2012/app/bousai/index.html，2012 年参照）

[27] 高度情報通信ネットワーク社会推進戦略本部（IT 戦略本部）（2009）：i-Japan 戦略 2015．（http://www.kantei.go.jp/jp/singi/it2/dai51/siryou3.pdf，2012 年 参照）

[28] 山本太基（2010）：気象庁防災情報 XML フォーマットについて．ソフトウェアジャパン 2010，一般社団法人情報処理学会，東京，日本.

[29] Dubuisson, O.（2011）：Introduction to Object Identifiers (OIDs) and their use by WMO．Common Alerting Protocol (CAP) Implementation Workshop，WMO，Geneva，Switzerland.

用 語 集

API：Application Programming Interface の 略．OS，ソフトウェア，ライブラリー，サービス等のある機能に対して，利用者・利用アプリケーションがその機能を利用するためのインターフェースを指す．API が標準化されていると，実装に関わらずプログラムが同一化・統一化でき，また洗練された API は利用者がその機能の実装構造を知らなくても利用できるので重要となる．

DOM・DOM ツリー：Document Object Model の略．W3C から勧告されている XML 文書を扱うための API．DOM は XML 構造に準じてツリー構造として取り扱えるので，このことを DOM ツリーともいう．構造に沿って自由に解析・拡張ができるが，実装においては処理速度とメモリー使用量が多めとなってしまう欠点がある．原典は次の URL を参照．http://www.w3.org/DOM/DOMTR

IDE：Integrated Development Environment の略で，統合開発環境を意味する．プロジェクト管理，バージョン管理，ビルド・デバグ補助機能などが付いている開発環境．Visual Studio や Eclipse が有名．

SAX：Simple API for XML の略．XML 文書を扱うための API で，標準化団体による規定はされていないが，事実上の標準化規格として利用されている．DOM と異なり，ツリー構造として取り扱うのではなく，イベント駆動型として取り扱う．構造に沿った自由な解析ができず，また XML の作成・拡張は全くできないが，ストリーム処理の一環となることから高速かつ省メモリーで動作する．原典は次の URL を参照．http://www.saxproject.org/

SOAP：ソフトウェア同士がメッセージ交換するためのプロトコルで，XML を利用する．W3C により標準化されている．

UNICODE：コンピュータ上の文字列を一貫した方式で符号化し取り扱うための標準規格．UNICODE の符号化方式として UTF-8 が広く使われている．

W3C：World Wide Web Consortium の略で，World Wide Web（WWW）で使用される各種技術の標準化を推進する団体．

XML：JIS による邦訳は「拡張可能なマーク付け言語」．第 3 章のほか，原典は次の URL を参照．http://www.w3.org/TR/xml/

XML インスタンス：一般的にオブジェクトの実体をインスタンスというが，XML インスタンスは XML スキーマに対して XML の情報そのものを指すときにいう．

XML スキーマ：XML の文書構造を定義する手法の 1 つで，いわば設計図に当たる言語．第 4 章を参照．

XML 電文：XML を利用した気象庁の電文形式の一つ．

XSLT：Extensible Stylesheet Language Transformation の略．W3C から勧告されている XML 文書を別の XML 文書等に変換するための言語･技術．単純な構造変換であれば，XSLT によりプログラムを書かずに XML を変換できる．原典は次の URL を参照．http://www.w3.org/TR/xslt

かな漢字電文：主として Shift_JIS のいわゆる全角（2 バイト文字）を利用した気象庁の電文形式の一つ．機械可読を目的としたコードを付記した物もある．

基底型：あるデータ（要素値や属性値など）が属するクラスのこと．プリミティブ型又はプリミティブ型を元に制限や拡張等を与えたクラスとなる．

コード電文：主として ASCII 文字のみを利用した気象庁の電文形式の一つ．フォーマットを明確にして機械可読を目的としている．

実装：ある機能を実現するために構成要素を具体化することを一般的に実装という．ソフトウェア分野では仕様・機能に対して実際に動作するプログラムを示す．

名前空間：XML において，違う体系（個々の XML スキーマ）に属する要素等が，お互いに別の体系に属していることを明確化するために用いられる区分手法．URI により定義し，接頭辞と呼ばれる簡略化した文字とのマッピング（“xmlns: 接頭辞 ="URI"”の記法）を通じて，所属が一意に定まるように各要素を修飾する．

バリデーション（Validation）：XML では DTD，各種 XML スキーマ等を用いて妥当性検証を行うことをバリデーションという．バリデーションするソフトウェアをバリデーター（Validator）という．第 4 章を参照．

平文：かな漢字電文に代表されるような，人間が自然言語として読むことを目的とした電文又は電文の一部分．機械可読を目的としたコード電文やコード行の対義語として用いる．

プリミティブ型：プログラミング言語において提供される基本的なデータ型．文字列型（string 型，token 型），整数型（integer 型），浮動小数点型（float 型）などがこれにあたる．

メタ情報：主たる情報についての情報である．具体的には主たる情報を要約したような情報を示す．

付録　辞書記述要領（抜粋版）

平成 21 年 3 月 2 日

辞書記述要領

1．項目表

接頭辞	“jmaxml”	名前空間	“uri”						
項番	親要素	子要素	属性	基底型	サイズ	出現回数	意味	とりうる値	解説
1	type.A1						…		…
2			C11	xs:string	128	1	…		…
3				*			…	“c11a”	…
4				*			…	“c11b”	…
5				*			…	RE”c11reg”	…
6				*			…	*	…
7		B11		type.A3		?	…		…
8		B12		xs:string	256	1	…		…
9				*			…	“b12a”	…
10		*		##any		1,3	…		…
11	type.A2						…		…
12			C12	xs:string		?	…	code.D2	…
13		B21		type.A1		*	…		…
14		B22		type.A4		+	…		…
15	type.A3						…		…
16		B31		xs:int		1(nil)	…		…
17		B32		xs:string		1	…	code.D1	…
18	type.A4			xs:string	*		…		…
19			C13	xs:string	128	?	…		…
20	(element)	B51		type.A2		1	…		…
21	(end)								

共通事項：

・表の初段に当該表の示すスキーマの接頭辞と名前空間を記述する．関連する表間で，同じ名前空間を示す接頭辞は常に同じとする．インスタンス名の記述がある場合はインスタンス型表記を行うことを明示している．

・属性，基底型で，他の名前空間に属するものについては，必ず接頭辞を付ける．

・型を示す場合は "type." で，コード辞書を参照する場合は，"code." を用いる．

項番：

・辞書中の位置を一意に表せるように番号を記述する．

親要素：

・辞書の項目の名前を記述する．基本的に型名となる．

・辞書中で唯一の名前とする．

例）A1，A2，A3，A4 は親要素（項番 1，11，15，18）

・特別な識別子 "(element)" により，子要素の要素名で基底型の要素をグローバルに宣言する．

例）要素名 B51 を型 A2 としてグローバルに宣言する（項番 20）

子要素：

・親要素に含まれる項目（element）の名前を記述する．
・同じ親要素内で原則，唯一な名前とする．辞書中で唯一の名前となれればより望ましい．
・子要素が他の名前空間接頭辞が付いた要素名となっている場合，他の名前空間の型に対する参照（Reference）であることを示す．
・例外的に任意の子要素も許容する場合は，子要素にアスタリスク（"*"）を記述し，許容する子要素の名前空間を基底型に記述する．（例：項番 10）
この場合で，許容する名前空間が複数ある場合は，基底型を識別子 "(namespace)" とすることにより，次行より列挙型とできる．

例）B11，B12 は A1 の子要素（項番 7，8），B21，B11 は A2 の子要素（項番 13，14），B31，B32 は A3 の子要素（項番 16，17）

属性：

・親要素の属性の名前を記述する．必ず親要素の次行からとする．
・同じ親要素内で唯一な名前とする．辞書中で唯一の名前となれればより望ましい．

例）C11 は親要素 A1 の属性（項番 2），C12 は親要素 A2 の属性（項番 12），C13 は親要素 A4 の属性（項番 19）

注 1）親要素，子要素，属性は各行に排他的に用いる．
注 2）属性は直前の親要素の属性（attribute）を表し，子要素は直前の親要素の項目（element）を表す．
注 3）親要素，子要素，属性，基底型の全てが空の行は表の最後以外には置かないものとする．

基底型：

・子要素および属性の基底型を記述する．
・子要素の基底型としては，XML Schema で定義されているビルトインデータ型または，辞書中で定義される親要素の名前もしくは他の名前空間により示される型を記述する．（例：項番 7，8，13，15，16，20）
・属性の基底型としては，XML Schema で定義されているビルトインデータ型，もしくは他の名前空間により示される型やコード名を記述する．（例：項番 2，12，19）
・子要素および属性の基底型は必ず記述する．
・親要素の基底型は通常記述しないが，子要素が無く Text ノードのみで属性しか持たない型の場合（simpleContent 型）のみ，親要素の行に基底型を記述する．（例：項番 17）
・子要素から引き続く行の親要素，子要素，属性が空欄で，基底型が "*" の場合，直前の子要素の基底型を用いた列挙を表す．（例：項番 8，9）
・属性から引き続く行の親要素，子要素，属性が空欄で，基底型が "*" の場合，直前の属性の基底型を用いた列挙を表す．（例：項番 2，3，4，5，6，）
・任意の子要素から引き続く行の親要素，子要素，属性が空欄で，基底型が "*" の場合，直前の任意の子要素が許容する名前空間の列挙を表す．
・リスト型の表記を行う場合，基底型として "xs:list（リスト型の基底型）" フォーマットで記述すものとし，リスト型の基底型としてビルトインデータ型のみを許容する．
・コード辞書を参照する場合は，コード辞書の規定となるビルトインデータ型を記述する．

サイズ：

・基底型に記載されているビルトインデータ型データのサイズを記述する．
・文字列型の場合，列長を記述する．数値型の場合，その型が規定している上下限を狭める場合に記述する．
・記述において，単独で記述する場合は上限を，それ以外は "（下限値 , 上限値）" の形式で，記述する．
・上限値が無期限，もしくは型の規定値の上限を示す場合は "*" を記述する．

出現回数：

・親要素の出現回数は記述しない．具体的な出現回数は親要素を参照する子要素中に定義される．
・子要素の出現回数は以下のように記述する．

必ず 1 回出現	1
0 回か 1 回出現	？
1 回～無限大に出現	＋
0 回～無限大に出現	＊
N 回～ M 回出現	N , M
xsi:nil により省略可能	(nil)

・属性の出現回数は以下のように記述する．

必ず 1 回出現	1
0 回か 1 回出現	？

例）子要素の「1」の例は項番 8，17，子要素の「？」の例は項番 7，子要素の「1 回～無限大」の例は項番 14，子要素の「0 回～無限大」の例は項番 13，子要素の「1 回～ 3 回」の例は項番 10．属性の「1」の例は項番 2，属性の「？」の例は項番 12，19．省略可能の例は項番 16．

意味：

・親要素，子要素，属性の意味を簡潔に記述する．

とりうる値：

・子要素，属性がとりうる値を記述する．
・子要素の基底型が他の親要素の名前となる場合は省略する．
・列挙を表す場合，実際の文字列をダブルクォートで挟んで記述する．複数列挙する場合は，次の行に連ねて記述する．（例：項番 3，4，9）
・とりうる値として正規表現による記述を行う場合，識別子 "RE" を冠した上で正規表現型をダブルクォートで挟んで記述する．この場合，列挙している他のとりうる値の行の中で，任意の値表現を除いた最後に記述する．（例：項番 5）
・とりうる値として任意の値を表現する場合，とりうる値を "*" としてダブルクォートで挟んで記述する．この場合，列挙している他のとりうる値の行の中で，最後に記述する．（例：項番 6）
・コード辞書を参照する場合，"code." をつけて記述する．（例：項番 12，17）複数のコード辞書を参照する場合も，とりうる値の列挙と同様にする．また，とりうる値としてコード辞書の参照を行う場合は，任意の値表現以外とは併用できない．

解説：

・親要素，子要素，属性を説明する文章を記述する．

注 4）子要素がアスタリスク（"*"）だけの行は，例外的に任意の子要素も許容する理由を解説に記述する．

注 5）子要素，属性，許容する名前空間の列挙を表す行は，基底型を "*" とした上で，意味，とりうる値，解説のみを記述する．

注 6）インスタンス記述のため，親要素，子要素，属性，基底型の全てが空の記述を認める．表の最後には親要素を "(end)" とした行を置くものとする．（例：項番 21）

2. XML Schema 生成の考え方

上記の項目表，コード表を元に XML Schema 生成の考え方を例示する．

前提条件：

・XML Schema の記述スタイルとしてベネチアンブラインドパターンをベースとする．
・親要素の名前は辞書中で唯一となっている．
・子要素，属性の名前は親要素内で唯一となっている．
・コードの名前は辞書中で唯一となっている．
・子要素は記述順に出現するとし，任意の順序は許容しない．
・他の名前空間に属する型，コード等については，必ず接頭辞を付ける．

生成規則：

・親要素からは，型定義（complexType）を生成する．
・親要素の名前からは，型定義の name 属性を生成する．
・親要素名として型を記載し，同一行にその基底型を表記した場合，子要素のない属性値付き要素（simpleContent）として拡張し，次行以降の属性のみを生成する．
・親要素が“(element)”の場合，name 属性を子要素とし，type 属性を基底型としてグローバルに element 要素を生成する．
・子要素からは，親要素に対応する型定義中の要素定義（element）を生成する．
・子要素の名前からは，要素定義の name 属性を生成する．
・子要素の基底型からは，要素定義の type 属性を生成する．
・子要素の出現回数からは，要素定義の minOccurs 属性（"1"，"0"）と maxOccurs 属性（"1"，"unbound"）を生成する．
・子要素の列挙からは，enumeration，そのとりうる値からは，enumeration の value 属性を生成する．
・子要素の名前に別の名前空間接頭辞が付いていた場合，基底型に記載の要素名に対して参照することとする．
・任意の子要素も許容する記述からは，要素定義に any の定義を生成する．この場合の基底型は許容する名前空間とし，列挙した場合は xs:list 型とした上で，namespace 属性を生成する．
・属性からは，親要素に対応する型定義中の属性定義（attribute）を生成する．
・属性の名前からは，属性定義の name 属性を生成する．
・属性の基底型からは，restriction の base 属性を生成する．
・属性の出現回数からは，属性定義の use 属性（"required"，"optional"）を生成する．

・属性の列挙からは，enumeration，そのとりうる値からは，enumeration の value 属性を生成する．
・いずれの場合の列挙においても，任意の値を許容する記述を含む場合，辞書に記載されている列挙型を生成せず，記述されている基底型の単純型として生成する．
・コード辞書を参照する場合，コード辞書に対する型への参照を生成せず，記述されている基底型の単純型として生成する．
・子要素として任意の子要素を許容する場合で，かつその子要素について許容する名前空間の列挙において，"##any"，もしくは "##other" を含む場合，辞書に記載されている名前空間の列挙を生成せず，記述されている "##any"，もしくは "##other" のみ許容するとして生成する．
・サイズは辞書のための情報とし，制限形式として生成しない．

上記生成規則を用いると辞書例から下記の XML Schema が生成される．

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
        xmlns:xs=http://www.w3.org/2001/XMLSchema
        xmlns:jmaxml="uri" elementFormDefault="qualified" targetNamespace=" uri ">
 <xs:complexType name="type.A1">
  <xs:sequence>
   <xs:element name="B11" minOccurs="0" maxOccurs="1" type="jmaxml:type.A3"/>
   <xs:element name="B12" minOccurs="1" maxOccurs="1" type="jmaxml:enum.type.A1.B12"/>
   <xs:any namespace="##any" minOccurs="1" maxOccurs="3" processContents="lax"/>
  </xs:sequence>
  <xs:attribute name="C11" use="required" type="jmaxml:enum.UNION.type.A1.C11.attr"/>
 </xs:complexType>
 <xs:complexType name="type.A2">
  <xs:sequence>
   <xs:element name="B21" minOccurs="0" maxOccurs="unbounded" type="jmaxml:type.A1"/>
   <xs:element name="B22" minOccurs="1" maxOccurs="unbounded" type="jmaxml:type.A4"/>
  </xs:sequence>
  <xs:attribute name="C12" use="optional" type="xs:string"/>
 </xs:complexType>
 <xs:complexType name="type.A3">
  <xs:sequence>
   <xs:element name="B31" minOccurs="1" maxOccurs="1" nillable="true" type="xs:int"/>
   <xs:element name="B32" minOccurs="1" maxOccurs="1" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="type.A4">
  <xs:simpleContent>
   <xs:extension base="xs:string">
    <xs:attribute name="C13" use="optional" type="xs:string"/>
```

```
      </xs:extension>
     </xs:simpleContent>
    </xs:complexType>
    <xs:element name="B51" type="jmaxml:type.A2"/>
    <!--  -->
    <!--Enumeration's  -->
    <!--  -->
    <xs:simpleType name="enum.type.A1.C11.attr">
     <xs:restriction base="xs:string">
      <xs:enumeration value="c11a"/>
      <xs:enumeration value="c11b"/>
     </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="enum.pattern.type.A1.C11.attr">
     <xs:restriction base="xs:string">
      <xs:pattern value="c11reg"/>
     </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="enum.ANY.type.A1.C11.attr">
     <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="enum.UNION.type.A1.C11.attr">
      <xs:union memberTypes="jmaxml:enum.type.A1.C11.attr jmaxml:enum.pattern.type.A1.C11.attr jmaxml:
enum.ANY.type.A1.C11.attr"/>
    </xs:simpleType>
    <xs:simpleType name="enum.type.A1.B12">
     <xs:restriction base="xs:string">
      <xs:enumeration value="b12a"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:schema>
```

以上により，次の XML データが許容される．

```
   <?xml version="1.0" encoding="UTF-8" ?>
   <B51 C12="code.D2 の各定義に従う " xmlns="uri/">
    <B21 C11="c11a">
     <B11>
      <B31>128</B31>
      <B32>code.D2 の各定義に従う </B32>
     </B11>
```

```
 <B12>b12a</B12>
 <any>any な要素 </any>
</B21>
<B21 C11="c11b">
 <B12>b12a</B12>
 <any>any な要素 1</any>
 <any2>any な要素 2</any2>
</B21>
<B22>b22 の値その 1</B22>
<B22 C13="c13 の属性値 ">b22 の値その 2</B22>
</B51>
```

3．その他，辞書エクセルシートの書き方

・全てのシートを辞書として解析する．解析しないシートは名称を“#”で始める．

・名前空間表を用意し，シート名を“#ns”とする．シートには各辞書で用いる名前空間のうち，W3C XML Schema 以外の名前空間について，その接頭辞とワールドワイドなスキーマ参照 URL，及びローカルスキーマ参照 URL を記述する．また，外部スキーマファイルを取り込む（xs:include）する際には，“include フラグ”列に“include”と記述する．
なお，ローカルスキーマ参照 URL 以外は記載必須とし，同 URL 省略時のローカルスキーマ作成時には，シート名をファイル名とする．

・作成されるスキーマにおいて，冒頭に出力する注釈文について，“# 注釈”シートに記載する．また名前空間別に管理される注釈文については，名前空間表に個別に記述することする．冒頭の注釈文においては，「“# 注釈”シートの記述+名前空間表の個別の記述」により出力される．

・共通項目表におけるエクセルシートのシート名を接頭辞とする．

4．参考事項

・any 要素の namespace 属性について，特別の値として次のものがとれる．なお，列挙型と併用できるのは，“##other”と“##local”のみである．

“##any”	任意の名前空間に属する要素
“##targetNamespace”	このスキーマの名前空間に属する要素
“##other”	このスキーマの名前空間以外の名前空間に属する要素
“##local”	名前空間に属さない要素（非推奨）

・別の名前空間に属する要素を作成する際は，要素名は呼び出される側の名前空間に属するようにする．そのために，次の 2 点の通りの設定を利用する．

1．別の名前空間接頭辞をつけた子要素を指定し，ref 参照要素を作成する．

2．別の名前空間の辞書において，当該子要素を親要素名“(element)”にてグローバル要素で宣言する．