

# 減災情報共有プロトコル仕様

Version1.0（案）

独立行政法人 防災科学技術研究所 独立行政法人 産業技術総合研究所

平成17年3月

## はじめに

本書は、文部科学省科学技術振興調整費・重点課題解決型研究プロジェクト「危機管理対応情報共有技術による減災対策」の研究項目「減災情報共有プラットフォームの開発」における「減災情報共有プロトコル」仕様を定めたものである。

本書は、独立行政法人防災科学技術研究所と独立行政法人産業技術総合研究所との共著である。前者の担当は、「減災情報共有プラットフォームの開発」の全般、および、「減災情報共有プロトコル」における時空間情報モデルに関する記述である。後者の担当は、後者が提案し、WFS ( Web Feature Service )<sup>3)</sup>、GML ( Geographical Markup Language )<sup>2) A1)</sup>を拡張したGeneral GeoGraphical Database ( GGGD )<sup>1)</sup> Systemの「減災情報共有プロトコル」への導入である。

## 目次

1 . 概要.....	1
2 . 目的.....	1
3 . 減災情報共有プロトコル仕様 .....	3
3 . 1 シーケンスモデル .....	3
3 . 2 電文の構造.....	5
3 . 3 リクエストとレスポンス .....	15
3 . 4 添付ファイル .....	27
4 . 制約事項等.....	32
4 . 1 利用可能なXML Schemaタグと属性 .....	32
4 . 2 Filter記述上の制約.....	32
4 . 3 Filter属性の記述要素 .....	33
4 . 4 事象とジオメトリの取り扱い .....	34
4 . 5 データ単位系について .....	35
4 . 6 時間スキーマ対応 .....	35
4 . 7 時間情報対応 .....	37
4 . 8 メッシュ情報の記述.....	39
5 . 使用例.....	45
参 考 文 献.....	52
付 録.....	53

## 1 . 概要

新潟中越地震では、阪神・淡路大震災の経験が生かされ、関連機関の迅速な初動対応を評価する声を耳にするが、孤立した自治体の状況把握、救援物資の需要と供給のバランス、道路網寸断状況と運搬手段のバランス等、被災状況の把握と情報共有に関する課題も多く指摘された。被災状況把握の基盤となる平常時の地域管理データと災害時活用の課題も見逃せない。一般には、台風や地震などの自然災害のみならず人為的災害の危険性も高まっており、災害情報の共有による減災のための戦略的・協調的対応の実現は愁眉の課題である。以上のような課題を解決するため、自治体等の災害関係機関における防災情報システムで使用でき、それらの防災情報システムが相互に必要な情報を交換して減災情報共有をするための「減災情報共有プロトコル」の開発が求められている。

文部科学省科学技術振興調整費重点課題解決型研究プロジェクト「危機管理対応情報共有技術による減災対策」では、普遍的な時間と空間における位置座標をキー（時空間キー）とした情報管理技術<sup>7)</sup>（時空間情報モデル、時空間データベース構造・KIWI+形式）を核とした「減災情報共有プラットフォームの開発」が行われることとなった。この開発において、標準的で広く認知されている通信プロトコルを用い、「減災情報共有プロトコル」の仕様を定める。

## 2 . 目的

「減災情報共有プロトコル」の目的を以下に示す。

災害対応の場に参画する自治体等、複数機関の防災情報システムが減災情報を共有するためのプロトコル仕様とする。

関係機関の創発的・戦略的・協調的な災害対応による人命・財産の被害軽減化に寄与する。

以上を踏まえ、「減災情報共有プロトコル」を以下のようなものとした。

- ・ オンデマンドで対等な通信ネットワークの構成。
- ・ 標準的で広く認知されているHTTP( Hyper Text Transfer Protocol )<sup>8)</sup>やSMTP( Simple Mail Transfer Protocol )<sup>8)</sup>、FTP(File Transfer Protocol)<sup>12)</sup>通信プロトコルの利用。  
本仕様ではSOAP(Simple Object Access Protocol)<sup>9)</sup>を使用するためTCP/IP (Transmission Control Protocol / Internet Protocol)<sup>16)</sup>ネットワークでのトランスポートプロトコルは規定しない。
- ・ スキーマ問い合わせを含め標準的な情報交換サービスを規定したWFS ( Web Feature Service )<sup>3)</sup>の利用、および、広く認知されているXML(Extensible Markup Language)<sup>4)</sup>、SOAP(Simple Object Access Protocol)<sup>6)5)</sup>、GML(Geography Markup Language)<sup>2)A1)</sup>、MIME(Multipurpose Internet Mail Extension)<sup>10)</sup>による電文の記述。
- ・ Base64 エンコード<sup>10)A7)</sup>による画像データ、GML(Geography Markup Language)<sup>2)A1)</sup>形式データ、KIWI+形式データ<sup>7)A2)</sup>、差分データ、シミュレーション結果のメッシュ形式データ等、各種ファイルの電文添付による効率と利便性の確保。

- ・別途仕様を定めるライブラリの装備による「減災情報共有プロトコル」利用の支援。
- ・GML(Geography Markup Language)<sup>2)</sup>・KIWI+形式<sup>7)A2)</sup>の変換プログラム装備による大容量空間基盤データの共用支援。

以降下記の表記を使用する。

HTTP ( Hyper Text Transfer Protocol )	HTTP
SMTP ( Simple Mail Transfer Protocol )	SMTP
FTP(File Transfer Protocol)	FTP
WFS ( Web Feature Service )	WFS
XML(Extensible Markup Language)	XML
SOAP(Simple Object Access Protocol)	SOAP
GML(Geography Markup Language)	GML
MIME(Multipurpose Internet Mail Extension)	MIME

### 3 . 減災情報共有プロトコル仕様

#### 3 . 1 シーケンスモデル

セッション開始側のクライアントである UC (User Client) 受け手側のサーバである US (User Server) のクライアント・サーバ型の形態をとり、UC によるリクエストと US によるレスポンスを繰り返すことで順次処理を行う。リクエストは動作を起動するメッセージであり、レスポンスはリクエストに対する結果の通知である。

セッションは1つの要求送信(応答送信)ごとに確立し解消する。

UC は US に対しセッションを確立後、要求メッセージを送信する(リクエスト)。US は一端セッションを解消した後にリクエストに対する処理を行う。US は処理終了後 UC に対し再びセッションを確立し応答メッセージを送信する(レスポンス)。UC は応答メッセージ受信後、セッションを解消する。

図 3.1-1 にセッションの構成、図 3.1-2 にシーケンスモデルを示す。

- ・オンデマンド(On Demand)型対等な通信ネットワーク
- ・ユーザーリクエストに応じてサービスに应答する
- ・サーバ機とクライアント機の違いがなく、すべてのコンピュータがサーバとしてもクライアントとしても機能する

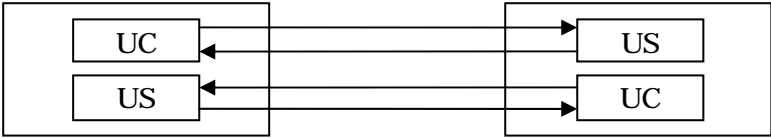


図 3.1-1 セッションの構成

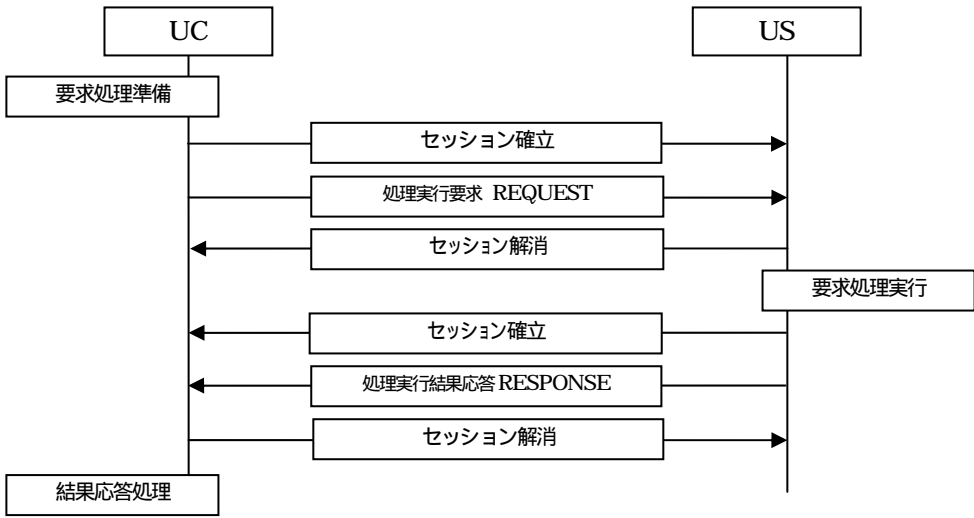


図 3.1-2 シーケンスモデル

UC: User Client  
当該プロトコルメッセージ通信を行う場合の開始側であり、リクエストメッセージを送出する端末または機能モジュールを指す。

US: User Sever  
当該プロトコルメッセージ通信を行う場合の受け手側であり、レスポンスメッセージを送出する端末または機能モジュールを指す。

### 3.2 電文の構造

電文の構造は、SOAPメッセージ<sup>4) 5) 6)</sup>の構成とし、ファイルの添付処理のためSOAP Messages with Attachments<sup>9)</sup>に準拠しインターネットメール等で利用されているMIME<sup>10)</sup>構造をとる。(図 3.2-1)(表 3.2-2)

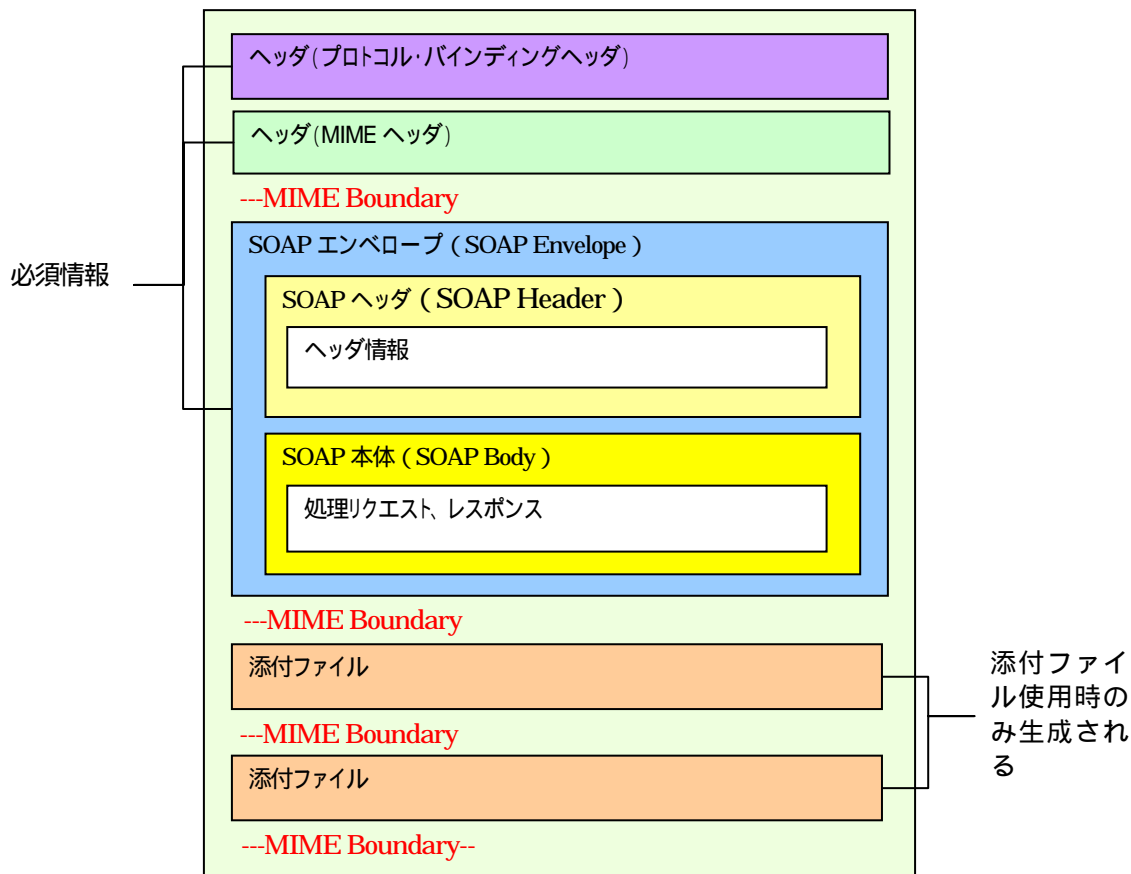


図 3.2-1 電文の構造

電文は大きくわけて、データ送信プロトコルに依存した「プロトコル・バインディングヘッダ」、MIME<sup>10)</sup>構造を定義する「MIME<sup>10)</sup>ヘッダ」、処理メッセージを記述する「SOAPエンベロープ (SOAP Envelope)」、及び添付ファイルから構成されている。

「SOAP エンベロープ」は、さらに「SOAP ヘッダ (SOAP Header)」と「SOAP 本体 (SOAP Body)」から構成される。

MIME Boundary で仕切られた各部を「パート(part)」と呼び、添付ファイルが存在する場合に仕切られる。添付ファイルが存在しない場合は、プロトコル・バインディングヘッダ + MIME ヘッダ + SOAP エンベロープ (SOAP Envelope) のみの構成になる。

本仕様においては、主として「処理メッセージ」、「ファイル添付仕様」について述べる。



表 3.2-2 電文の構造概要

電文各部		概 要
ヘッダ		プロトコル・バインディングヘッダ。 SOAP の実装を行うデータ通信プロトコルに依存したヘッダ。
ヘッダ		MIME ヘッダ。添付ファイルの有無、MIME 構造の BoundaryString の指定等。
SOAP エンベロープ		SOAP ヘッダと SOAP ボディを含む。 エンベロープ自身にはネームスペース以外の記述のみ具体的な処理内容およびデータは、SOAP ヘッダと SOAP ボディに記述される。
	SOAP ヘッダ	SOAP ボディのメッセージの管理に関するデータを記述する。 ・ヘッダ情報。
	SOAP ボディ	処理を実行するためのメッセージを記述する。 ・処理メッセージ (WFS コマンド、ステータス等)。
添付ファイル		画像・メッシュ情報・更新 LOG 等。

表 3.2-3 に以降で説明される表記の凡例を示す。

表 3.2-3 表記の凡例

表記	意味
<pre> &lt;tagname&gt;   &lt;tagname2&gt;     ....   &lt;/tagname2&gt;   &lt;tagname3&gt;     &lt;tagname31&gt;       ....     &lt;/tagname31&gt;   &lt;/tagname3&gt; &lt;/tagname&gt; </pre>	XML タグ表記。 <tagname> と </tagname> の間を 1 ブロックとする。ブロックは入れ子表現を許す。
[.....]	ブロック。
[.....]?	0 回または 1 回。
[.....]*	0 回以上。
[.....]+	1 回以上。
[項目 1   項目 2   ...]	複数項目から 1 つを選択する。
<i>italic</i>	指示、応答の任意内容。

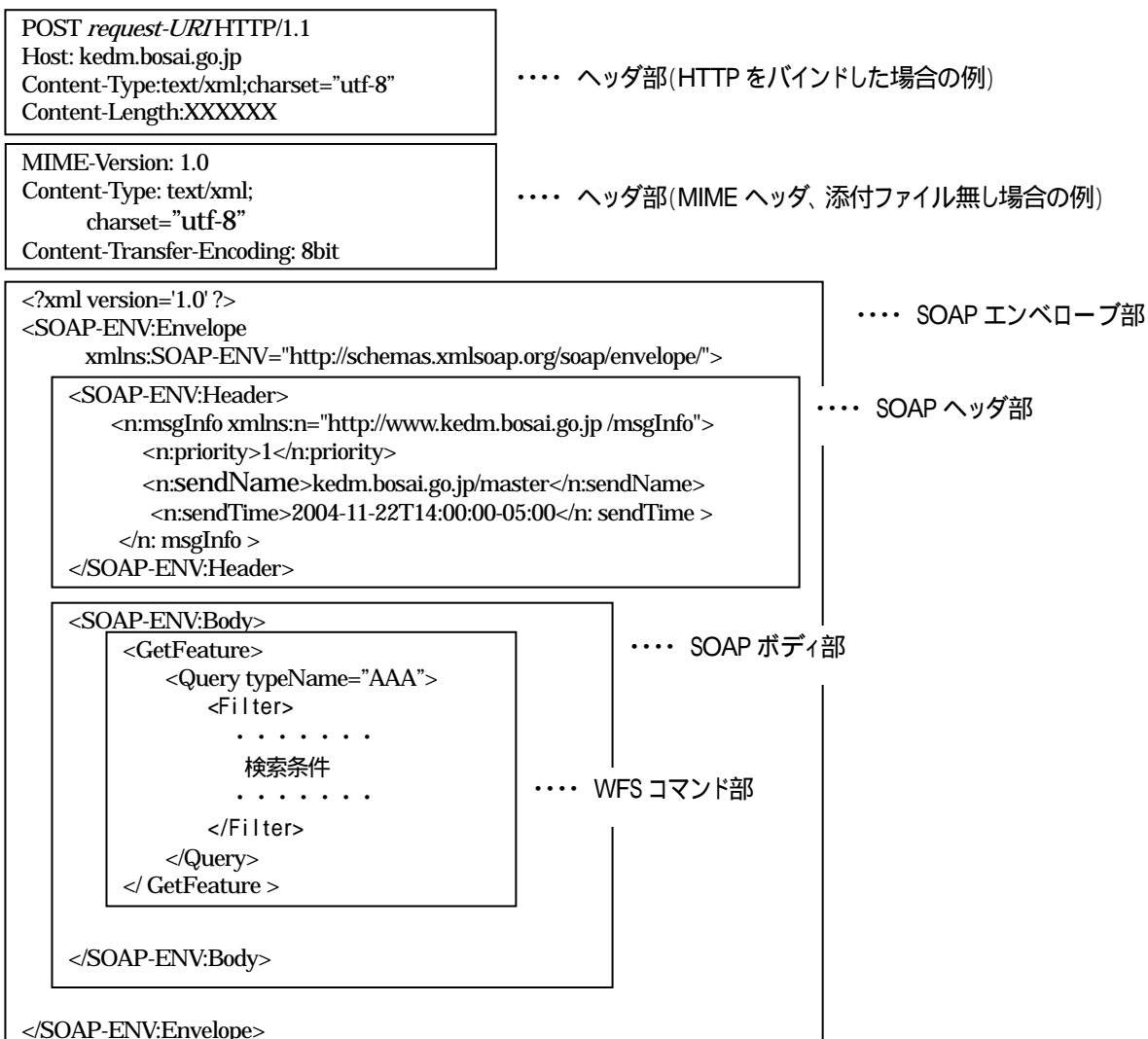
以下に代表的な電文の記述例を示す。

記述例 1 :

データ転送プロトコルとしてHTTP<sup>8)</sup>を利用して検索条件に見合った”AAA”の情報を要求する検索要求。

処理プライオリティ 1、送信時間 2004 年 11 月 22 日 午前 5 時 0 分

送信者 : kedm.bosai.go.jp/master



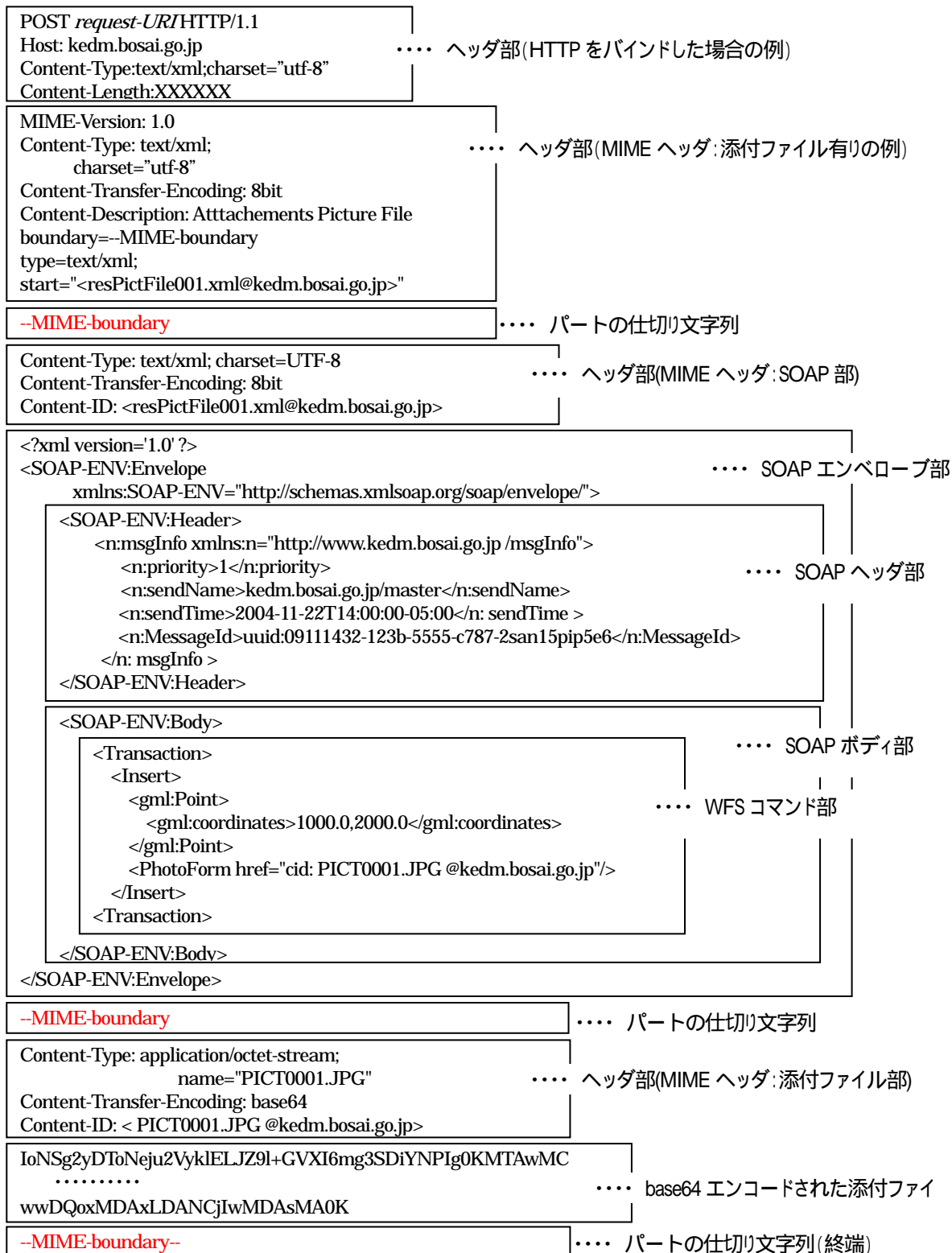
記述例 2 :

データ転送プロトコルとしてHTTP<sup>8)</sup>を利用して画像ファイルを添付ファイルとして送  
出する。

処理プライオリティ 1、送信時間 2004 年 11 月 22 日 午前 5 時 0 分

送信者 : kedm.bosai.go.jp/master

添付ファイル名 : PICT0001.JPG



## ヘッダ (プロトコル・バインディングヘッダ)

### ( 1 ) ヘッダ (プロトコル・バインディングヘッダ)

概要：

送受信に SOAP で利用するデータ送信プロトコルに対するヘッダ情報を記述する。

実装するトランスポートプロトコルに依存するヘッダで、この中にプロトコルごとに定められているヘッダ情報が記述される。これを受け取ったサーバは、続くメッセージが SOAP であることを理解して SOAP に関する処理を実行する。

SOAP 1.1<sup>6)</sup>では、実際にデータの送受信に使う下位プロトコル(トランスポートプロトコル)は、すでに広く普及しているHTTP(Hyper Text Transfer Protocol)<sup>8)</sup>やSMTP (Simple Mail Transfer Protocol)<sup>11)</sup>、FTP(File Transfer Protocol)<sup>12)</sup>などから選択できるようになっており、組織間で利用する場合でもファイヤーウォールなどを安全に通過することができる。

このように、SOAP を使用することによりトランスポートプロトコルに依存しないメッセージ交換が可能であるため、本仕様ではトランスポートプロトコルについては規定しない。

記述例：

一般的なデータ転送プロトコルであるHTTP(Hyper Text Transfer Protocol)<sup>8)</sup>にバインドしPOSTメソッドを用いた場合の記述形式例を示す。

```
POST request-URI HTTP/1.1
Host :host-name
Content-Type:text/xml;charset="utf-8"
Content-Length: *****
```

注記

*request-URI* : 送信元を識別する URI

*host-name* : 送信先ホスト名

charset="utf-8" : テキストが記述されている文字コード UTF-8 が一般的

参考：

HTTP(Hyper Text Transfer Protocol)<sup>8)</sup>記述構文の詳細については「RFC2616」<sup>8)</sup>参照のこと。

## ヘッダ (MIME ヘッダ)

### ( 2 ) ヘッダ (MIME ヘッダ)

概要：

電文に添付ファイルが存在するかの識別を記述する。

本仕様は、SOAPにファイルを添付する「SOAP Messages with Attachments」<sup>9)</sup>の仕様に準拠して記述する。

記述形式：

```
MIME-Version: 1.0
Content-Type: MIME-Media-Type,
charset="utf-8"
Content-Transfer-Encoding: 7bit | 8bit
[Content-Description: Description-message?]
[boundary=MIME-boundary-String?]
type=text/xml;
[start="<SOAP-Envelope-Part-Name>"]?
```

注記

MIME-Version :

MIME のバージョンを表す。現在は Version1.0。

*MIME-Media-Type* :

電文メッセージの種類を表す。

ファイルを添付しない場合は、SOAP メッセージのみになるので text/xml を使用する。

ファイルを添付する場合は、 multipart/related を使用し、*MIME-boundary-String* で仕切られる各パート毎に MIME ヘッダを定義し、パート単位での意味内容を指示する。

詳細は後述「2.4 添付ファイル」参照。

charset :

文字コードを指示する。(暫定的に UTF-8(Unicode)を使用する)

他に ISO-2022-JP(日本語)、US-ASCII(英語)、ISO-8859-1(ヨーロッパの主要言語)等も使用可能。

Content-Transfer-Encoding :

符号方式。

7bit ... 記述内容が全て 7bit 文字の場合。

8bit ... 1 Byte でも 8bit 文字が含まれている場合。

*Description-message* :

パートの説明を記述する。

*MIME-boundary-String* :

添付ファイルが存在する場合に記述する。

Content-Type が multipart/related の場合に必要。

*SOAP-Envelope-Part-Name* :

添付ファイルが存在する場合に記述する。

SOAP エンベロープ部をファイル化した場合の名称を記述する。

#### 記述例

添付ファイルが存在しない場合。

```
MIME-Version: 1.0
Content-Type: text/xml;
charset="utf-8"
Content-Transfer-Encoding: 8bit
Content-Description: Non Attachements File
type=text/xml;
```

添付ファイルが存在する場合。

```
MIME-Version: 1.0
Content-Type: text/xml;
charset="utf-8"
Content-Transfer-Encoding: 8bit
Content-Description: Attachements File
boundary=--MIME-boundary
type=text/xml;
start="<SOAP-Envelope.xml>"
```

## SOAP エンベロープ

### ( 3 ) SOAP エンベロープ

概要：

SOAP メッセージのルート要素。エンベロープ自身には名前空間（ネームスペース）のみの記述となる。

具体的な処理の内容は SOAP ヘッダ・SOAP ボディに記述される。

記述形式：

名前空間には"SOAP-ENV"を使用する。

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  SOAP ヘッダ
  SOAP 本体
</SOAP-ENV:Envelope>
```

参考：

名前空間については「XML Information Set」参照(付録：A-4)。

## SOAP ヘッダ

### ( 4 ) SOAP ヘッダ

#### 概要：

SOAP ボディに記載される内容に関して記述する。  
減災情報プラットフォーム独自に拡張した要素により構成される。

#### 記述形式：

```
<SOAP-ENV:Header>
  <n: MsgHeader xmlns:n="http://www.kedm.bosai.go.jp /schema/2005">
    <n:priority> any priority number </n:priority>
    <n:sendName> send URI </n:sendName>
    <n:sendTime> sendding dateTime </n: sendTime >
    <n:MessageId> UUID sendTo </n:MessageId>
    <n:ResponseTo> UUID responseTo </n:ResponseTo>
  </n: MsgHeader >
</SOAP-ENV:Header>
```

#### 注記

*any priority number* ：  
処理プライオリティ。 1 ～ 1 2 8 (高 1 ～ 1 2 8 低)。

*send URI* ：  
処理者識別子 送信者(組織)を示す任意の URI。

*sendding dateTime* ：  
処理送信時間 ISO8601 dateTime型<sup>(13) 14)</sup>。  
タイムゾーンは JST とする。

*UUID sendTo* ：  
処理要求時は UC の UUID<sup>A6)</sup>。  
処理結果応答時は US の UC の UUID<sup>A6)</sup>。

記述例：  
uuid:09111432-123b-2392-p423-9add28pon3j1

*UUID responseTo* ：  
送信応答 ID。  
US が受信した UC の MessageId を使用。

#### 参考：

これらの要素は将来的に必要なに応じて追加される。  
ISO8601 の dateTime 時間表現<sup>(13) 14)</sup>。

dateTime	年月日時分秒	YYYY-MM-DDThh:mm:ssTZD	2004-06-02T16:27:15Z
----------	--------	------------------------	----------------------

頭文字 T は日付と時間のコンポーネントを分離するために使用される。



## SOAP ボディ

### ( 5 ) SOAP ボディ

概要：

UC、US が処理すべきメッセージの本文およびエンコードされた添付ファイルを実際にする XML データとして記述する。リクエストでは、WFS コマンドとパラメータ、添付ファイル等が記述され、レスポンスでは、処理結果やエラー情報がこの場所に記載される。

記述形式：

```
<SOAP-ENV:Body>
```

```
  処理メッセージ
```

```
</SOAP-ENV:Body>
```

処理メッセージ内容に関しては「 2 . 3 リクエストとレスポンス」に記述する。

### 3.3 リクエストとレスポンス

WFS (Web Feature Service) プロトコルに準拠し独自の拡張をしたものを用いる。WFSで定義されているプロトコルのうち、以下のものをサポートする<sup>3)</sup>。

また、本仕様においては添付ファイルの規定を定める。

添付ファイル仕様は後述「2.4 添付ファイル」参照

GetCapabilities	US が提供するサービスに関する情報の問い合わせ。
DescribeFeatureType	登録されているデータの型（応用スキーマ、事象の型）の情報を、XML Schema の形式で問い合わせる。
GetFeature	データベースを検索しデータを取得する。
Transaction	
- Insert	データベースに新たなデータを追加記録する。
- Update	データベースに記録されているデータの内容を変更する。
- Delete	データベースに記録されているデータを削除する。
- Query	データベースに対し検索要求を行う(GetFeature と関連)。
Register	データベースに新たなデータ型を定義し、その型のデータを扱うための設定などを行う。

US から情報を取得する一連の使用例

UC は GetCapabilities リクエストを US に送り、US から提供サービス情報のレスポンスを受け取る（提供サービス情報は、US の利用可能なサービスのリストである）。次に、UC は DescribeFeatureType リクエストを送る。このリクエストは GetCapabilities のレスポンスから取得された FeatureType を含むことができる。

US は要求されたデータの型に関する情報を XML Schema の形式でレスポンスする。これらの情報を元に UC は GetFeature をリクエストすることにより US からデータを取得する。（図 3.3-1）

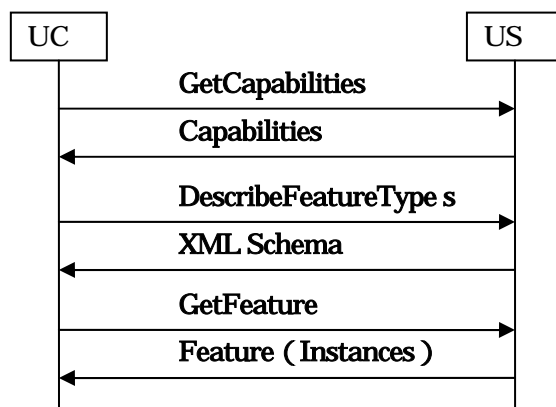


図 3.3-1 US から情報を取得する例

## GetCapabilities

### ( 1 ) GetCapabilities

#### 概要：

US が提供するサービス情報リストの取得要求を行う。  
 GetCapabilities の要求に対して WFS\_Capabilities により応答する。  
 応答内容には US が公開するサービスのリスト、データベース操作コマンド、処理可能な地物名称の一覧が含まれる。

#### 記述形式：

##### 処理要求時

```
<GetCapabilities version="1.0.0" service="WFS"
    xmlns=http://www.opengis.net/wfs" />
```

##### 処理応答時

```
<WFS_Capabilities>
  <Capabilities>
    list of services
  </Capabilities>
  <FeatureTypeList>
    [<Operations>
      list of operations available for all feature types
    </Operations>]*
    [<FeatureType>
      <Name>
        a name of a feature type
      </Name>
    </FeatureType>]*
  </FeatureTypeList>
</WFS_Capabilities>
```

#### 注記

*list of services* : USが公開する利用可能なサービスリストが列挙される。

```
<GetCapabilities/>
<DescribeFeatureType/>
<Transaction/>
<GetFeature/>
<Register/>
```

*list of operations available for all feature types* :

すべてのデータ型に対して利用可能なデータベース操作のリストが列挙される。  
 <Insert/>

<Update/>

<Delete/>

<Query/>

*a name of a feature type* : 情報共有データに定義されている参照可能な地物名称。

## DescribeFeatureType

### ( 2 ) DescribeFeatureType

概要：

UC は US が公開提供するデータ型（応用スキーマ、事象の型）を取得要求する。  
US は該当情報を XML Schema 形式で応答する。

記述形式：

処理要求時

```
<DescribeFeatureType>
  [<TypeName> a name of a feature type </TypeName>]+
</DescribeFeatureType>
```

注記

*a name of a feature type* :

情報共有データに定義されている参照可能なデータ型。  
GetCapabilities により応答された FeatureType を記述する。

処理応答時

```
<xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
  [ schema definitions for the feature types ]+
</xsd:schema>
```

注記

*schema definitions for the feature types* :

Register プロトコルによって登録されるものと同じ内容。  
情報共有データに定義されている地物スキーマのリスト。  
XML Schema 形式で記述する。

## GetFeature

### ( 3 ) GetFeature

#### 概要：

US のデータベースを条件によって検索しデータを取得する。下記文字列を US に送信する。GetFeature の要求に対しては FeatureCollection で応答する。

#### 記述形式：

##### 処理要求時

```
<GetFeature>
  <Query typeName="Name of Toplevel Element"> .....
    <Filter>
      any conditions .....
    </Filter>
  </Query>
</GetFeature >
```

#### 注記

*Name of Toplevel Element* :

Register プロトコル中の XML Schema で定義されたデータ型  
( トップレベルの element の名前 )

*any conditions* :

対象データに対する検索条件を記述する。

OGC Filter Encoding Implementation Specification<sup>15)</sup>に従って検索条件を記述する。

##### 処理応答時

```
<FeatureCollection xmlns:gml="http://www.opengis.net/gml">
  <ElapsedTime>
    <gml:beginPosition> Start Time </gml:beginPosition>
    <gml:endPosition> End Time </gml:endPosition>
  </ElapsedTime>
  [<gml:featureMember>
    element of retrieved data
  </gml:featureMember>]*
</FeatureCollection>
```

#### 注記

*Start Time, End Time* :

検索の開始および終了時刻(ISO8601 dateTime)。

タイムゾーンは JST とする。

*element of retrieved data* :

検索結果データ

#### 記述例：

矩形領域(100.0,200.0)-(200.0,300.0)に包含されるビル情報の取得要求を行いその結果としてビルの頂点座標値(130.0,220.0)-(130.0,245.0)-(170.0,245.0)-

(170.0,220.0)-(130.0,220.0)を応答する記述例。

#### 検索処理要求時

```
<GetFeature>
  <Query typeName="Building"> ..... 矩形範囲指定
    <Filter> ..... 検索条件指定
      <BBox> ..... 矩形範囲指定
        <PropertyName>gml:geometryProperty</PropertyName>
        <gml:Box>
          <gml:coordinates> ..... 矩形座標指定
            100.0,200.0
            200.0,300.0
          <gml:coordinates>
        </gml:Box>
      </BBox>
    </Filter>
  </Query>
</GetFeature>
```

#### 処理結果応答時

```
<FeatureCollection xmlns:gml="http://www.opengis.net/gml">
  <ElapsedTime>
    <gml:beginPosition> 2004-06-02T16:27:15 </gml:beginPosition>
    <gml:endPosition> 2004-06-02T16:27:17 </gml:endPosition>
  </ElapsedTime>
  <gml:featureMember>
    <Building>
      <gml:geometryProperty>
        <gml:LineString>
          <gml:LineString>
            <gml:Coordinates>
              130.0,220.0
              130.0,245.0
              170.0,245.0
              170.0,220.0
              130.0,220.0
            </gml:Coordinates>
          </gml:LineString>
        </gml:LineString>
      </gml:geometryProperty>
    </Building>
  </gml:featureMember>*
</FeatureCollection>
```

## Transaction

### ( 4 ) Transaction

#### 概要：

データの追加記録 ( Insert )、更新 ( Update ) および削除 ( Delete ) には、WFS の Transaction プロトコルを用いる。

一つの Transaction プロトコル中に複数の Insert、Update、Delete の記述が可能であり、これら複数の Insert、Update、Delete は混在してもよい。

Insert、Update、Delete は、出現順に処理される。

#### 記述形式：

##### 処理要求時

Transaction の共通の形式は以下の通りである。

```
<Transaction>
  [ Insert element | Update element | Delete element ]+
</Transaction>
```

#### 注記

*Insert*、*Update*、*Delete* の各 *element* 記述方法は ( 4-1 ) ~ ( 4-3 ) 参照。

#### 処理応答時

Transaction 要求に対する処理結果の復帰情報となる。

```
<TransactionResponse>
  <ElapsedTime>
    <gml:beginPosition>Start Time</gml:beginPosition>
    <gml:endPosition>End Time</gml:endPosition>
  </ElapsedTime>
  <TransactionResult>
    <Status> [ SUCCESS | FAIL ] </Status>
    [<Message>message from US</Message>]?
  </TransactionResult>
</TransactionResponse>
```

#### 注記

*Start Time*, *End Time* :

検索の開始および終了時刻(ISO8601 dateTime)。  
タイムゾーンは JST とする。

*SUCCESS* | *FAIL* :

Transaction 要求処理が成功した場合は SUCCESS、  
失敗した場合は FAIL を設定する。

*message from US* :

処理結果に対するメッセージ。



## Insert

### ( 4 - 1 ) Insert

#### 概要：

データベースに対してデータエレメントを追加する。

#### 記述形式：

```
<Insert>
  [ Data Element ]+
</Insert>
```

#### 注記

*Data Element* 挿入しようとする地物の記述。

#### 記述例：

平面座標値 (-13022.0, -52682.0) - (-13018.0, -52671.0) - (-13017.0, -52676.0) - (-13022.0, -52682.0) のを持つ地物「Building」をデータベースに挿入する。

```
<Insert>
  <Building>
    <gml:geometryPropaty>
      <gml:Polygon>
        <gml:LinerRing>
          <gml:coordinates>
            -13022.0,-52682.0
            -13018.0,-52671.0
            -13017.0,-52676.0
            -13022.0,-52682.0
          </gml:coordinates>
        </gml:LinerRing>
      </gml:Polygon>
    </gml:geometryPropaty>
  </Building>
</Insert>
```

## Update

### ( 4 - 2 ) Update

概要：

データベースのデータエレメントを変更する。

記述形式：

```
<Update typeName="Name of Data Element"
           mode="preserve or restore or override">
  [<Property>
    <Name>XPath of the property to modify</Name>
    <Value>new value</Value>
  </Property>]+
  [<Filter>
    any conditions
  </Filter>]?

```

注記

*Name of Data Element* :

更新対象要素名称。

*XPath of the property to modify* :

更新対象属性名称。

*new value* :

更新後の属性値。

*any conditions* :

更新対象データに対する絞り込み条件。

OGC Filter Encoding Implementation Specification<sup>15)</sup>に従って絞り込み条件を記述する。

<Update>タグ中の mode 属性の値は以下のような意味を持つ。

- mode の値が *preserve* である場合、元のデータエントリには Update フラグが付けられるだけで、データベースからは消去されない。
- mode の値が *restore* である場合、元のデータエントリにある Update フラグが消され、データベースから復元する。
- mode の値が *override* である場合、元のデータエントリはデータベースから消去される。

mode 属性が省略された場合には、*preserve* 指定(デフォルト)として処理される。

## Delete

### ( 4 - 3 ) Delete

概要：

データベースのデータを削除する。

記述形式：

```
<Delete typeName="Name of Data Element"
           mode="preserve or restore or override">
  <Filter>
    any conditions
  </Filter>
</Delete>
```

注記

*Name of Data Element* :

削除対象要素名称。

*any conditions* :

削除対象データに対する絞り込み条件。

OGC Filter Encoding Implementation Specification<sup>15)</sup>に従って絞り込み条件を記述する。

<Delete>タグ中の mode 属性の値は以下のような意味を持つ。

- mode の値が *preserve* である場合、元のデータエントリには deleted フラグが付けられるだけで、データベースからは消去されない。
- mode の値が *restore* である場合、元のデータエントリにある deleted フラグが消され、データベースから復元する。
- mode の値が *override* である場合、元のデータエントリはデータベースから消去される。

mode 属性が省略された場合には、*preserve* 指定(デフォルト)として処理される。

## Register

### ( 5 ) Register

概要：

UC がデータベースに新しいデータ型を登録する。

記述形式：

```
<Register uri=" URI for the new data type">
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:gml="http://www.opengis.net/gml">
    any XML schema definition.
  </xsd:schema>
</Register>
```

注記

*URI for the new data type* ：  
この登録を参照する際の ID として用いる一意名。  
*any XML schema definition* ：  
登録しようとする任意のXML Schema定義。

文法：

```
<xsd:element name='Register' type='RegisterType'/>
<xsd:complexType name='RegisterType'>
  <xsd:complexContent>
    <xsd:sequence>
      <xsd:element name='xsd:schema'
                    type='xsd:schemaType'
                    minOccurs='0' maxOccurs='unbounded'/>
    </xsd:sequence>
    <xsd:attribute name='uri' type='uri'/>
  </xsd:complexContent>
</xsd:complexType>
```

ただし、上記の `xsd:schemaType` は、「3.1 利用可能なXML Schema タグと属性」を参照。

記述例：

include 機能を用いてすでに登録されたデータ型を参照する場合。

```
<Register uri="bar">
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:gml="http://www.opengis.net/gml">
    <xsd:include schemaLocation="foo" />
    any XML schema definition.
  </xsd:schema>
</Register>
```

この例では、'foo'としてすでに登録されているスキーマを参照して新たなデータ型あるいはスキーマを定義する。この定義は、後で'bar'として参照することができる。注意しなければならないのは、この uri が、データの型の名前と異なることである。Register プロトコルでは、XML Schema を用いて複数のデータ型を同時に登録することができる。よって、Register タグの uri は同時に登録された複数のデータ型に対応することになる。

### 3.4 添付ファイル

概要：

減災情報共有データとしてメッシュ情報、画像情報等のファイルが想定される。

プロトコルとして SOAP メッセージに対しファイルを添付してやりとりを行う仕様として「SOAP Messages with Attachments」(以降 Attachments 仕様と表記)が定められており同仕様に準じて添付ファイルの処理を行う。

これは SOAP1.1 仕様の関連仕様という位置づけになる。

Attachments 仕様により SOAP メッセージに添付ファイルを付加したメッセージ全体を「SOAP メッセージ・パッケージ」と呼ぶ。(図 3.4)

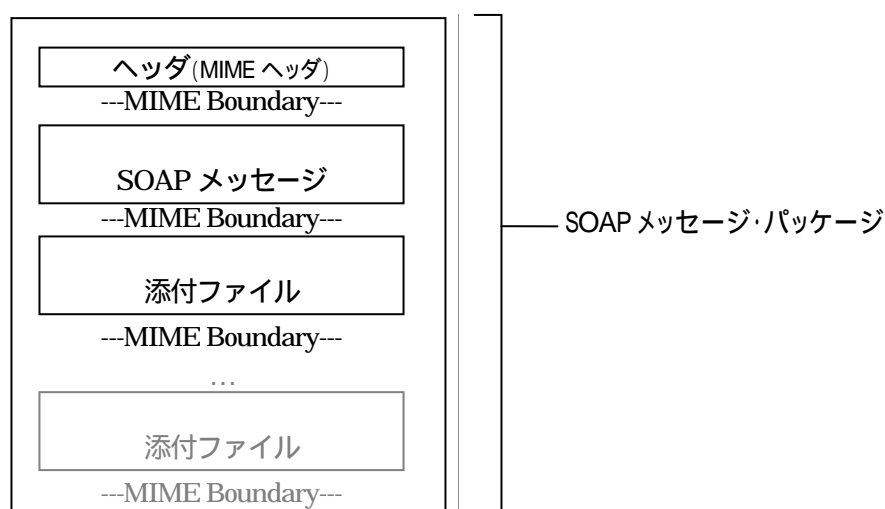


図 3.4 Attachments 仕様

このパッケージ構造は添付ファイルを含む HTML 形式のメール等で使用されている multipart/related の MIME 構造になっている。添付ファイルが複数個の場合は MIME Boundary で仕切り必要個数を羅列する。

添付ファイルの関連づけは SOAP メッセージ内の「cid」(Content-ID)と添付ファイルパートの Content-ID により行う。(図 3.4-1)

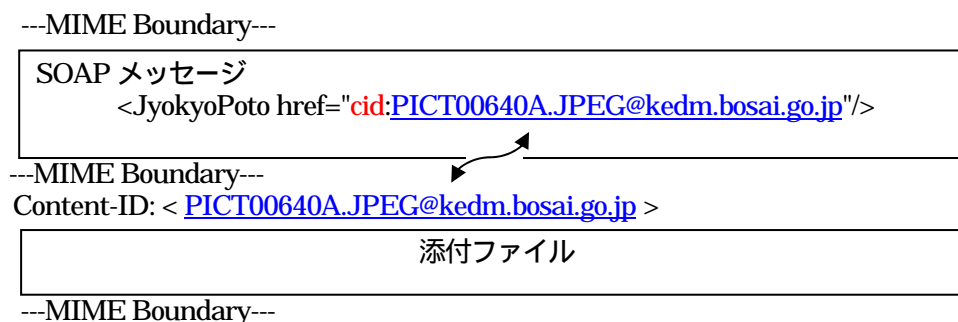


図 3.4-1 SOAP メッセージと添付ファイルの関連づけ

添付可能なファイルは MIME 仕様により許可されているファイル形式 (メディアタイプ) に準ずる。

- |  |
|--|
| <ul style="list-style-type: none"><li>(1) text -- テキスト情報</li><li>(2) image -- 画像データ</li><li>(3) audio -- 音響データ</li><li>(4) video -- ビデオ・データ</li><li>(5) application -- その他の応用データ</li></ul> |
|--|

図 3.4-2 添付可能なメディアタイプ

記述形式：

ヘッダの記述（記述詳細は2.2（2）ヘッダ（MIMEヘッダ）参照）

```
MIME-Version: 1.0
Content-Type: Multipart/Related;
charset="utf-8"
Content-Transfer-Encoding: 7bit | 8bit
[Content-Description: Description-message]?
boundary=MIME-boundary-String
type=text/xml;
start="<SOAP-Envelope-Part-Name>"
```

プライマリパート（SOAPエンベロープ）の記述

```
MIME-boundary-String
Content-Type: text/xml;
charset="utf-8"
Content-Transfer-Encoding: 7bit | 8bit
Content-ID: < SOAP-Envelope-Part-Name >

<?xml version='1.0' ?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    ..
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    ..
    <AttachFile href="cid: Attachement-File-Name" />
    ..
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

添付ファイルパートの記述

```
MIME-boundary-String
Content-Type: MediaType;
charset=UTF-8
Content-Transfer-Encoding: Base64
Content-ID: < Attachement-File-Name >
```



注記

*MIME-boundary-String* :

各パートを仕切るための文字列。

使用文字は空白無し文字列長とし 69 文字以内とする。

電文最終行に終端判定として「*MIME-boundary-String*」  
+「--」がセットされる。

*SOAP-Envelope-Part-Name* :

SOAP エンベロープ部をファイル化した場合の名称。

*Attachement-File-Name* :

一意に定められたファイル名称。

記述形式 : ファイル名@送信者識別名

記述例 : pict0001.jpg@kedm.bosai.go.jp

*MediaType* :

添付ファイルのタイプを MIME のメディアタイプ  
(Content-Type)に準拠して記述。

Content-Type: には主型と副型をスラッシュで区切って記述する。大文字小文字は区別しない。たとえば、通常のテキストなら Text/Plain、GIF イメージなら Image/Gif となる。以下に MIME の 5 つの主型を列挙する。

Text ..... テキストデータ

通常のテキストなら Text/Plain、XML なら Text/xml。

Image ..... 画像データ

GIF、JPEG、TIFF、PNG などは、それぞれの名前がそのまま副型になる。

Audio ..... 音声データ

8 ビットの PCM は Audio/Basic。

Video ..... ビデオ・データ

副型には MPEG がある。

Application .. さまざまなアプリケーション独自のデータを示すときに使用する。

たとえば、PostScript ファイルは Application/PostScript になる。

また、任意のファイルという意味を示すために、  
Application/Octet-Stream が用意されている。

注意

添付ファイルのエンコードは Base64 のみとし、バイナリーファイルのエンコードは使用しない。

Content-Transfer-Encoding: Base64

Content-Transfer-Encoding: binary ×

# ● 添付ファイルの応用

SOAP Messages with Attachments は本来 SOAP ボディ内部に記述されるべき XML 記述のデータを一般化したファイルとして添付ファイルにすることも可能とする。この応用範囲としては、減災情報共有プラットフォームデータベース (KIWI+) の更新差分データレコードなど GML の冗長な記述部分をそっくりと切り出しファイル化し Base64 エンコードにより添付ファイルとすることにより実現できる、また、プラットフォームデータベースのデータレコードを Application/Octet-Stream のメディアタイプで送受信することも可能になり、POI+ 領域情報など減災情報共有プラットフォーム間の高精度な情報交換も GML スキーマ定義の範疇を超えて可能となる。

添付ファイル化し送受信サイズを小さくすることにより、通信インフラに対する負荷を軽減することができる。

基図の初期生成及び大量データの更新など、更新量が大量の場合でも添付ファイルを利用すると、減災情報プラットフォームは直接取り込むことが出来るためロスが少ない。

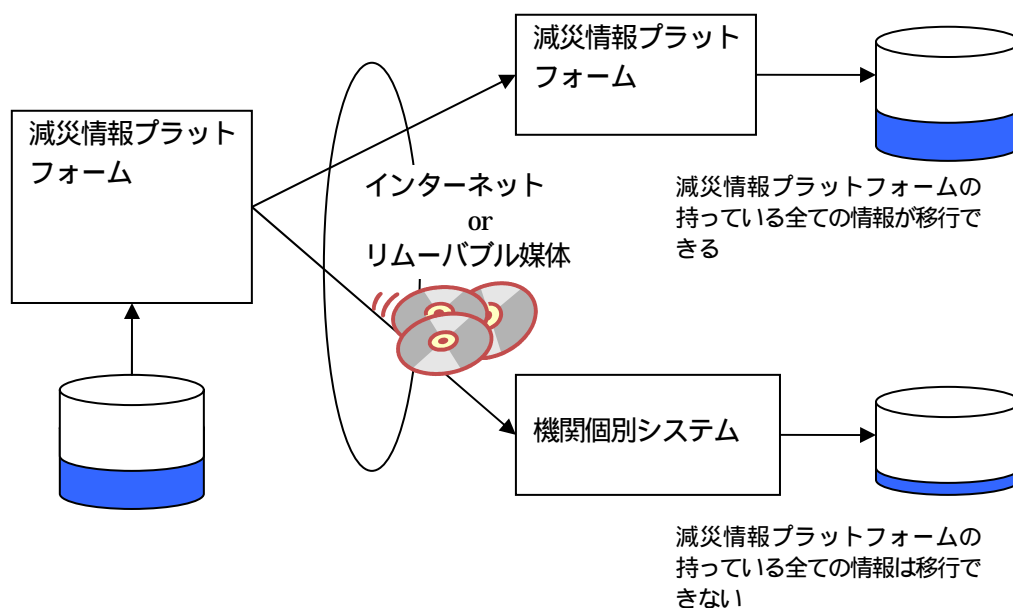


図 2.4-3 添付ファイルの応用

## 4 . 制約事項等

### 4 . 1 利用可能な XML Schema タグと属性

当プロトコルで認識されるXML Schemaのタグおよび属性 ( attribute ) は、以下のものとする。

```
<xsd:schema [xmlns:prefix="URI"]*>
<xsd:include schemaLocation="URI">
<xsd:element name="Name of Element" type="Name of Type"
    [minOccurs="number"] [maxOccurs="number"]>
<xsd:complexType name="Name of Type">
<xsd:group name="Name of Group">
<xsd:sequence>
<xsd:choice>
<xsd:all>
<xsd:complexContent>
<xsd:simpleContent>
<xsd:extension base="Name of Type">
<xsd:restriction base="Name of Type">
```

### 4 . 2 Filter 記述上の制約

GetFeatureおよびTransactionのUpdate、Deleteプロトコルでは、検索条件をOGC Filter Encoding Implementation Specification に従って記述するが、現状では同規格に完全には対応していない。具体的には以下のような制約がある。

空間オペレーション ( Spatial Operators ) では、図形の包含関係などを表すが、現状では外接長方形による同等の空間オペレーションで近似されている。このため、厳密な検索を行うためには、検索結果に対し再度幾何学計算を行う必要がある。

WFS の Filter 定義では複数回現れる element に対する条件の記述は定められてない。このような element に対する条件記述では、以下のタイプの論理オペレータを追加する必要がある

for-all : 全てのelementで条件成立  
exists : どれかひとつのelementで条件成立  
nth : n番目のelementで条件成立

#### 4.3 Filter 属性の記述要素

Filter 属性では、下記のオペレータを用いて記述することができる。

- 空間オペレータ  
これらオペレータの意味はOpenGIS Simple Features Specification For SQL Revision 1.1<sup>(18)</sup>に述べられている。  
Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects, Contains, BBox
- 比較オペレータ  
PropertyIsEqualTo, PropertyIsGreaterThan, PropertyIsLessThan, PropertyIsBetween, PropertyIsLike, PropertyIsNull
- 論理オペレータ  
And, Or, Not
- その他、算術表現  
Add(+), Sub(-), Mul(×), Div(/), PropertyName, Literal, Function

記述例：

SAMPLE\_DATE とされる時刻 “ 2001-01-15T20:07:48 ” から時刻 “ 2001-03-06T12:00:00 ” までの間に存在する地物を指定する。

```
<Filter>
  <Between>
    <PropertyName>SAMPLE_DATE</PropertyName>
    <LowerBoundary>
      <Literal>2001-01-15T20:07:48</Literal>
    </LowerBoundary>
    <UpperBoundary>
      <Literal>2001-03-06T12:00:00</Literal>
    </UpperBoundary>
  </Between>
</Filter>
```

DEPTH が 400 から 800 までの間で、WKB\_GEOM の領域にあるポリゴンを指定する。

```
<Filter>
  <AND>
    <Within>
      <PropertyName>WKB_GEOM</PropertyName>
      <Polygon name="1" srsName="EPSG:4326">
        <outerBoundaryIs>
          <LinearRing>
            <coordinates>-98.3,23.2, ...</coordinates>
          </LinearRing>
        </outerBoundaryIs>
      </Within>
      <PropertyIsBetween>
        <PropertyName>DEPTH</PropertyName>
        <LowerBoundary>400</LowerBoundary>
        <UpperBoundary>800</UpperBoundary>
      </PropertyIsBetween>
    </AND>
  </Filter>
```

#### 4.4 事象とジオメトリの取り扱い

GML の幾何プリミティブを分類すると大きく point (点) curve (曲線) surface (局面) solid (立体) に分類できるが情報共有を行う上では複雑な立体形状表現は必ずしも必須といえないため 0 ~ 2 次元までの基本表現のみを取り扱うこととし solid (立体) に関しては取り扱わない。但し、高さ情報等は対象物の属性として取り扱う。

具体的には 印の表現記述を使用する。(図 4.4.1)

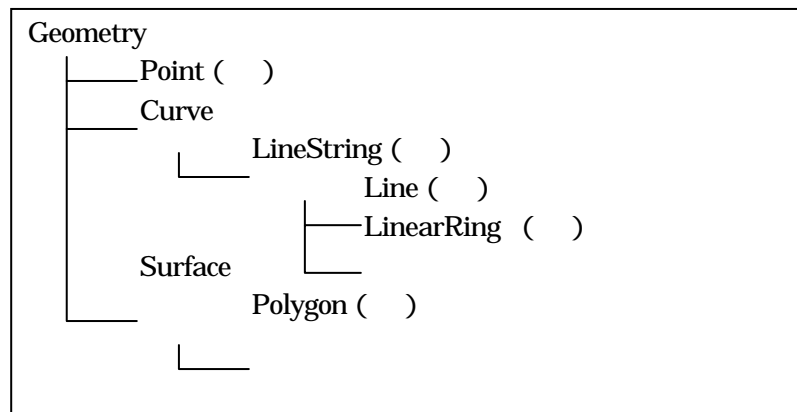


図 4.4.1 ジオメトリ階層

本プロトコル仕様においては実体 (位置とその属性を表す「点実体」、線的な要素に対応する「線実体」、面的な要素に対応する「面実体」) に対して構成するジオメトリが付随し、それに属性及び添付ファイルが付与された組み合わせの事象のみを取り扱う。

またその構成は普遍的・一般的な表現により下記の様に表すことができる。(図 4.4.2)

事象 = <実体種別、実体 ID、<ジオメトリ、時間情報 [、<属性>*][、<添付情報>*]>*>	
実体種別	..... 実体を表す名称
実体 ID	..... <点 線 面>
時間情報	..... 発生開始時間, 発生確定時間, 消滅開始時間, 消滅確定時間
属性	..... <属性タイプ、<値>*>
添付情報	..... <ファイルタイプ,ファイル本体>
	<...> 表現ブロック
	[...] 省略可
	* 繰り返し

図 4.4.2 事象表現

#### 4.5 データ単位系について

単位系（距離、時間、位置座標）は下記の表記により行う。

距離	メートル
時間	秒
位置座標	緯度経度

#### 4.6 時間スキーマ対応

GML を用いれば、時間経過に伴って変化する事象を履歴(history)、経路(track)プロパティで定義することができる。但し、移動するオブジェクト(人、自動車、鉄道等)を対象としないため、ある期間の始まりと終わりを表現する記述を使用する。

減災共有プラットフォームにおいては、「発生が開始した時間(a)」、「発生が確定した時間(b)」、「消滅が開始した時間(c)」、「消滅が終了した時間(d)」の4つの時間を用いて管理する。

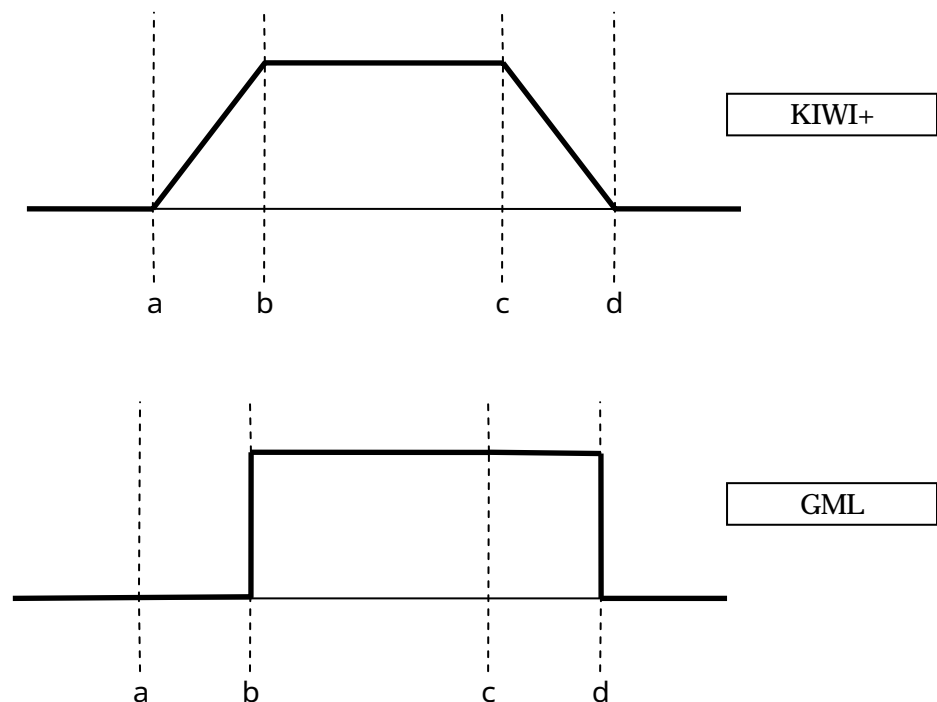


図 4.6 時間概念の対応

図 4.6 では発生が開始した時間から発生が確定した時間、消滅が開始した時間から消滅が終了した時間までを同じ事象として取り扱った。

下記の例では、ある学区「District 15」に学校「Alpha」を建築した例である。プロパティには name(名前)、boundeBy(境界)、および extentOf(広がり)を用意した。

EnvelopeWithTimePeriod における timePosition プロパティの値は 1960 年 1 月 1 日 ~ 1972 年 3 月 14 日と表現することにより、この期間には学区「District15」に学校「Alpha」が存在したことを示している。同様に例 2 では指定期間には学校は存在しないことを示している。

例 1 :

```
<app:SchoolDistrict gml="SD100">
  <gml:name>District 15</gml:name>
  <gml:boundedBy>
    <gml:EnvelopeWithTimePeriod srsName="..">
      <gml:timePosition>1960-01-01</gml:timePosition>
      <gml:timePosition>1972-03-14</gml:timePosition>
    </gml:EnvelopeWithTimePeriod>
  </gml:boundedBy>
  <gml:featureMember>
    <app:School>
      <gml:name>Alpha</gml:name>
      <gml:address>100 Kawasaki</gml:address>
      <gml:location>
        ...
      </gml:location>
    </app:School>
  </gml:featureMember>
</app:SchoolDistrict>
```

例 2 :

```
<app:SchoolDistrict gml="SD100">
  <gml:name>District 15</gml:name>
  <gml:boundedBy>
    <gml:EnvelopeWithTimePeriod srsName="..">
      <gml:timePosition>1950-01-01</gml:timePosition>
      <gml:timePosition>1959-12-31</gml:timePosition>
    </gml:EnvelopeWithTimePeriod>
  </gml:boundedBy>
  <gml:featureMember>
    <app: School >
      <gml:name> ... </gml:name>
      <gml:address> ... </gml:address>
      <gml:location>
        ...
      </gml:location>
    </app:School>
  </gml:featureMember>
</app:SchoolDistrict>
```

#### 4.7 時間情報対応

時間情報を以下のように取り入れることにより、空間データの時間軸上の管理を行うことができる。時間情報は、あらゆる地物オブジェクトに加えることができる。

オブジェクトデータには2種類の状態が考えられる。1つはデータの持ち方であり、もう1つはデータの時間軸上の状態である。

オブジェクトの持ち方は、図 4.7 object1 ~ object4 のように4種類の状態が考えられ、オブジェクトに対する WFS の Delete の操作としては次のように定義した。

```
<Delete typeName="Name of Data Element"
  mode="preserve or restore or override">
  <Filter>
    any conditions
  </Filter>
</Delete>
```

<Delete>タグ中の mode 属性の値は以下のような意味を持つ。

- mode の値が *preserve* である場合、元のデータエントリには deleted フラグが付けられるだけで、データベースからは消去されない。
- mode の値が *restore* である場合、元のデータエントリにある deleted フラグが消され、データベースから復元する。
- mode の値が *override* である場合、元のデータエントリはデータベースから消去される。

このように mode 属性を定義することにより、地物オブジェクトデータに対して削除・回復・抹消を指定することが可能となる。

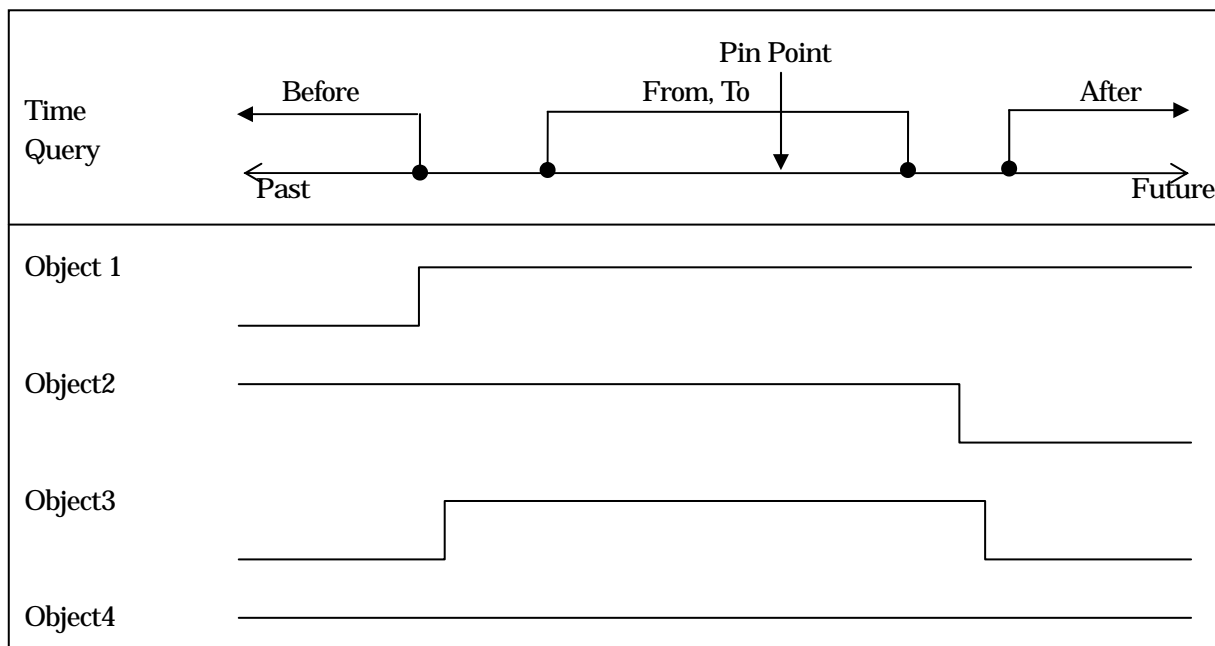


図 4.7 データへの時間情報の反映



時間軸上での状態に対しては、図 4.7 に示すように 4 種の時間指定が考えられる。  
この操作は、地物オブジェクトの生存時間（開始・終了）という時間属性の更新であると考え、WFS の Update によって操作することとした。

```
<Update typeName="Name of Data Element"
           mode="preserve or restore or override">
  [<Property>
    <Name>XPath of the property to modify</Name>
    <Value>new value</Value>
  </Property>]+
  [<Filter>
    any conditions
  </Filter>]?
</Update>
```

<Update>タグ中の mode 属性の値は以下のような意味を持つ。

- mode の値が *preserve* である場合、元のデータエントリには Update フラグが付けられるだけで、データベースからは消去されない。
- mode の値が *restore* である場合、元のデータエントリにある Update フラグが消され、データベースから復元する。
- mode の値が *override* である場合、元のデータエントリはデータベースから消去される。

このように mode 属性を定義することにより、Filter で指定された生存時間（開始時間・終了時間）を指定することが可能となり、オブジェクトを時間軸上で操作可能となる。

時間属性の更新例：

- Before           : 開始時刻 “ 0000-00-00T00:00:00 ”、  
                      終了時刻 “ 2005-03-31T12:30:23 ”  
                      “ 0000-00-00T00:00:00 ” は時刻を指定しない場合に用いることとする。
- After            : 開始時刻 “ 2005-03-20T12:30:23 ”、  
                      終了時刻 “ 0000-00-00T00:00:00 ”
- From, To        : 開始時刻 “ 2005-03-13T08:30:23 ”、  
                      終了時刻 “ 2005-03-23T12:30:23 ”
- Pin Point        : 開始時刻 “ 2005-03-31T12:30:23 ”、  
                      終了時刻 “ 2005-03-31T12:30:23 ”

## 4.8 メッシュ情報の記述

## 概要：

被覆表現について、GMLは4つの独立した型とプロパティを提供している(表 3.6)。本仕様においてはRectifiedGridCoverage(補正グリッド被覆)を使用する。

GMLにおけるメッシュデータ表現は、以下の3つの内部構成により表現される。

- (1) 定義域集合 (domainSet)                      メッシュデータの幾何形状表現。
- (2) 値域集合 (rangeSet)                      幾何形状にマッピングされる値。
- (3) 被覆関数 (coverageFunction)              定義域集合に値域集合をマッピングさせる際の規則。

これらの表現により多岐にわたるメッシュ表現が可能になるが、本仕様においては定義域集合 (domainSet)、値域集合 (rangeSet) のみの構成により表現する。

表 4.8 GML3.1 被覆と定義域プロパティ

被覆タイプ		定義域プロパティ(符号化)	
MultiPointCoverage	多重点被覆	multiPointDomain	多重点定義域
MultiSurfaceCoverage	多重曲面被覆	multiSurfaceDomain	多重曲面定義域
GridCoverage	グリッド被覆	gridDomain	グリッド定義域
RectifiedCoverage	補正グリッド被覆	rectifiedDomain	補正グリッド定義域

原点(origin)、オフセットベクトル(offsetVector)の要素項目で表現される。(図 3.6)

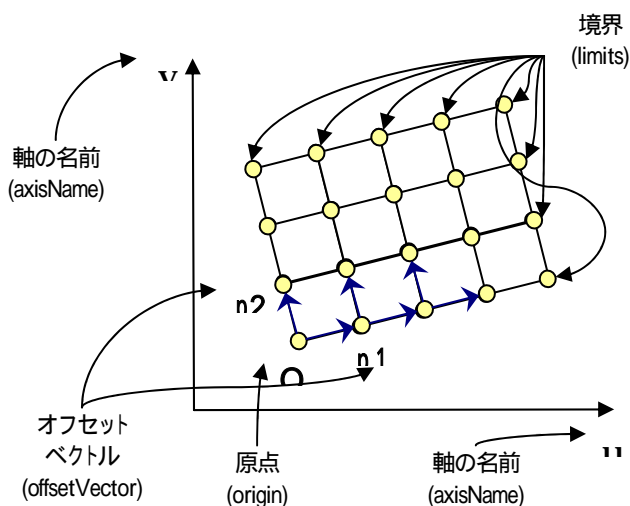


図4.8 補正グリッド被覆について

境界(limits) :

low、high のプロパティ要素により構成される。

メッシュのグリッド構成点数。

軸の名前(axisName) :

グリッドの縦横の軸に関する名称。

原点(origin) :

グリッドの原点座標。

オフセットベクトル(offsetVector) :

原点座標からの軸の方向と軸上のグリッド点間の距離を示すベクトル情報。

グリッド点の座標は以下のような単純なベクトル方程式によって与えられる。

$$P_{n,m} = np_1 + mp_2 + O$$

P1、p2 は図 3.6 に示されるオフセットベクトルである。n と m は境界(limit)で与えられる low、high による下(lowx,lowy)から上(highx,highy)までの整数値、O は原点のベクトルである。

記述形式 :

```
<gml:RectifiedGridCoverage>

  <gml:rectifiedGridDomain>
    <gml:RectifiedGrid dimension="coverage dimension">
      <gml:limits>
        <gml:GridEnvelope>
          <gml:low>list of lower bounds of each axis</gml:low>
          <gml:high>list of upper bounds of each axis</gml:high>
        </gml:GridEnvelope>
      </gml:limits>
      [<gml:axisName>name of the first axis</gml:axisName>]+
      <gml:origin>
        <gml:Point>
          <gml:coordinates>position of the origin</gml:coordinates>
        </gml:Point>
      </gml:origin>
      [<gml:offsetVector>offset vector for the first axis</gml:offsetVector>]+
    </gml:RectifiedGrid>
  </gml:rectifiedGridDomain>

  <gml:rangeSet>
    data part
  </gml:rangeSet>

</gml:RectifiedGridCoverage>
```

注記

*coverage dimension* :

グリッドの次元。

*list of lower bounds of each axis* :

グリッドの下限整数値。次元数分羅列される。

*list of upper bounds of each axis* :

グリッドの上限整数値。次元数分羅列される。

*name of the first axis* :

グリッド次元の各軸の名称。

*position of the origin* :

グリッドの原点座標。

*offset vector for the first axis* :

グリッド各次元の方向ベクトル。

*data part* :

グリッド各点に対する値のタプルリストを直接記述 (gml:DataBlock)、  
もしくは値を格納したファイル参照先 (gml:File) を記述する。

## 以降GGGD(Ver-0.86)より抜粋

### 1.8.5 被覆データ型

ラスターデータやメッシュデータなどの被覆データ型については、GML 3.1 の RectifiedGridCoverageType の記述方法に準拠する。ただし、同Coverage Model の記述方法にはいくつかの冗長性があるため、以下に示す形式のみを扱うものとする。

```
<gml:RectifiedGridCoverage>
  <gml:rectifiedGridDomain>
    <gml:RectifiedGrid dimension="coverage dimension">
      <gml:limits>
        <gml:GridEnvelope>
          <gml:low>list of lower bounds of each axis</gml:low>
          <gml:high>list of upper bounds of each axis</gml:high>
        </gml:GridEnvelope>
      </gml:limits>
      <gml:axisName>name of the first axis</gml:axisName>
      <gml:axisName>name of the second axis</gml:axisName>
      ...
      <gml:axisName>name of the last axis</gml:axisName>
    <gml:origin>
      <gml:Point>
        <gml:coordinates>position of the origin</gml:coordinates>
      </gml:Point>
    </gml:origin>
    <gml:offsetVector>offset vector for the first axis</gml:offsetVector>
    <gml:offsetVector>offset vector for the second axis</gml:offsetVector>
    ...
    <gml:offsetVector>offset vector for the last axis</gml:offsetVector>
  </gml:RectifiedGrid>
</gml:rectifiedGridDomain>
<gml:rangeSet>
  data part
</gml:rangeSet>
</gml:RectifiedGridCoverage>
```

*coverage dimension* は、グリッドの次元を表す。例えば、直線上に並んだデータ列の場合は1を、2次元地図上にマップされる場合は2を、3次元空間を埋める場合には3を指定する。

*list of {flower/upperg} bounds of each axis* には、グリッドの各次元の下限・上限を示す。

*name of the {ffirst/second/.../lastg} axis* には、グリッドの次元の名前を記述する。

*position of the origin* には原点の座標を指定する。

*offset vector for the {ffirst/second/.../lastg} axis* には、グリッドの各次元の方向ベクトルを記述する。よって、グリッドの各点の座標  $p$  は、dimension=3、原点が  $o$ 、グリッドの各次元の方向ベクトルが  $u$ 、 $v$ 、 $w$  とすれば、

$$p_{ijk} = o + iu + jv + kw$$

となる。ただし、添字の  $i, j, k$  は *list of {flower/upperg} bounds of each axis* に指定された範囲を定義域とする。

*data part* は、gml:File もしくはgml:DataBlock 形式のいずれかをとるものとする。  
gml:File 形式は以下の通りである。

```
<gml:File>
  <gml:rangeParameters>
    <gml:CompositeValue>
      <gml:valueComponent>
        <user-defined tag for the first parameter
        uom="urn for unit of the parameter" //>
      </gml:valueComponent>
      <gml:valueComponent>
        <user-defined tag for the second parameter
        uom="urn for unit of the parameter" //>
      </gml:valueComponent>
      ...
      <gml:valueComponent>
        <user-defined tag for the last parameter
        uom="urn for unit of the parameter" //>
      </gml:valueComponent>
    </gml:CompositeValue>
  </gml:rangeParameters>
  <gml:fileName>URI for the data file</gml:fileName>
  <gml:fileStructure>Record Interleaved</gml:fileStructure>
</gml:File>
```

gml:DataBlock 形式は以下の通りである。

```
<gml:DataBlock>
  <gml:rangeParameters>
    <gml:CompositeValue>
      <gml:valueComponent>
        <user-defined tag for the first parameter
        uom="urn for unit of the parameter" //>
      </gml:valueComponent>
      <gml:valueComponent>
        <user-defined tag for the second parameter
        uom="urn for unit of the parameter" //>
      </gml:valueComponent>
      ...
      <gml:valueComponent>
        <user-defined tag for the last parameter
        uom="urn for unit of the parameter" //>
      </gml:valueComponent>
    </gml:CompositeValue>
  </gml:rangeParameters>
  <gml:tupelList>list of parameters in CSV style</gml:tupelList>
</gml:DataBlock>
```

以下は、gml:DataBlock を用いた、平均気温・気圧のメッシュ上のデータの記述例である。

```
<app:AverageTempPressuer>
```

```

<gml:rectifiedGridDomain>
  <gml:RectifiedGrid dimension="2">
    <gml:limits>
      <gml:GridEnvelope>
        <gml:low>1 1</gml:low>
        <gml:high>9 6</gml:high>
      </gml:GridEnvelope>
    </gml:limits>
    <gml:axisName>u</gml:axisName>
    <gml:axisName>v</gml:axisName>
    <gml:origin>
      <gml:Point>
        <gml:coordinates>2.0,1.0</gml:coordinates>
      </gml:Point>
    </gml:origin>
    <gml:offsetVector>1.0 0.2</gml:offsetVector>
    <gml:offsetVector>-0.2 1.0</gml:offsetVector>
  </gml:RectifiedGrid>
</gml:rectifiedGridDomain>
<gml:rangeSet>
  <gml:DataBlock>
    <gml:rangeParameters>
      <gml:CompositeValue>
        <gml:valueComponent>
          <app:Temperature uom="urn:x-si:v1999:uom:degreesC"/>
        </gml:valueComponent>
        <gml:valueComponent>
          <app:Pressuer uom="urn:x-si:v1999:uom:kPA"/>
        </gml:valueComponent>
      </gml:CompositeValue>
    </gml:rangeParameters>
    <gml:tupleList>2.0,101.2 5.0,101.3 7.0,101.4 11.0,101.5
13.0,101.6 17.0,101.7 19.0,101.7 23.0,101.8 29.0,101.9
...
...</gml:tupleList>
  </gml:DataBlock>
</gml:rangeSet>
</app:AverageTempPressuer>

```

上記のデータ型は、gml:RectifiedGridCoverageType という名前で‘common.xsd’に登録されている。よって、例えば上記のapp:AverageTempPressuer というデータ型を扱えるようにするためには、下のような定義をRegister プロトコルにより登録すればよい。

```
<element name="AverageTempPressuer" type="gml:RectifiedGridCoverageType"/>
```

## 5. 使用例

以下に標高情報などのメッシュデータを取得し、シミュレーションを行い、結果をメッシュ情報ファイルとしてプラットフォームデータベースに登録する使用例を示す(図 4.0)。

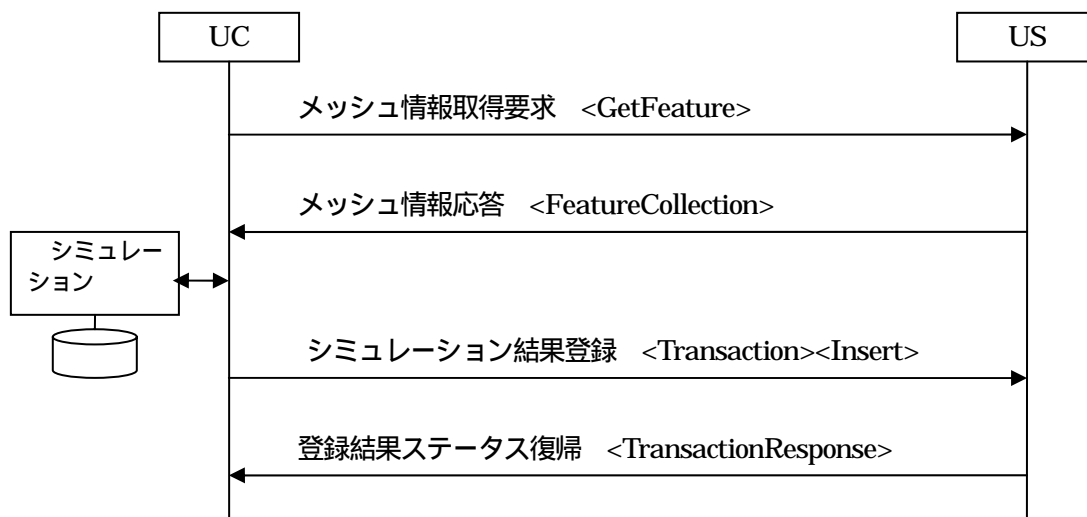


図 5.0 シミュレーション結果共有フロー

メッシュ情報取得要求

UC から US に対して、標高値のメッシュデータを要求する。

メッシュ情報応答

US から UC に対して、標高値のメッシュデータをタプルリストとして応答する。

シミュレーション

シミュレーションの結果（メッシュ情報）を CSV ファイルに出力する。

シミュレーション結果登録

により得られた CSV ファイルをメッシュ情報として US に登録する。

登録結果ステータス復帰

US により登録処理をした結果ステータスを復帰する。



## メッシュ情報取得要求

UC から US に対して、標高値のメッシュデータを要求する。

該当メッシュデータが「Altitude50m」であることは、事前に GetCapabilities により取得しておく。要求情報は 50m メッシュデータの 1000m 四方の情報を要求する。

```
MIME-Version: 1.0
Content-Type: text/xml;
charset="utf-8"
Content-Transfer-Encoding: 8bit
Content-Description: Non Attachments File
type=text/xml;
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <n:MsgHeader xmlns:n="http://www.kedm.bosai.go.jp/schema/2005">
      <n:priority>12 </n:priority>
      <n:sendName>master/kedm.bosai.go.jp</n:sendName>
      <n:sendTime>2004-06-02T16:27:14</n:sendTime>
      <n:MessageId>uuid:09111432-123b-5555-c787-2san15pip5e6</n:MessageId>
    </n:MsgHeader>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <GetFeature>
      <Query typeName="Altitude50m"> ..... 矩形範囲指定
        <Filter> ..... 検索条件指定
          <BBox> ..... 矩形範囲指定
            <PropertyName>gml:geometryProperty</PropertyName>
            <gml:Box>
              <gml:coordinates> ..... 矩形座標指定
                1000.0,2000.0
                2000.0,3000.0
              </gml:coordinates>
            </gml:Box>
          </BBox>
        </Filter>
      </Query>
    </GetFeature>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## メッシュ情報応答

US から UC に対して、標高値のメッシュデータをタプルリストとして応答する。  
この例はタプルリストとして冗長な記述の例であるが、「シミュレーション結果登録」の  
様にファイルとしての受け渡しも可能である。

```
MIME-Version: 1.0
Content-Type: text/xml;
charset="utf-8"
Content-Transfer-Encoding: 8bit
Content-Description: Non Attachements File
type=text/xml;
```

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <n: MsgHeader xmlns:n="http://www.kedm.bosai.go.jp /schema/2005">
      <n:priority>12 </n:priority>
      <n:sendName>master/kedm.bosai.go.jp</n:sendName>
      <n:sendTime>2004-06-02T16:27:14</n: sendTime >
      <n:MessageId>uuid:09111432-123b-2392-p423-9add28pon3j1</n:MessageId>
      <n:ResponseTo> uuid:09111432-123b-5555-c787-2san15pip5e6</n:ResponseTo>
    </n: MsgHeader >
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <FeatureCollection xmlns:gml="http://www.opengis.net/gml">
      <ElapsedTime>
        <gml:beginPosition> 2004-06-02T16:27:15 </gml:beginPosition>
        <gml:endPosition> 2004-06-02T16:27:17 </gml:endPosition>
      </ElapsedTime>
      <gml:rectifiedGridDomain>
        <gml:RectifiedGrid dimension="2">
          <gml:limits>
            <gml:GridEnvelope>
              <gml:low>1 1</gml:low>
              <gml:high>20 20</gml:high>
            </gml:GridEnvelope>
            </gml:limits>
            <gml:axisName>X</gml:axisName>
            <gml:axisName>Y</gml:axisName>
            <gml:origin>
              <gml:Point>
                <gml:coordinates>1000.0,2000.0</gml:coordinates>
              </gml:Point>
            </gml:origin>
            <gml:offsetVector>-50.0 50.0</gml:offsetVector>
          </gml:RectifiedGrid>
        </gml:rectifiedGridDomain>
```

```
<gml:rangeSet>
  <gml:DataBlock>
    <gml:rangeParameters>
      <gml:CompositeValue>
        <gml:valueComponent>
          <app:Temperature uom="urn:x-si:v1999:uom:m"/>
        </gml:valueComponent>
      </gml:CompositeValue>
    </gml:rangeParameters>
    <gml:tupletList>2.0 1.2 5.0 11.3 7.0 11.4 11.0 11.5
                    13.0 11.6 17.0 11.7 19.0 11.7 23.0 11.8 29.
                    ...</gml:tupletList>
  </gml:DataBlock>
</gml:rangeSet>
</FeatureCollection>
<SOAP-ENV:Body>
<SOAP-ENV:Envelope>
```

#### シミュレーション

シミュレーションの結果（メッシュ情報）を CSV ファイルに出力する。

#### シミュレーション結果登録

により得られた CSV ファイルをメッシュ情報として US に登録する。

```

MIME-Version: 1.0
Content-Type: text/xml;
charset="utf-8"
Content-Transfer-Encoding: 8bit
Content-Description: SimResult Attachements File
boundary="--MIME-boundary
type=text/xml;
start="<SimResult643.xml@kedm.bosai.go.jp>"

--MIME-boundary
Content-Type: text/xml; charset=UTF-8
Content-Transfer-Encoding: 8bit
Content-ID: <SimResult643.xml@kedm.bosai.go.jp>
<?xml version='1.0' ?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <n:MsgHeader xmlns:n="http://www.kedm.bosai.go.jp/schema/2005">
      <n:priority>12 </n:priority>
      <n:sendName>master/kedm.bosai.go.jp</n:sendName>
      <n:sendTime>2004-06-02T16:27:14</n: sendTime >
      <n:MessageId>uuid:09111432-123b-5555-c787-2san15pip5e6</n:MessageId>
    </n: MsgHeader >
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <Transaction>
      <Insert>
        <gml:rectifiedGridDomain>
          <gml:RectifiedGrid dimension="2">
            <gml:limits>
              <gml:GridEnvelope>
                <gml:low>1 1</gml:low>
                <gml:high>20 20</gml:high>
              </gml:GridEnvelope>
            </gml:limits>
            <gml:axisName>X</gml:axisName>
            <gml:axisName>Y</gml:axisName>
            <gml:origin>
              <gml:Point>
                <gml:coordinates>1000.0,2000.0</gml:coordinates>
              </gml:Point>
            </gml:origin>
            <gml:offsetVector>-50.0 50.0</gml:offsetVector>
          </gml:RectifiedGrid>
        </gml:rectifiedGridDomain>
      </Insert>
    </Transaction>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

```

        </gml:RectifiedGrid>
      </gml:rectifiedGridDomain>
    <gml:rangeSet>
      <gml:File>
        <gml:rangeParameters>
          <gml:CompositeValue>
            <gml:valueComponent>
              <app:Temperature uom="urn:x-si:v1999:uom:m"/>
            </gml:valueComponent>
          </gml:CompositeValue>
        </gml:rangeParameters>
        <gml:filename href="cid:simValue001.csv@kedm.bosai.go.jp"/>
        <gml:fileStructure>Record Interleaved</gml:fileStructure>
      </gml:File>
    </gml:rangeSet>
  </Insert>
</Transaction>
<SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--MIME-boundary
Content-Type: application/octet-stream;
          name=" simValue001.csv "
Content-Transfer-Encoding: base64
Content-ID: < simValue001.csv@kedm.bosai.go.jp>

IoNSg2yDT0Neju2VyklELJZ9l+GVXI6mg3SDiYNPIg0KMTAwMC
.....
wwDQoxMDAxLDANCjIwMDAsMA0K

--MIME-boundary--

```

#### 登録結果ステータス復帰

US により登録処理をした結果ステータスを復帰する。

```

MIME-Version: 1.0
Content-Type: text/xml;
charset="utf-8"
Content-Transfer-Encoding: 8bit
Content-Description: result  cmd - Insert
type=text/xml;
<?xml version='1.0' ?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <n: MsgHeader xmlns:n="http://www.kedm.bosai.go.jp /schema/2005">
      <n:priority>12 </n:priority>
      <n:sendName>master/kedm.bosai.go.jp</n:sendName>
      <n:sendTime>2004-06-02T16:30:16</n: sendTime >
      <n:MessageId>uuid:09111432-123b-2392-p423-9add28pon3j1</n:MessageId

```

```
<n:ResponseTo> uuid:09111432-123b-5555-c787-2san15pip5e6</n:ResponseTo>
</n:MsgHeader>
</SOAP-ENV:Header>
<SOAP-ENV:Body>
  <TransactionResponse>
    <ElapsedTime>
      <gml:beginPosition>2004-06-02T16:29:10</gml:beginPosition>
      <gml:endPosition>2004-06-02T16:29:11</gml:endPosition>
    </ElapsedTime>
    <TransactionResult>
      <Status>SUCCESS</Status>
    </TransactionResult>
  </TransactionResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## 参 考 文 献

- 1) General GeoGraphical Database(GGGD) System Ver-0.86  
独立行政法人 産業技術総合研究所 プロトコル
- 2) OpenGIS Geography Markup Language(GML)Implementation Specification  
version 3.0  
<http://www.opengeospatial.org/docs/02-023r4.pdf>, (2003).
- 3) WFS Open Geospatial Consortium .Inc (OGC),  
Web Feature Service Implementation Specification,  
<http://www.opengeospatial.org/docs/02-058.pdf>, (2002).
- 4) World Wide Web consortium (W3C), Extensible Markup Language (XML)  
<http://www.w3.org/XML/>.
- 5) World Wide Web consortium (W3C), SOAP Version 1.2  
<http://www.w3.org/TR/2001/WD-soap12-20010709/>, (2001).
- 6) Simple Object Access Protocol(SOAP) 1.1,  
<http://www.trl.ibm.com/projects/xml/SOAP1.1-j-ibm-revision2.html>, (2000).
- 7) 独立行政法人 防災科学技術研究所 地震防災フロンティア研究システム  
川崎ラボラトリー, 時空間データベース構造第 0.9 版, (2004),  
[http://www.kedm.bosai.go.jp/DyLUPAS/KIWI\\_PlusV09.zip](http://www.kedm.bosai.go.jp/DyLUPAS/KIWI_PlusV09.zip).
- 8) Hypertext Transfer Protocol -- HTTP/1.1 [RFC2616]  
The Internet Society,, (1999)  
<http://www.ietf.org/rfc/rfc2616.txt>.
- 9) SOAP Messages with Attachments W3C Note 11 December 2000  
<http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211> (2000)
- 10) MIME (Multipurpose Internet Mail Extension)  
The Internet Engineering Task Force RFC 2045 ~ 2049  
<http://www.ietf.org/rfc/rfc2045.txt>  
<http://www.ietf.org/rfc/rfc2046.txt>  
<http://www.ietf.org/rfc/rfc2047.txt>  
<http://www.ietf.org/rfc/rfc2048.txt>  
<http://www.ietf.org/rfc/rfc2049.txt>
- 11) SMTP (Simple Mail Transfer Protocol)  
The Internet Engineering Task Force RFC 2821  
<http://www.ietf.org/rfc/rfc2821.txt>
- 12) FTP (FILE TRANSFER PROTOCOL)  
The Internet Engineering Task Force RFC 959  
<http://www.ietf.org/rfc/rfc959.txt>
- 13) Date and Time Formats W3C ISO 8601  
<http://www.w3.org/TR/NOTE-datetime>
- 14) Numeric representation of Dates and Time  
(ISO : International Organization for Standardization)  
<http://www.iso.org/iso/en/prods-services/popstds/datesandtime.html>
- 15) Filter Encoding Implementation Specification OGC  
<http://www.opengeospatial.org/docs/02-059.pdf>

- 16) TCP/IP (Transmission Control Protocol / Internet Protocol)  
RFC 793 - Transmission Control Protocol DARPA INTERNET PROGRAM  
PROTOCOL SPECIFICATION September 1981  
<ftp://ftp.rfc-editor.org/in-notes/rfc793.txt>  
RFC 791 - Internet Protocol DARPA INTERNET PROGRAM PROTOCOL  
SPECIFICATION September 1981  
<ftp://ftp.rfc-editor.org/in-notes/rfc791.txt>
- 17) Base64 エンコード  
RFC 3548 - The Base16, Base32, and Base64 Data Encodings  
<ftp://ftp.rfc-editor.org/in-notes/rfc3548.txt>
- 18) OpenGIS Simple Features Specification For SQL Revision 1.1  
<http://www.opengeospatial.org/docs/99-049.pdf>,

## 付 録

### A.1 GML :

ISO TC211 が推し進め、国際標準として審議中であり認定されれば JIS 化が予定されている。  
「減災情報共有プロトコル」では、実践的なジオメトリである「点」、「線」、「面」を用いる。

### A.2 KIWI+ :

ISO TC204 において審議中の時空間データベース構造。その組み込み型のデータ構造の KIWI  
は、JIS D0810 として JIS 化されカーナビ業界標準として広く利用されている。「減災情報共有  
プロトコル」では、事象表現（実体の図形と属性の組）を用いる。

### A.3 電文内容および添付ファイル内容 :

今後、共有する減災情報の標準化研究等の成果により規定する。

### A.4 名前空間接頭辞

XML Information Set <http://www.w3.org/TR/xml-infoset/>

### A.5 IETF ( Internet Engineering Task Force ) <http://www.ietf.org/>

HTTP/1.0 RFC 1945 <http://www.ietf.org/rfc/rfc1945.txt?number=1945>

HTTP/1.1 RFC 2616 <http://www.ietf.org/rfc/rfc2616.txt?number=2616>

FTP RFC 959 <http://www.ietf.org/rfc/rfc0959.txt?number=959>

### A.6 UUID ( Universally Unique IDentifier )

世界中でユニークな 128bit 幅の 2 進数値である。オブジェクトやインターフェイスごとに固有  
の UUID を割り当て、お互いを識別する。UUID は、OSF ( Open Software Foundation ) の  
DCE ( Distributed Computing Environment ) 仕様によって決められた。実際の UUID では、  
ユニーク性を保証するために、UUID を作成した時間や作成に使用したマシンに装着されている  
ネットワーク・カードの MAC アドレスなどを数値の一部に組み入れたりしている。イーサネッ  
トの MAC アドレスは（登録制なので）世界中でユニークであることが保証されているからであ  
る。

引用 <http://www.atmarkit.co.jp/icd/root/52/94084052.html>

### A.7 Base64 エンコード

RFC 3548 により規定されている。

Base64 とは、3 バイトのデータ（バイナリデータを含む）を 4 バイトのテキストデータに変  
換するエンコード方式。例えば 3 バイトのバイナリデータは  $3 \times 8 = 24$  ビットで表される。これを



6 ビットごとの 4 つのデータに区切り、この区切られた 4 つのデータ(6 ビット)の上位 2 ビットを 00 とみなすと、4 バイトのデータをみなすことができる。各バイトは上位ビットが 0 のため、これらの 4 バイトはテキスト文字列として表現できる。以上により 3 バイトのバイナリデータが 4 バイトのテキストデータに変換される。