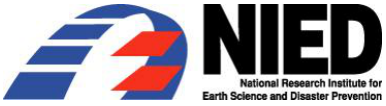
減災情報共有プロトコル

Ver. 1.00.038s

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 20 年 12 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。

目 次

第1章　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) とする情報管理を核とし、地理的属性を持たない一般の情報も含めた多種多様な情報共有を可能とする「減災情報共有プラットフォームの開発」が行われている。この開発においては既存システムや最新技術を用いたシステムとの接続を容易とするため、広く認知されている各種標準を基本とした「減災情報共有プロトコル」の仕様を定めることとなった。

本文書では、この仕様の規定の詳細について述べる。

第2章　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。

- 標準的で広く認知されている HTTP(Hyper Text Transfer Protocol)[15] や SMTP(Simple Mail Transfer Protocol)[6]、FTP(File Transfer Protocol) [2] 通信プロトコルの利用を可能とするため、本仕様では SOAP(Simple Object Access Protocol) [14, 1] を基本とした要求/応答プロトコルを規定する。なお本仕様では、HTTP、TCP/IP(Transmission Control Protocol / Internet Protocol) [13, 12] 等、下位の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)[9] をベースとし、XML(Extensible Markup Language)[3]、SOAP(Simple Object Access Protocol) [14]、GML(Geography Markup Language)*1*[10]、MIME(Multipurpose Internet Mail Extension) [5] によって電文を記述する。
- 要求/応答で扱われる単位データは、地理的情報を持たないものも含め、全て地物として扱われる。

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

なお、以下の説明では下記に挙げるネームスペース、プレフィクスの対応は、例文中で明示的に宣言していない場合、すでに暗黙的に宣言されているものとする。

misp	→	http://www.infosharp.org/misp
SOAP-ENV	→	http://schemas.xmlsoap.org/soap/envelope/
wfs	→	http://www.opengis.net/wfs
gml	→	http://www.opengis.net/gml
xsd	→	http://www.w3.org/2001/XMLSchema

第3章　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、受け手側のサーバのクライアント・サーバ型の形態をとる。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。クライアントはサーバに対しコネクションを確立後、要求メッセージを送信する。サーバは要求を受けたコネクションを通して、クライアントに応答メッセージを送信し、クライアントはサーバの応答メッセージを読み取り、コネクションをクローズする。

3.2　電文の構造

電文の構造は、SOAP メッセージ [14, 4] の構成とする。「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。

表 3.2 に以降で説明される表記の凡例を示す。

以下に代表的な電文の記述例を示す。

表 3.1: 電文の構造概要

電文各部		概要
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)。

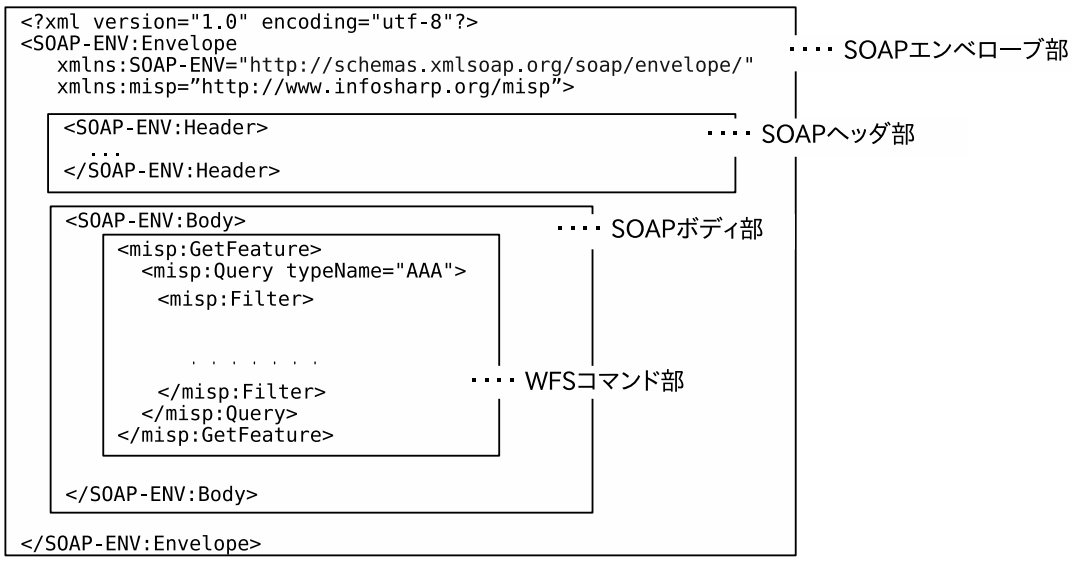
表 3.2: 表記の凡例

表記	意味

<tagname>
  <tagname2>
  ...
  </tagname2>
  <tagname3>
  <tagname31>
  ...
  </tagname31>
  </tagname3>
</tagname>
	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。
italic	指示、応答の可変な内容。

記述例:



SOAP エンベロープ

3.2.1 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:

SOAP-ENV の名前空間には ”http://schemas.xmlsoap.org/soap/envelope/” を使用する。

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

    SOAP ボディ

</SOAP-ENV:Envelope>
```

参考:

名前空間 *3* については「XML Information Set」参照。

SOAP ヘッダ

3.2.2　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。

記述形式:

```
<SOAP-ENV:Header>
        [任意要素]*
</SOAP-ENV:Header>
```

参考:

Header 中の要素は将来、必要に応じて追加される可能性がある。

SOAP ボディ

3.2.3 SOAP ボディ

概要:

クライアント、サーバが処理するべきメッセージの本文を 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中に記載される。

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 3.3 節 に記述する。

3.3 リクエストとレスポンス

WFS(Web Feature Service)[9] プロトコルを元に、独自に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。ただし、各要素のネームスペースは”http://www.opengis.net/wfs” (wfs) ではなく、”http://www.infosharp.org/misp” (misp) を用いるものとする。なお、RegisterFeatureType、RegisterCoordinateSystemTransformation は MISP で独自に拡張したものである。

GetCapabilities	サーバ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 - Update - Delete	 データベースに新たなデータを追加記録する。 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。 データベースに記録されているデータを削除する。
RegisterFeatureType	データベースに新たなデータ型を定義する。
RegisterCoordinateSystemTransformation	データベースに新たな座標系変換ルールを定義する。

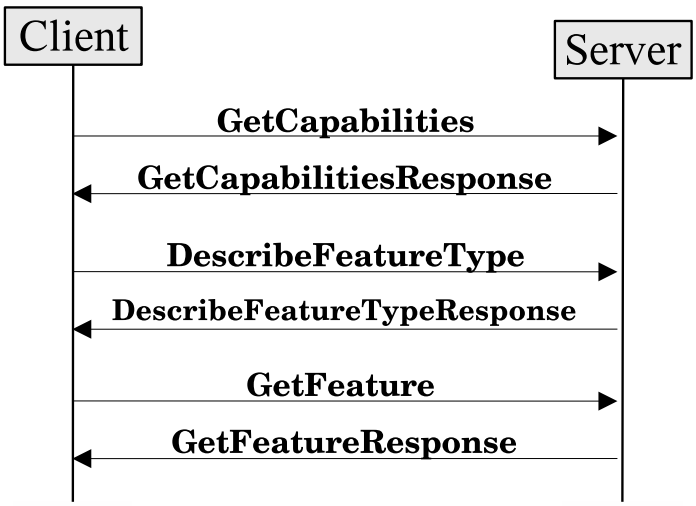


図 3.1: サーバから情報を取得する例

サーバから情報を取得する一連の使用例

クライアントは GetCapabilities リクエストをサーバに送り、サーバ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、サーバの利用可能なサービスのリストである）。次に、クライアントは 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した FeatureType を指定することができる。

サーバは要求されたデータの型に関する情報を XML Schema を含む形式で返答する。これらの情報を元にクライアントは GetFeature リクエストを行なうことによりサーバからデータを取得する。（図 3.1）

GetCapabilities

3.3.1 GetCapabilities

概要:

サーバ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。

応答内容にはサーバ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 xmlns="http://www.infosharp.org/misp"/>
```

処理応答時

```
<GetCapabilitiesResponse xmlns="http://www.infosharp.org/misp">
  <ResponseStatus>
    <MostRecentTransactionID> URI of the most recent transaction
</MostRecentTransactionID>
    <ElapsedTime>
      <gml:beginPosition> Begin Time </gml:beginPosition>
      <gml:endPosition> End Time </gml:endPosition>
    </ElapsedTime>
  </ResponseStatus>
  <WFS_Capabilities>
    <Capabilities> list of services </Capabilities>
    <FeatureTypeList>
      [ <Operations>
          list of operations available for all feature types
        </Operations> ]?
      [ <FeatureType>
          <Name xmlns:prefix="namespace">prefix: a name of a feature type</Name>
          <MostRecentTransactionID> URI of the most recent transac-
tion for the feature type </MostRecentTransactionID>
        </FeatureType> ]*
    </FeatureTypeList>
  </WFS_Capabilities>
</GetCapabilitiesResponse>
```

注記:

list of services : サーバ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/>
<RegisterFeatureType/>
<RegisterCoordinateSystemTransformation/>
```

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/>
```

name of 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。

URI of the most recent transaction for the feature type : その地物がもっとも最近更新された際の transaction を示す URI。

DescribeFeatureType

3.3.2 DescribeFeatureType

概要:

クライアントはサーバが公開提供するデータ型（スキーマ、事象の型）を取得要求する。
サーバは該当情報を XML Schema を含む形式で応答する。

記述形式:

処理要求時

```
<DescribeFeatureType xmlns="http://www.infosharp.org/misp">
  [ <TypeName xmlns:prefix="namespace">prefix: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

```
<DescribeFeatureTypeResponse xmlns="http://www.infosharp.org/misp">
  <ResponseStatus>
    <MostRecentTransactionID> URI of the most recent transaction
</MostRecentTransactionID>
    <ElapsedTime>
      <gml:beginPosition> Begin Time </gml:beginPosition>
      <gml:endPosition> End Time </gml:endPosition>
    </ElapsedTime>
  </ResponseStatus>
  [ <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="Namespace for this schema"
      xmlns:misp="http://www.infosharp.org/misp"
      misp:id="URI location for this schema">
    schema definitions for the feature types
  </xsd:schema> ]*
</DescribeFeatureTypeResponse>
```

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

3.3.3 GetFeature

概要:

サーバ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。下記文字列をサーバに送信する。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 xmlns="http://www.infosharp.org/misp"
            [ startPosition="Offset of features" ]
            [ maxFeatures="Maximum number of features" ]>
  <Query typeName="Name of Toplevel Element"
         [ mode="features" | "count" | "boundedBy" ]
         [ srsName="name of coordinate system" ]
         [ transformType="selective|conv|noconv" ]>
    <Filter>
      any conditions
    </Filter>
  [ <SortBy>
      [ <SortProperty>
          <PropertyName>Property Name for Sorting</PropertyName>
          <SortOrder>Sort Order</SortOrder>
        </SortProperty> ]+
    </SortBy> ]
  </Query>
</GetFeature>
```

注記:

Offset of features : 返答で結果を返す際のフィーチャーのオフセットを表わす正の整数。N の場合、N 番目のフィーチャーから返す。すなわち先頭から N-1 個のフィーチャーをスキップする。この値のデフォルトは 1。

Maximum number of features : 検索で返されるフィーチャーの最大数を表わす正の整数。

Name of Toplevel Element : RegisterFeatureType 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)。

mode : 検索のモードを指定する。features の場合は地物のリストを返す。count の場合は検索にマッチした地物の数を返す。boundedBy の場合は検索にマッチした地物の boundary box を返す。デフォルトは features。

srsName : 出力結果の地理情報の座標系を指定する。詳細は RegisterCoordinateSystemTransformation の項を参照。

transformType : 座標変換をどう行なうかを指定する。noconv の場合、座標変換を行なわない。conv の場合、srsName で示された座標系に変換して返す。selective の場合、srsName で示された座標系に属するフィーチャーのみを返す。

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 [8] に従って検索条件を記述する。

Property Name for Sorting : ソートのために使用するプロパティの名

Sort Order : 昇順もしくは降順を指定する。昇順の場合は ASC、降順の場合は DESC を指定する。

処理応答時

```
<GetFeatureResponse xmlns="http://www.infosharp.org/misp"
    xmlns:gml="http://www.opengis.net/gml">
  <ResponseStatus>
    <MostRecentTransactionID> URI of the most recent transaction
</MostRecentTransactionID>
    <ElapsedTime>
      <gml:beginPosition> Begin Time </gml:beginPosition>
      <gml:endPosition> End Time </gml:endPosition>
    </ElapsedTime>
  </ResponseStatus>
  [ <FeatureCount> the number of features </FeatureCount> ]
  <FeatureCollection>
    [ <gml:featureMember>
        an element of retrieved data
    </gml:featureMember> ]*
  </FeatureCollection>
</GetFeatureResponse>
```

注記:

Begin Time, *End Time* : 検索の開始および終了時刻 (ISO8601 dateTime)。

the number of features : 検索に適合した地物の数を返す。Query における mode 属性が count の場合に返される。

an 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) に包含される建物の情報の取得要求を行い、その結果として建物の頂点座標値 (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 xmlns="http://www.infosharp.org/misp"
    xmlns:gml="http://www.opengis.net/gml"
    xmlns:foo="http://...">
  <Query typeName="foo:Building"> <!-- ---------- 検索地物指定 -->
    <Filter>                      <!-- ---------- 検索条件指定 -->
      <BBox>                      <!-- ---------- 矩形範囲指定 -->
        <PropertyName>foo:outline</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

```
<GetFeatureResponse xmlns="http://www.infosharp.org/misp"
    xmlns:gml="http://www.opengis.net/gml">
  <ResponseStatus>
    <MostRecentTransactionID>uri:uuid:f359...7c47</MostRecentTransactionID>
    <ElapsedTime>
      <gml:beginPosition>2004-06-02T16:27:15Z</gml:beginPosition>
      <gml:endPosition>2004-06-02T16:27:17Z</gml:endPosition>
    </ElapsedTime>
  </ResponseStatus>
  <ElapsedTime>
  </ElapsedTime>
  <FeatureCollection>
    <gml:featureMember>
      <Building xmlns="http://...">
        <outline>
          <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>
        </outline>
      </Building>
    </gml:featureMember>
  </FeatureCollection>
</GetFeatureResponse>
```

Transaction

3.3.4 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 xmlns="http://www.infosharp.org/misp">
  [ Insert element | Update element | Delete element ]+
</Transaction>
```

注記: *Insert*、*Update*、*Delete* の各 *element* 記述方法は 3.3.4.1 節〜3.3.4.3 節参照。

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 xmlns="http://www.infosharp.org/misp"
     xmlns:gml="http://www.opengis.net/gml">
  <ResponseStatus>
    <MostRecentTransactionID> URI of the most recent transaction
</MostRecentTransactionID>
    <ElapsedTime>
      <gml:beginPosition> Begin Time </gml:beginPosition>
      <gml:endPosition> End Time </gml:endPosition>
    </ElapsedTime>
  </ResponseStatus>
  <TransactionResult>
    <TransactionID> URI of this Transaction </TransactionID>
    <Status> [ SUCCESS — FAILURE ] </Status>
  [ <Message> message from server </Message> ]?
  </TransactionResult>
</TransactionResponse>
```

注記:

Begin Time, *End Time* : 検索の開始および終了時刻 (ISO8601 dateTime)。

`SUCCESS` | `FAILURE` ：Transaction 要求処理が成功した場合は `SUCCESS`、失敗した場合は `FAILURE` となる。

message from server：処理結果に対するメッセージ。

Insert

3.3.4.1 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:

```
<misp:Insert xmlns:misp="http://www.infosharp.org/misp">
 [ Data Element ]*
</misp: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。

```
<misp:Insert>
   <Building>
      <outline>
          <gml:Polygon>
              <gml:LinearRing>
                  <gml:coordinates>
                      -13022.0,-52682.0
                      -13018.0,-52671.0
                      -13017.0,-52676.0
                      -13022.0,-52682.0
                  </gml:coordinates>
              </gml:LinearRing>
          </gml:Polygon>
      </outline>
   </Building>
</misp:Insert>
```

Update

3.3.4.2 Update

概要:

データベース中のエレメントの内容を変更する。

記述形式:

```
<misp:Update xmlns:misp="http://www.infosharp.org/misp"
typeName="Name of Data Element"
        [ mode="preserve"|"override"|"restore" ]>
  [ <misp:Property>
      <misp:Name>XPath of the property to modify</misp:Name>
     <misp:Value>new value</misp:Value>
    </misp:Property> ]+
    <misp:Filter>
      any conditions
    </misp:Filter>
</misp:Update>
```

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 [8] に従って絞り込み条件を記述する。

`<misp: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 に restore を指定することでデータを復元することができる。
- `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。restore でも復元できなくなる。
- `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` 属性が省略された場合には、 `preserve` が指定されたものとして処理される。

Delete

3.3.4.3 Delete

概要:

データベース中のデータを削除する。

記述形式:

```
<misp:Delete xmlns:misp="http://www.infosharp.org/misp"
             typeName="Name of Data Element"
             [ mode="preserve" | "override" | "restore" ]>
  <misp:Filter>
    any conditions
  </misp:Filter>
</misp: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 [8] に従って絞り込み条件を記述する。

`<misp: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 に restore を指定することでデータを復元することができる。
- `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。restore でも復元できなくなる。
- `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` 属性が省略された場合には、 `preserve` が指定とされたものとして処理される。

RegisterFeatureType

3.3.5 RegisterFeatureType

概要:

データベースに新しいデータ型を登録する。

記述形式:

```
<misp:RegisterFeatureType xmlns:misp="http://www.infosharp.org/misp">
  [ <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="Namespace for this schema"
      xmlns:misp="http://www.infosharp.org/misp"
      misp:id="URI location for this schema">
    any XML Schema definition.
  </xsd:schema> ]*
</misp:RegisterFeatureType>
```

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。

XML Schema による RegisterFeatureType 自体の文法定義:

```
<xsd:element name="RegisterFeatureType"
             type="RegisterFeatureTypeType"/>
<xsd:complexType name="RegisterFeatureTypeType">
  <xsd:complexContent>
    <xsd:sequence>
      <xsd:element ref="xsd:schema"
                   minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexContent>
</xsd:complexType>
```

注記: RegisterFeatureType は WFS では定義されておらず、MISP による拡張である。

MISP では使用可能なスキーマに制約を課している。詳細については 4.1 節を参照。

記述例:

import を用いてすでに登録されたデータ型を参照する場合。

```
<RegisterFeatureType xmlns="http://www.infosharp.org/misp">
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      targetNamespace="Namespace for this schema"
      xmlns:misp="http://www.infosharp.org/misp"
      misp:id="bar">
    <xsd:import namespace="foo" schemaLocation="bar" /> />
    any XML schema definition.
  </xsd:schema>
</RegisterFeatureType>
```

この例では、”bar”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(misp:id) が、データの型の名前と異なることである。RegisterFeatureType 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、RegisterFeatureType タグの URI(misp:id) 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。

処理要求時

RegisterFeatureType 要求に対する処理結果の復帰情報となる。

```
<RegisterFeatureTypeResponse xmlns="http://www.infosharp.org/misp">
  <ResponseStatus>
    <MostRecentTransactionID> URI of the most recent transaction
</MostRecentTransactionID>
    <ElapsedTime>
      <gml:beginPosition> Begin Time </gml:beginPosition>
      <gml:endPosition> End Time </gml:endPosition>
    </ElapsedTime>
  </ResponseStatus>
  <Status>SUCCESS</Status> <!-- または <Status>FAILURE</Status> -->
</RegisterFeatureTypeResponse>
```

RegisterCoordinateSystemTransformation

3.3.6 RegisterCoordinateSystemTransformation

概要:

地理データの座標系間の関係を登録する。

記述形式:

```
<misp:RegisterCoordinateSystemTransformation
    xmlns:misp="http://www.infosharp.org/misp">
  [<misp:Transformation>
    <misp:identifier>URI of this transformation</misp:identifier>
    <misp:sourceCS>name of source coordinate system</misp:sourceCS>
    <misp:targetCS>name of target coordinate system</misp:targetCS>
    <misp:transformationMethod>
      <misp:AffineTransformation>
        <misp:sourceDimensions>dimensions of source coordinate sys-
tem</misp:sourceDimensions>
        <misp:targetDimensions>dimensions of target coordinate sys-
tem</misp:targetDimensions>
        <misp:parameters>
          affine transformation parameters
        </misp:parameters>
      </misp:AffineTransformation>
    </misp:transformationMethod>
  </misp:Transformation>]*
</misp:RegisterCoordinateSystemTransformation>
```

注記:

URI of this transformation : この登録を参照する際の ID として用いる一意な名前

name of source coordinate system : 変換元座標系の一意な名前

name of target coordinate system : 変換先座標系の一意な名前

dimensions of source coordinate system : 変換元座標系の座標軸の数

dimensions of target coordinate system : 変換先座標系の座標軸の数

affine transformation parameters : アフィン変換のためのパラメータ列、実数値を空白文字で区切り羅列する。

注記: RegisterCoordinateSystemTransformation は WFS では定義されておらず、MISP による拡張である。

処理応答時: RegisterCoordinateSystemTransformation 要求に対する返答は以下のとおりである。

```
<misp:RegisterCoordinateSystemTransformationResponse
    xmlns:misp="http://www.infosharp.org/misp"
    xmlns:gml="http://www.opengis.net/gml">
  <misp:ResponseStatus>
    <misp:MostRecentTransactionID>URI of the most recent transaction
</misp:MostRecentTransactionID>
    <misp:ElapsedTime>
      <gml:beginPosition>Begin Time</gml:beginPosition>
      <gml:endPosition>End Time</gml:endPosition>
    </misp:ElapsedTime>
  </misp:ResponseStatus>
  <Status>SUCCESS</Status> <!-- または <Status>FAILURE</Status> -->
</misp:RegisterCoordinateSystemTransformationResponse>
```

MISP では、登録した地理データを必要な座標系に変換して取り出すことができる。そのためには座標系間の定義と、地理データを登録する際の地理データの座標系の指定が必要である。座標系変換定義はデータ登録の先、後のどちらでも行ってよい。座標系定義の例を以下に示す。なお、登録される変換は変換元座標系から変換先座標系への一方向の変換となる。そのため変換先座標系から変換元座標系への逆変換が必要な場合は別途登録する必要がある。

記述例:

```
<misp:RegisterCoordinateSystemTransformation
    xmlns:misp="http://www.infosharp.org/misp">
  <misp:Transformation>
    <misp:identifier>urn:misp:example:table1-to-room1</misp:identifier>
    <misp:sourceCS>urn:misp:example:table1</misp:sourceCS>
    <misp:targetCS>urn:misp:example:room1</misp:targetCS>
    <misp:transformationMethod>
      <misp:AffineTransformation>
        <misp:sourceDimensions>2</misp:sourceDimensions>
        <misp:targetDimensions>2</misp:targetDimensions>
        <misp:parameters>
          1 2 3
          4 5 6
        </misp:parameters>
      </misp:AffineTransformation>
    </misp:transformationMethod>
  </misp:Transformation>
</misp:RegisterCoordinateSystemTransformation>
```

この例では、urn:misp:example:table1 座標系から urn:misp:example:room1 座標系への変換を登録している。この変換により変換元座標系 urn:misp:example:table1 上の点 (x,y) は、以下のよ

うに変換先座標系 urn:misp:example:room1 の点 (x',y') に変換される。

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

地理データを登録する際には、以下のように地理形状に srsName 属性を付け、座標系も共に指定する。

```
<misp:Transaction xmlns:misp="http://www.infosharp.org/misp">
  <misp:Insert>
    <SampleObject xmlns="http://www.infosharp.org/misp/sample"
                  xmlns:gml="http://www.opengis.net/gml">
      <location>
        <gml:Point srsName="urn:misp:example:table1">
          <gml:coordinates>-1.8,2.0</gml:coordinates>
        </gml:Point>
      </location>
    </SampleObject>
  </misp:Insert>
</misp:Transaction>
```

以下のようにデータ取得時に、取得する座標系を指定して自動的にデータの座標を変換することができる。

```
<misp:GetFeature xmlns:misp="http://www.infosharp.org/misp"
                 xmlns="http://www.infosharp.org/misp/sample">
  <misp:Query typeName="SampleObject" srsName="urn:misp:example:room1">
    <misp:Filter>
      <misp:True/>
    </misp:Filter>
  </misp:Query>
</misp:GetFeature>
```

データ取得結果:

```
<misp:GetFeatureResponse xmlns:misp="http://www.infosharp.org/misp"
                         xmlns:gml="http://www.opengis.net/gml">
  <misp:ResponseStatus>
    <!-- 略 -->
  </misp:ResponseStatus>
  <misp:FeatureCollection>
    <gml:featureMember>
      <SampleObject xmlns="http://www.infosharp.org/misp/sample">
        <location>
          <gml:Point srsName="urn:misp:example:room1">
            <gml:coordinates>5.2,8.8</gml:coordinates>
          </gml:Point>
        </location>
      </SampleObject>
    </gml:featureMember>
  </misp:FeatureCollection>
</misp:GetFeatureResponse>
```

元データの座標系と、取得時の座標系の変換が直接登録されていない場合でも、複数回の変換の組み合せによって間接的に変換可能な場合、座標系間の関係を自動で検索し、変換を行う。例えば、座標系 A,B,C,D において、A→B, B→C, C→D の変換が与えられているとき、A 座標で登録されている点を D 座標で取得する場合は、自動的に A→B, B→C, C→D の組を発見して変換を行なう。

第4章　制約事項等

4.1　利用可能なXML Schemaタグと属性

本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`
  `misp:id="`*URI for this schema*`"`
  `TargetNamespace="`*URI for this schema*`"`
  [`xmlns:`*prefix*`="`*namespace URI*`"`*]**`>`
- `<xsd:import`
  `namespace="`*URI for other schema*`"`
  `schemaLocation="`*namespace URI*`">`
- `<xsd:element name="`*Name of Element*`" type="`*Name of Type*`"`
  [`minOccurs="`*number*`"`*]?*
  [`maxOccurs="`*number or* `unbounded"`*]?*
  [`misp:indexed="true"`*]?*`>` [1]
- `<xsd:complexType name="`*Name of Type*`">`
- `<xsd:group name="`*Name of Group*`">` [2]
- `<xsd:sequence>`
- `<xsd:choice>`
- `<xsd:all>`
- `<xsd:complexContent>`
- `<xsd:simpleContent>`
- `<xsd:extention base="`*Name of Type*`">`
- `<xsd:restriction base="`*Name of Type*`>"`

ただし、以下の項目はMISP独自の拡張であり、各々以下の意味を持つ。

- `xsd:schema`タグ中の `misp:id`: このschemaに対するID。`xsd:schema`の `schemaLocation`で指定する際に用いる。MISPで使用する場合は必ず付ける必要がある。
- `xsd:element`タグ中の `misp:indexed`: その要素(element)のデータでの検索(GetFeature)を高速化するためにデータベース中に、indexを作成する。

また、‘ _’ (アンダースコア)で始まる要素名、型名および属性名はプロトコル上の予約語として用いるため、上記Schemaにおいて定義してはならない。

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のFilter Encoding Implementation [8]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。

[1] 子要素に xsd:simpleType/xsd:complexType を許さない。将来的には解消予定。
[2] 現在の DaRuMa では未実装。

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 (Spacial 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[3]。
- 1 つの地物の中で複数回現れる element に対して条件を指定することはできない。

  WFS の Filter 定義では、複数回現れる element に対する条件の記述はまだ定められておらず、本プロトコルでもそのような検索は採用していない[4]。

また、MISP では暗黙の内、以下にあげるメタ情報を各地物に与える。これらは Filter の条件に用いることができる。

- `@_transaction_id_` : その地物が最後に変更 (登録を含む) された際のトランザクションの ID(URI) を示す。トランザクションの ID については、各レスポンスの返答文に書かれている `MostRecentTransaction` 等から取得できる。もし、サーバにとって未知の URI が指定された場合、あらゆる transaction より前として扱われる。
- `@_create_time_` : その地物が最初にデータベースに登録された時刻を示す。
- `@_update_time_` : その地物が最後に変更 (登録を含む) された時刻を示す。

例えば、1995 年 1 月 17 日午前 5 時 46 分 0 秒 (日本標準時) 以降に登録された地物 (Foo タイプ) を検索するためには、以下のように記述する。

```
<GetFeature>
  <Query typeName='Foo'>
    <Filter>
      <PropertyIsGreaterThan>
        <PropertyName>@_create_time_</PropertyName>
        <Literal>1995-01-17T05:46:00+09:00</Literal>
      </PropertyIsGreaterThan>
    </Filter>
  </Query>
</GetFeature>
```

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[5]。

- 空間オペレータ:

  Equals, Disjoint, Touches, Within, Overlaps, Crosses, Intersects, Contains, BBox

[3] これについては、将来的に解消する予定である。

[4] 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 1 つの element で条件成立
- nth: n 番目の element で条件成立

これらのオペレータの導入については次期バージョンで検討する。

[5] *印が付いているものは現在の DaRuMa では未実装

- 比較オペレータ:
  PropertyIsEqualTo, PropertyIsGreaterThan, PropertyIsGreaterThanOrEqualTo, PropertyIsLessThan, PropertyIsLessThanOrEqualTo, PropertyIsBetween *, PropertyIsLike*, PropertyIsNull*
- 論理オペレータ:
  And , Or , Not, True, False
- その他、算術表現
  Add(+)* , Sub(−)* , Mul(×)* , Div(/)*, PropertyName, Literal, Function*

これらオペレータの内、True と False については、以下の意味を持つ。

- `<misp:True/>` : 常に真となる。
- `<misp:False/>` : 常に偽となる。

残りのオペレータの意味は OpenGIS Simple Features Specification For SQL Revision 1.1 [11] に述べられている。

記述例:

SAMPLE_DATE という時刻プロパティが 2001-01-15 20:07:48 (JST) から 2001-03-06 12:00:00 (JST) の間に存在する地物の条件を指定する。

```
<misp:Filter>
  <misp:PropertyIsBetween>
    <misp:PropertyName>SAMPLE_DATE</misp:PropertyName>
    <misp:LowerBoundary>
      <misp:Literal>2001-01-15T20:07:48+09:00</misp:Literal>
    </misp:LowerBoundary>
    <misp:UpperBoundary>
      <misp:Literal>2001-03-06T12:00:00+09:00</misp:Literal>
    </misp:UpperBoundary>
  </misp:PropertyIsBetween>
</misp:Filter>
```

DEPTH が 400 から 800 までの間であり、指定されたポリゴン (各頂点の座標値が -98.3,23.2, ...) に location が含まれている地物の条件を指定する。

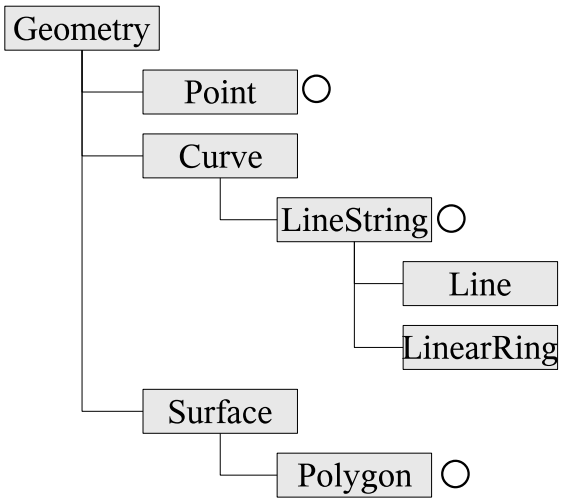


図 4.1: ジオメトリ階層

```
<misp:Filter>
  <misp:And>
    <misp:Within>
      <misp:PropertyName>location</misp:PropertyName>
      <gml:Polygon srsName='urn:epsg:v6.1:coordinateReferenceSystem:4326'>
        <gml:outerBoundaryIs>
          <gml:LinearRing>
            <gml:coordinates>-98.3,23.2, ...</gml:coordinates>
          </gml:LinearRing>
        </gml:outerBoundaryIs>
      </gml:Polygon>
    </gml:Within>
    <misp:PropertyIsBetween>
      <misp:PropertyName>DEPTH</misp:PropertyName>
      <misp:LowerBoundary>400</misp:LowerBoundary>
      <misp:UpperBoundary>800</misp:UpperBoundary>
    </misp:PropertyIsBetween>
  </misp:And>
</misp: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○印のジオメトリが使用可能である。(図 4.1)

4.5 時刻表現について

時刻表現には ISO8601 の dateTime 時刻表現 [7][16] を用いる。

dateTime	年月日時分秒	*YYYY-MM-DD*T*hh*:*mm*:*ssTZD*	2004-06-02T16:27:15Z

真中の 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。*TZD* は UTC からの時差を “±hh:mm” 形式で表した物か、米軍規格で UTC からの時差が 0 である事を示す “Z” である。日本標準時を使用する場合は “+09:00” となる。これは正確には ISO8601 ではなく、ISO8601 のサブセットである W3C の規格 “Date and Time Formats” [16] である。ISO8601 はその歴史的な理由から過去に存在したさまざまな時間の表記法を許容するが、すべてに対応する事は事実上困難である。そこで本規格では、機械的処理が簡便な W3C の時間表記規格を採用する。

関連図書

[1] W3C Note 11. Soap messages with attachments. http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211, Dec. 2000.

[2] The Internet Engineering Task Force RFC 959. Ftp (file transfer protocol). http://www.ietf.org/rfc/rfc959.txt.

[3] World Wide Web consortium (W3C). Extensible markup language (xml). http://www.w3.org/XML/.

[4] World Wide Web consortium (W3C). Soap version 1.2. http://www.w3.org/TR/2001/WD-soap12-20010709/, 2001.

[5] The Internet Engineering Task Force. Mime (multipurpose internet mail extension) (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.

[6] The Internet Engineering Task Force. Smtp (simple mail transfer protocol) rfc 2821. http://www.ietf.org/rfc/rfc2821.txt.

[7] International Organization for Standardization. *ISO 8601:2000 : Data elements and interchange formats – Information interchange – Representation of dates and times*, 2 edition, Jun. 2004. http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=2% 6780.

[8] Open GIS Consotium, Inc. *Filter Encoding Implementation Specification (OGC 02-059)*, ver. 1.0.0 edition, Sep. 2001. http://www.opengeospatial.org/docs/02-059.pdf.

[9] Open GIS Consotium, Inc. *Web Feature Service Implementation Specification (OGC 02-058)*, ver. 1.0.0 edition, May. 2002. https://portal.opengeospatial.org/files/?artifact_id=7176.

[10] Open GIS Consotium, Inc. *OpenGIS Geography Markup Language (GML) Implementation Specification (OGC-02-023r4)*, ver. 3.00 edition, Jan. 2003. http://www.opengis.org/docs/02-023r4.pdf.

[11] OpenGIS. Opengis simple features specification for sql revision 1.1. http://www.opengeospatial.org/docs/99-049.pdf.

[12] DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION September. Rfc 791 - internet protocol. ftp://ftp.rfc-editor.org/in-notes/rfc791.txt, Sep. 1981.

[13] DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION September. Rfc 793 - transmission control protocol. ftp://ftp.rfc-editor.org/in-notes/rfc793.txt, 1981.

[14] Latest soap versions. http://www.w3.org/TR/soap/.

[15] The Internet Society. Hypertext transfer protocol – http/1.1 [rfc2616]. http://www.ietf.org/rfc/rfc2616.txt, 1999.

[16] W3C. Date and time formats　w3c　iso 8601. http://www.w3.org/TR/NOTE-datetime.

付 録A　用語集

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。

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。

3 **名前空間接頭辞**:
XML Information Set(http://www.w3.org/TR/xml-infoset/)

4 **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

5 **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)

6 **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 3x8=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。

付 録B　変更履歴

- **ver.1.00.038s**　[2009-09-15:H.Shimora]
  - GetFeature のデフォルトモード normal を features に変更した。
- **ver.1.00.037s**　[2009-03-11:H.Shimora]
  - Query 中の SortBy の項目を追加した。
  - GetFeature の属性 maxFeatures, startPosition を追加した。
  - Query の属性 transformType を追加した。
- **ver.1.00.036s**　[2008-12-26:H.Shimora]
  - RegisterCoordinateSystemTransformation の項を追加した。それにともない GetFeature の srsName の指定を追加した。
- **ver.1.00.035s**　[2008-10-27:H.Shimora]
  - 例の中のタグ内で misp: のプレフィクスがが抜けていた部分を修正した。読み難かった部分に少しの空白を追加した。
- **ver.1.00.034s**　[2008-10-03:H.Shimora]
  - Delete、Update の mode の説明を追加した。
- **ver.1.00.033s**　[2008-10-03:H.Shimora]
  - 文章が途中で途切れていたのを修正した。
- **ver.1.00.032s**　[2008-08-20:H.Shimora]
  - 例中のスペルミスを修正した。
- **ver.1.00.031n**　[2006-12-03:I.Noda]
  - GetFeature の Filter の記述の細かい間違いなどを修正。
- **ver.1.00.030s**　[2006-10-16:H.Shimora]

- ドキュメントの細かな加筆。RegisterFeature が WFS ではなく MISP の拡張であることを明記。

- **ver.1.00.029n** [2006-10-16:I.Noda]
  - 組み込み属性の `_udpateTime_` と `_createTime_` を `_update_time_` と `_create_time_` に変更。

- **ver.1.00.028s** [2006-08-29:H.Shimora]
  - 仕様を大幅に変更し、現状扱われている仕様と整合させた。具体的には、SOAP の HTTP バインディング部分と、GML 被覆部分を削除した。その他、細かな修正を多数行った。

- **ver.1.00.027n** [2006-08-29:I.Noda]
  - システム予約属性の prefix 記述の修正。

- **ver.1.00.026n** [2006-08-24:H.Yokota]
  - schema 定義の制約事項等を加筆。

- **ver.1.00.025y** [2006-08-24:H.Yokota]
  - ISO8601 形式に関する解説、間違いの修正、既約違反の指摘を追加。
  - 図の修正。

- **ver.1.00.024n** [2006-08-21:I.Noda]
  - GetCapabilities の 各 feature の MostRecentTransactionID に関する記述の追加。

- **ver.1.00.023y** [2006-08-17:H.Yokota]
  - ISO8601 形式に関する解説を追加。

- **ver.1.00.022n** [2006-08-17:I.Noda]
  - ResponseStatus を各レスポンスに追加。加。
  - Transaction のレスポンスに Transaction Id に関する記述を追加。
  - _transaction_id_ へのアクセスに関して記述。

- **ver.1.00.021y** [2006-08-17:H.Yokota]
  - サーバの更新に伴い GetFeature コマンドによる検索の拡張オプション “modifyFlag” は無くなったのでこれに関する解説を削除。

- **ver.1.00.020y** [2006-08-09:H.Yokota]

  - GetFeature コマンドによる検索の拡張オプション “modifyFlag” に関する解説を追加。

- **ver.1.00.019s** [2006-06-12:H.Shimora]
  - 防災科学技術研究所のロゴを変更。

- **ver.1.00.018s** [2006-06-09:H.Shimora]
  - MISP のタグにネームスペースを追加。
  - RegisterFeatureType、DescribeFeatureType でのスキーマに targetNamespace、misp:id を追加。

- **ver.1.00.017y** [2006-04-12:H.Yokota]
  - 崩れた図を全体的に補正する。

- **ver.1.00.016n** [2006-04-12:I.Noda]
  - Filter で利用できる論理オペレータの説明で、True、 False を追加。

- **ver.1.00.015y** [2006-04-11:H.Yokota]
  - 画像ファイルを読み込む方法を psfig.sty から graphicx.sty に変更し、それに併せて読み込みコマンドを変更。

- **ver.1.00.014s** [2006-01-06:H.Shimora]
  - RegisterFeatureType と Transaction の返答を SUCCESS/FAILURE で統一した。

- **ver.1.00.013n** [2006-01-05:I.Noda]
  - 存在しない URI などの修正。
  - typo など。

- **ver.1.00.012s** [2005-12-27:H.Shimora]
  - typo の修正。(MIME タイプの記述中)

- **ver.1.00.011n** [2005-12-22:I.Noda]
  - `_modifyTime_` を `_updateTime_` へ。

- **ver.1.00.010n** [2005-12-21:I.Noda]
  - プロトコルの返答を、すべて ...Response に変更。SOAP の標準的な返答にあわせるため。
  - ‘ _’ (アンダースコア) で始まる要素名、属性名を予約語として指定。

- **ver.1.00.009n** [2005-11-02:I.Noda]
  - 観測データの UML を修正。gggd を ddt に。
- **ver.1.00.007n** [2005-09-29:I.Noda]
  - 細かい修正
- **ver.1.00.006n** [2005-08-26:I.Noda]
  - 観測データの UML クラス図追加。
  - input を include に変更。
  - 観測データの例の説明追加。
  - gml:validTime の書き方変更。
- **ver.1.00.005n** [2005-08-22:I.Noda]
  - myProg を変更、周囲を囲むようにした。行間も詰める。このため、myProg 中で framebox を使う際にちょっと不具合が生じる結果になった。要検討。
  - 不足していた style file をパッケージに入れる。
  - appendix の順序を変更。
  - 図の不具合修正。
- **ver.1.00.004n** [2005-08-22:I.Noda]
  - version.tex を changes.tex に集約。
  - 地物データの標準テンプレートについて、featureTmpl.tex を Appendix に追加。
  - 上記にともない、時間モデルの説明をその部分に追加。本文中の時間モデルとの整合をとる必要あり。
  - Filter に関する制約の部分で、繰り返し要素についての記述を修正。
- **ver.1.00.003n** [2005-08-20:I.Noda]
  - 字句修正。
  - 添付ファイルの効用についての記述を修正。
  - 時間指定なしを、0000-00-00T00:00:00 の代りに null 値に。
  - Pin Point 時間も最小時間単位の幅を持つことにする。
  - メモ多数。
- **ver.1.00.002n** [2005-08-19:I.Noda]

  字句修正。

- **ver.1.00.001n** [2005-08-19:I.Noda]
  MS-Word 版から LaTeX 版へ変換。
- **ver.1.00.000** [2005-08-18:I.Noda]
  MS-Word 版を version 1.00.000 とする。