OpenGIS Consortium Discussion Paper 01-035

" Geoparser Service Draft Candidate Implementation Specification 0.7.1"

This paper presents a discussion of technology issues considered in a Special Interest Group of the Open GIS Consortium Technical Committee. The content of this paper is presented to create discussion in the geospatial information industry on this topic; the content of this paper is not to be considered an adopted specification of any kind. This paper does not represent the official position of the Open GIS Consortium nor of the OGC Technical Committee.

TITLE: Geoparser Service Specification

DOCUMENT: OGC- 01-035

VERSION 0.7.1

DATE: *27 March, 2001*

TYPE: OGC-IP Draft Candidate Specification, Discussion Paper

This version:

<http://feature.opengis.org/members/archive/arch01/01-035.pdf>

Previous version:

00-055(Geoparse)r5

Editor:

Jeff Lansing, Polexis, jeff@polexis.com

Contributors:

Rob Atkinson, Social Change Online, rob@socialchange.net.au

Graeme Cox, tSA Consulting

John Davidson, OGC IPTeam Member, georef@erols.com
Harry Niedzwiadek, OGC IPTeam Architect, harry1@erols.com
Carl Reed, OGC IPTeam Member, creediii@mindpsring.com

Abstract

This document describes the OGC interfaces for a network-accessible Geoparser Service.

Status of this document

Issue Status Summary Table

Issue Name	Status
Dependencies on WFS	Open
DescribeFeatureType is required	Closed jl, 30 Jan 01.
LockFeature interface is required (GFSPP-300101)	Closed jl, 30 Jan 01.
Need use case (GFSPP-300101-2)	Closed jl, hn, 30 Mar 01
Vocabulary operations	Open
Replace references to WFS	Open
GetCapabilities	Open
Schema repository. (GFSPP-1)	Closed jl, mt, 16 Feb 01
Update GetFeature for XML Schema	Closed jl, 2 Feb 01
GetFeature request update	Closed jl, 2 Feb 01
GetFeature must support GML output	Closed jl, 29 Mar 01
Align GetFeature with XML Schema	Closed jl, 2 Feb 01
Update GetFeature response schema.	Closed jl, 2 Feb 01
Exceptions not defined	Open
Reference the XML Schema document at the schema repository	Closed jl, 29 Mar 01
Vocabulary metadata is embedded into DescribeFeatureType response	Closed jl, 2 Feb 01
Transaction Interface	Closed jl, 2 Feb 01
Update with new examples	Closed jl, 30 Mar 01
Reference to HTTP get	Open

Issue Name: [Dependencies on Web Feature Server specification. (GFSPP, 30 Jan 01)]

Issue Description: [Some of the interfaces used in this specification are identical to interfaces defined in the Web Feature Server specification. There is an outstanding action to separately

GetFeature, DescribeFeatureType, LockFeature and Transaction. The most likely place to capture this is in the upcoming General Services Model (GSM) spec work planned for OGC Web Services 2001. No further action should be taken until that initiative is underway.]

Resolution: [Open. Resolve during OGC Web Services 2001. Then replace WFS references with BSM/GSM references. (HAN, 3/27/01)]]

This document is both a Draft Candidate Implementation Specification (resulting from the Geospatial Fusion Services Testbed, and the Geospatial Fusion Services Pilot Project, of the OGC Interoperability Program) and a Discussion Paper (OGC Technical Committee) for the OGC Geoparser Service. It represents "work in progress", and should be treated accordingly. This version of the specification supersedes all previous Geoparser Service documents.

The revision history is summarized below:

- 0.7.2 – Final draft of Discussion Paper resulting from completion of the GFS Pilot Project.
- 0.7.1 - Second draft of Discussion Paper.
- 0.6 – First draft of Discussion Paper.
- 0.4 gc– Updated version after the GFS Testbed, with no changes to v0.5 interfaces
- 0.5 – Version implemented for the GFS Testbed, completed on November 17, 2000. This version was harmonized with the WFS spec.
- Earlier versions were draft candidate specifications leading up to 0.4 gc.

Editorial Comments

ED: [Editorial Notes are inserted in RED, as displayed here, wherever needed.]

Issues

All issues, and applicable resolutions, are documented inline. Please use the format below as a guideline for documenting issues.

Issue Name: [Issue Name in RED, BLUE, or GREEN based upon criticality of the issue, with Red being highest priority, and Green the lowest. (Initials, Date)]

Issue Description: [Issue Description.]

Resolution: [Insert Resolution Details and History. (Initials, Date)]

Table of contents

1. Introduction

A **Geoparser Service** is a network-accessible service that focuses on the geoparsing and marking of free text messages using a vocabulary, such as place names for Canada, which is possibly specified by the user. Output from a Geoparser Service is a collection of features that identifies words and phrases in the original text resource. The returned collection of features is suitable for subsequent processing, such as user-controlled geocoding. It is anticipated that this Geoparser Service will have a significant impact on the ability of applications to share multiple distributed interoperable Geoparser Services and offer a useful service to the geospatial community.

The minimal function of a Geoparser Service is to find instances of interesting (to the user) words and phrases, such as a place name or time of day, in a text resource, and to return an indication of where those instances occur in the resource. How the Geoparser Service goes about finding the kinds of words or phrases that it targets is not necessarily limited to vocabulary matching or partial matching. There are also geoparsing methods that are based on discovering the patterns of usage of the target kinds of words or phrases, and then looking for those patterns. Nonetheless, any method the Geoparser Service uses can be helped by controlling the vocabulary or vocabularies that the Geoparser Service has access to. In effect, the geoparsing methodology is a function of the Geoparser Service, and this not included in the scope of this document. The only requirement is that the capability be effectively described within the Capabilities Statement, which is attainable through a Geoparser Service interface (<getCapabilities>).

A slight extension of the minimal Geoparser Service functionality just described would be the provision of additional information about the particular strings that have been identified. For example, in the case of place names, it might be able to say what the source of those names was, such as a given gazetteer. In this way a geocoder that has to use the geoparsed strings would have a way of knowing where to look for more information about them.

A further extension of functionality would be for the Geoparser Service to provide additional information directly. This would be the case for a Geoparser Service that has information available when it is built. This information may be just a list of interesting strings. But for a spatially aware Geoparser Service (one that is able to filter its output based on a bounding box), the additional information could include location-based information for each string. It could also include facts about associated addresses, population, containing geographic region, associated hyperlinks, or almost anything else. As specified in this document, a Geoparser Service may support any of these levels of functionality. In a sense then, a Geoparser Service could provide a basic geocoding capability.

In order to provide control over the vocabulary, or vocabularies that the Geoparser Service has available to it, the following scheme has been adopted. The various kinds of geoparsing tasks, such as recognizing place names, recognizing dates, etc., are treated as if they were all cases of retrieving various feature types. In this way a Geoparser Service which recognizes place names is treated as a service which supports retrieval of place name type features; a Geoparser Service which recognizes dates/times is treated as a service which supports retrieval of date/time type features; and so on. The capabilities statement will identify the Feature Types the service returns. However, a Geoparser Service is not a store for the features (of the type) that it returns; in fact, these come from the text resource, which the Geoparser Service parses.

There are therefore three interfaces required for geoparsing:

1. GetCapabilities interface
2. GetFeature interface
3. DescribeFeatureType

Issue Name: [DescribeFeatureType is required. (GFSPP, 30 Jan 01)]
Issue Description: [DescribeFeatureType is a required interface.]
Resolution: [Closed. Added the DescribeFeatureType interface. (GFSPP, 30 Jan 01)]

And two optional interfaces:

4. Transaction interface – this allows the client to tell the Geoparser Service what vocabulary to use. This implies a stateful operation.
5. LockFeature interface – this allows a stateful operation to function correctly with multiple users.

Issue Name: [LockFeature interface is required. (GFSPP-300101) (GFSPP, 30 Jan 01)]

Issue Description: [Can't have Transaction without LockFeature.]

Resolution: [Closed. Added the LockFeature interface (GFSPP, 30 Jan 01)]

This interface specification for Geoparser Service is harmonized with the syntax and semantics of other OGC interfaces, namely Web Feature Server, Gazetteer Service, and Geocoder Service, and it is intended to be compliant with the Basic Service Model.

1.1. Use Cases (Informative)

Issue Name: [Need use case. (GFSPP-300101-2) (GFSPP, 30 Jan 01)]

Issue Description: [Add use case.]

Resolution: [Closed. Use case provided. (HN, JL, 30 Mar 01)]

Use Case - Application of Vocabularies within the GFS Pilot Project (GFSPP)

The concept of a "*vocabulary*" was employed during the development of the Geoparser Draft Candidate Implementation Specification, under the Geospatial Fusion Services Testbed Project (2000). Within the context of this development, *vocabulary* has the following meaning:

> **Definition:** A named set of independent terms; a list of terms in string form that are order independent. The meanings of the terms and possible relationships between the terms are not explicitly represented in the *vocabulary*. Each *vocabulary* has a name (VocabularyKey), a character encoding (charset) and other possible attributes (e.g. bounding box). [Note: A Geoparser treats a vocabulary as a GML abstract feature.]
>
> **Application Overview:** The *vocabulary* represents a set of terms that describe a *domain of interest*. The scope of the domain depends on the application. For example, the *domain of interest* might be represented by a set of terms that consist of the landmarks in an area of interest, or a set of terms that characterize terrorist activities, or terms that an

analyst has compiled for a given problem set. Thus, vocabularies may be built for any application (*domain of interest*). The Geoparser Service presently accepts one or more vocabularies, which can be loaded and registered with the Geoparser at runtime, by a client application, or pre-loaded with a utility. When a given instance of a Geoparser has two or more registered vocabularies, each vocabulary is treated as a "sub-vocabulary". A client that requests service involving a multi-vocabulary Geoparser must stipulate which of the registered vocabularies apply in a given Geoparser run. Otherwise, all registered vocabularies will apply. The Geoparser employs the specified registered vocabulary(ies) during its parsing process, finding all *matching* terms in the text (or XML) document that is being parsed. For GFSPP, *matching* means finding terms in the text/XML document that exactly match terms in the vocabulary(ies).

Use Case for Runtime GFSPP. This use case involves a client application performing three basic functions involving a Geoparser and its associated vocabularies:

- **Register New Vocabulary**
- **Discover Registered Vocabularies**
- **Parse Text Document**

Register New Vocabulary. We start with a Geoparser service that has two registered "Reference Vocabularies[1]" that have been pre-loaded with a Utility. [Note: The Utility (notionally) has been used to seed the Geoparser with one Reference Vocabulary that has been extracted from a Gazetteer Service (i.e. Landmarks) and another that has been extracted from a Yellow Page Service (i.e. Hotels). The Utility employs the <transaction> interface of Geoparser to register both vocabularies.] In parallel, a user employing a runtime Application Client (App) has built their own vocabulary of terms, which they call MyVocabulary[2]. In this case, MyVocabulary consists of the names of certain people. The App allows the user to register this vocabulary with the Geoparser Service, using the Geoparser's <Transaction> interface.

Issue Name: [Vocabulary operations. (HN, 30 Mar 01)]

[1] A Reference Vocabulary is a well-known vocabulary from an authoritative source.

[2] The concept of "MyVocabulary" recognizes that users often have unique uses for vocabularies that they create, depending on their requirements for analysis.

Issue Description: [We need to address the basic issues of creating, managing and otherwise exploiting vocabularies throughout the OGC Web Services arena. As a general concept, the scope of vocabularies might broadly cover the use of terms/phrases, with their semantics, for all Web services. For example, the topic might generally include Web Service namespaces, Gazetteers, Type Dictionaries, Geocoding Types, Thesauri, etc., in addition to the general notion of vocabularies as used in Geoparser, which is very broad in application scope. Generally, a vocabulary will likely pertain to any well-known set of terms/concepts.

During GFSPP, we ascertained that "Vocabulary Manager Functions" were needed to deal with the requirements for vocabulary management and manipulation operations. The following requirements were determined by the team:

- Creation – tools to populate (seed) a vocabulary from any source, such as: a Gazetteer, .txt, or a .csv.
- AddTerm – tool to add a term to a known vocabulary (e.g., a term which may have been defined by a process, or interactively by a user).
- RemoveTerm – a filter operation that "deletes" select terms.
- EditTerm – tools to modify select terms (and their metadata), including the ability to set usage properties, like "turning off a term".
- Extend – tools to populate a known vocabulary with additional terms.
- Subset – tools to manage sub-vocabularies within a larger vocabulary (manage hierarchies).
- View – tools to search and browse the contents of a vocabulary.
- Administration – tools to manage and administer a vocabulary registry.

]

Resolution: [Open. (aaa, date)]

Discover Registered Vocabularies. The Geoparser Service now has three registered vocabularies: two Reference Vocabularies, Landmarks and Hotels, and one personal vocabulary, MyVocabulary. The user now wants to run the Geoparser. The user directs the App to bring up a Geoparser control panel. The user next selects a text document to parse. Now they have an opportunity to select the vocabularies they wish to employ in the pending Geoparser run. The App queries for the currently registered vocabularies, using the <DescribeFeatureType> interface. The App presents this information to the user and allows the user to select the desired vocabularies for the Geoparser run. Hotels and MyVocabulary are selected.

Parse Text Document. With the text document and vocabulary subset identified, the user now runs the Geoparser, employing the <GetFeature> interface. The Geoparser scans the text document for occurrences of the names of certain hotels (matching terms in the vocabulary: Hotel), and the names of certain people (matching terms in the vocabulary: MyVocabulary). The matched terms found in the text document are pinpointed in the <GetFeature> response. Finally, the App allows the user to examine the marked text document for correlations between certain people and certain hotels.

1.2. Architectural Constraints (Informative)

Web Feature Server compatibility

Issue Name: [Replace references to WFS. (GFSPP, 30 Jan 01)] **Issue Description:** [Replace WFS references with the appropriate OGC BSM/GSM references, when they become available.] **Resolution:** replace WFS references with “OGC Interfaces” (Jeff L, 30 Jan 01)]

By harmonizing the interfaces for a Geoparser Service to be syntactically and semantically similar with the interfaces for WFS, it is possible to then use a WFS, and its associated databases of features, as the underlying technology for a general purpose Geoparser Service. However, this is not meant to constrain implementations of Geoparser Services to be dependent on WFS capabilities.

As with the WFS, Gazetteer and Geocoder Services, the response to a <GetCapabilities> request for a Geoparser Service contains the list of supported well-known Feature Types defined for the service. Client applications must use the <DescribeFeatureType> interface to discover the specific set of properties for these Feature Types. Geoparser Services require the <GetFeature> interface to return a specific, specialized set of feature instances whose types are listed in the <GetCapabilities> response and whose properties conform to the schema returned by <DescribeFeatureType>.

Five WFS interfaces apply to this version of Geoparser Service:

- GetCapabilities
- DescribeFeatureType
- GetFeature
- Transaction
- LockFeature

Query constraints

A Geoparser Service may be able to support spatial and thematic search constraints, per the OGC Filter Encoding Spec. Such a capability depends on the information, which is available in the vocabularies, which are loaded in the Geoparser Service.

Terms addressable via a URI

The Geoparser Service should allow feature relationships to be traversed easily, via OGC Geolinks (see the GML 2.0 spec for a definition of Geolink). This implies that a Geoparser Service response should have resolvable URI references for related features. This mandates support for a URI interface for features, as well as creating local references by instantiating the features in the response document.

2. Interface Definitions (Normative)

The following interfaces are required for a Geoparser Service:

- GetCapabilities
- GetFeature
- DescribeFeatureType

The following interfaces are optional for a Geoparser Service:

- Transaction
- LockFeature

Appendix A has examples of requests and responses for the GetFeature interface. Appendix B has HTTP GET request semantics for all interfaces.

2.1. Interface *<GetCapabilities>*

Request

This interface is used to request a capabilities document from a Geoparser Service. (It is expected that a primary common reference for capabilities will be the Basic Service Model.) It is assumed that the client has referenced a services registry to get the location of the service, and now needs to discover its capabilities.

Issue Name: [GetCapabilities. (GFSPP, 30 Jan 01)] **Issue Description:** [Extending GetCapabilities from OGC Interface is not possible until the OGC GetCapabilities Interface is defined in XML Schema and is generally accepted by the OGC IP & TC. This work is still being investigated.] **Resolution:** [We must wait until the OGC GetCapabilities (BSM) is finalized. For the moment this interface, including the response, remains as XML DTD. (GFSPP, 30 Jan 01)]

Request Syntax

The way that a client determines what capabilities a Geoparser Service can provide is through the GetCapabilities interface. The GetCapabilities request is shown below (as defined in the Basic Services Model).

```
<!ELEMENT GetCapabilities EMPTY>
     <!ATTLIST GetCapabilities version CDATA #REQUIRED>
```

Response (Return Values)

The GetCapabilities response is derived from the BSM specification.

Exceptions

None.

2.2. Interface *<GetFeature>*

Issue Name: [Schema repository. **GFSPP-1, (1/9/01)].**

Issue Description: Doesn't follow WFS GetFeature request. It extends it.

Actions: [(GFSPP 1/9/01)] A WFS/BSM Interface Schema document must first exist in a public place like opengis.net. This interface schema must import the WFS/BSM schema document and derive from it. Alternatively, change the generic interface schema to be more flexible/robust.

a) **Publish a Filter schema document (.xsd). (Who: Ron Lake/Serge. When: 16January01. Complete)**

b) **GetFeature schema document published (i.e., update WFS spec and post the .xsd file) (All interface schemas to be published). Who: Ron Lake. When: 16 January 01.**

c) **get Geoparser schema published (i.e., add to Geoparser spec and post the .xsd file. (All interfaces in Geoparser spec). Who: Jeff Lansing. When: 16 January 01.**

Resolution: [Closed. (jl, mt, 16 Feb 01)]

Issue Name: [Update GetFeature for XML Schema. (GFSPP, 30 Jan 01)]

Issue Description: [GetFeature extends BSM GetFeature.]

Resolution: [Closed. (jl, 2 Feb 01)]

The GetFeature interface is where parsing of the resource takes place. There are a number of different cases that can occur. The input resource can either be included within the body of the request as Contents, or it can be referenced by the request as a URL. The format of the input resource can be either plain text, or it can be XML, or it can be some other format or formats not treated in this version of this specification (such as one or more versions of HTML). The output result of geoparsing is a list of pointers to matched portions of the input.

Notes:

(a) Included xml must be in a CDATA section because it may have its own DTD defining entities that will need to be resolved.

(b) The xml referenced by a URL must be well formed, any DTD's or schemas referenced by the xml must be accessible, and all entities must be resolvable. Parsing will be with respect to resolved entities.

(c) If the input type is xml, pointers will be offsets into the string-value of the root element of the xml.

(d) Pointers will use range elements, which are described in this specification.

(e) A text resource included in the body of the request is assumed to be XML encoded, and will be decoded before parsing.

Request

Issue Name: [GetFeature request update. (GFSPP, 30 Jan 01)]

Issue Description: [This needs to be updated to reflect the XML Schema implementation. So far, changes extend OGC GetFeature interface to include resource element (source document) and add enumeration case of output formats (i.e., XML, GML Feature Collection)]

Resolution: [Closed. (jl, 2 Feb 01)]

Issue Name: [GetFeature must support GML output. (GFSPP, 30 Jan 01)]

Issue Description: [It is mandatory that Geoparser implementations minimally support output format == GML Feature Collection]

Resolution: [Closed. (jl, 29 Mar 01)]

The GetFeature interface responds to requests that conform to the following schema. For interoperability, any Geoparser implementation must be able to respond with GML format output results.

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema targetNamespace="http://www.opengis.net/gp"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:gp="http://www.opengis.net/gp" elementFormDefault="qualified">

  <!-- ================================================================

       Includes and Imports

    ================================================================
-->

  <xsd:import namespace="http://www.opengis.net/wfs"
schemaLocation="..\wfs\GetFeatureRequest.xsd"/>

  <!-- ================================================================

       Global elements and attributes

    ================================================================
-->

  <!-- ================================================================

  Root element

  ============================================================ -->

  <xsd:element name="GetFeature" type="gp:GetFeatureType"/>

  <!-- ================================================================

  Types

  ============================================================ -->

  <xsd:complexType name="GetFeatureType">

    <xsd:complexContent>

      <xsd:extension base="wfs:GetFeatureType">

        <xsd:sequence>
```

```
          <xsd:element name="Resource" type="gp:ResourceType"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ResourceType">
    <xsd:sequence>
      <xsd:element ref="gp:Contents" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>Must be present if href is
missing.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
    <xsd:attribute name="href" type="xsd:uriReference"
use="optional">
      <xsd:annotation>
        <xsd:documentation>Must be present if TermNames is
missing.</xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="mime">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="text/plain"/>
          <xsd:enumeration value="text/xml"/>
          <xsd:enumeration value="application/xml"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
  <xsd:element name="Contents" type="xsd:string">
    <xsd:annotation>
      <xsd:documentation>If mime is text/xml or application/xml then
the value of Contents must be wrapped in a CDATA
section.</xsd:documentation>
    </xsd:annotation>
  </xsd:element>
```

```
</xsd:schema>
```

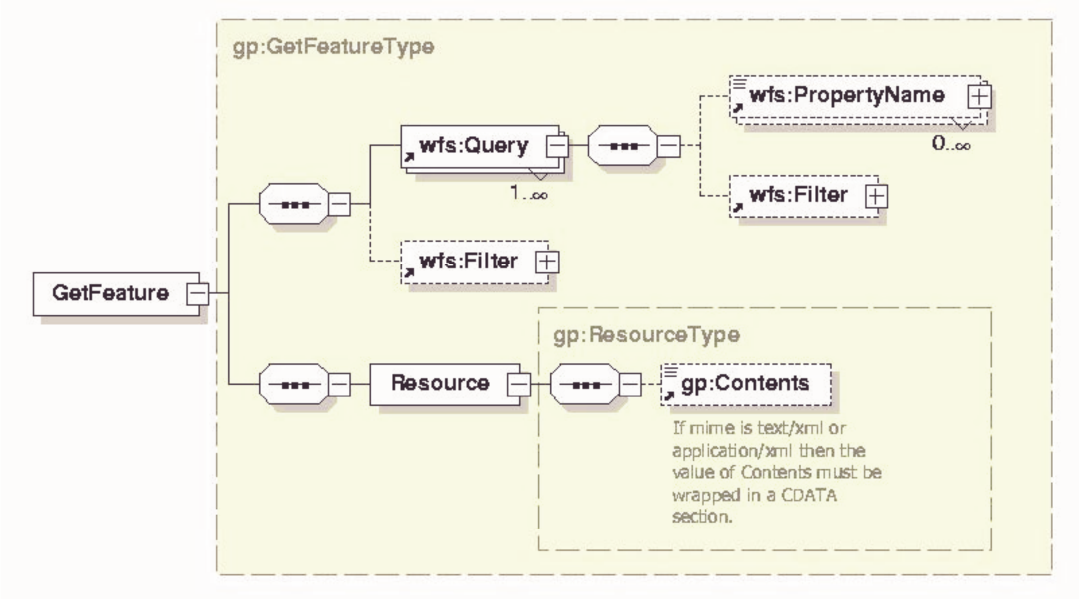
Issue Name: [Align GetFeature with XML Schema. (GFSPP, 30 Jan 01)]

Issue Description: [description of these parameters should be aligned with those defined in the XML Schema above. Focus on those parameters that extend the BSM GetFeature interface.

Resolution: [Closed. (jl, 2 Feb 01)]

Type Definitions

The following diagram shows the elements that are defined here.



GetFeature

The GetFeature element can be used to package one or more query descriptions into a single request. The results of all queries packaged in a GetFeature request are concatenated to produce the result set.

Query

Each individual query packaged in a request is defined using the Query element. This element defines which feature type to query, what properties to retrieve, and what constraints to apply to those properties.

Filter

The Filter element is used to constrain a query. Both spatial and/or non-spatial constraints can be specified as described in the OGC Filter Encoding Specification. For example, since a Vocabulary is a GML abstract feature, it can have a bounding box that can be referenced in a spatial constraint.

Resource

The format and source of the input resource are controlled by the Resource child of the GetFeature element. The “mime” attribute of the Resource element describes the media type of the resource, and the alternation between the “href” attribute and the Contents child of the Resource element controls, where the Geoparser should look for the input resource. Note that if both the “href” attribute and the Contents child are present, then the “href” attribute is ignored. (If neither is present, an exception occurs.)

PropertyName

Used to enumerate the feature properties or attributes that should be selected. If no tags are specified then all properties should be fetched.

Attributes

outputFormat

This attribute defines the format to use to generate the result set. The default value is GML2 indicating that GML Profile 2 shall be used. Vendor specific formats, as declared in the capabilities document, are also possible.

handle

The “handle” attribute is used in exception processing to identify the part of the request that lead to the exception.

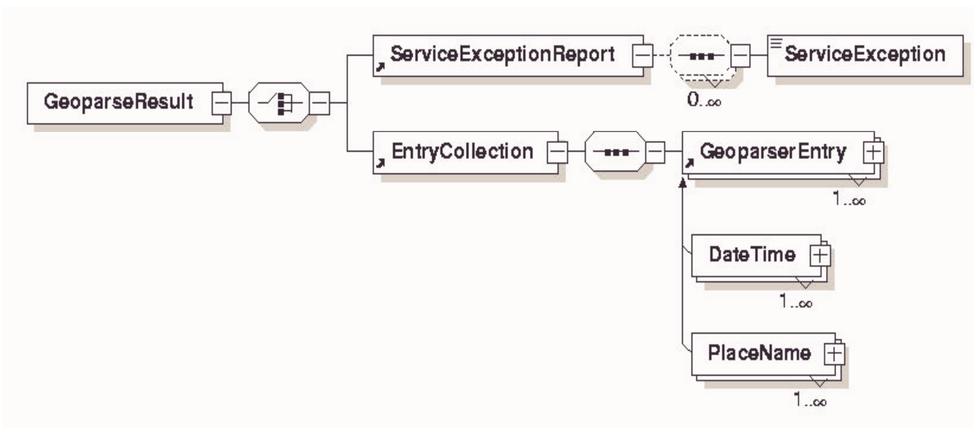
maxFeatures

This attribute can be used to limit the number of features that a <getFeature> request retrieves. Once the *maxFeatures* limit is reached, the result set is truncated at that point.

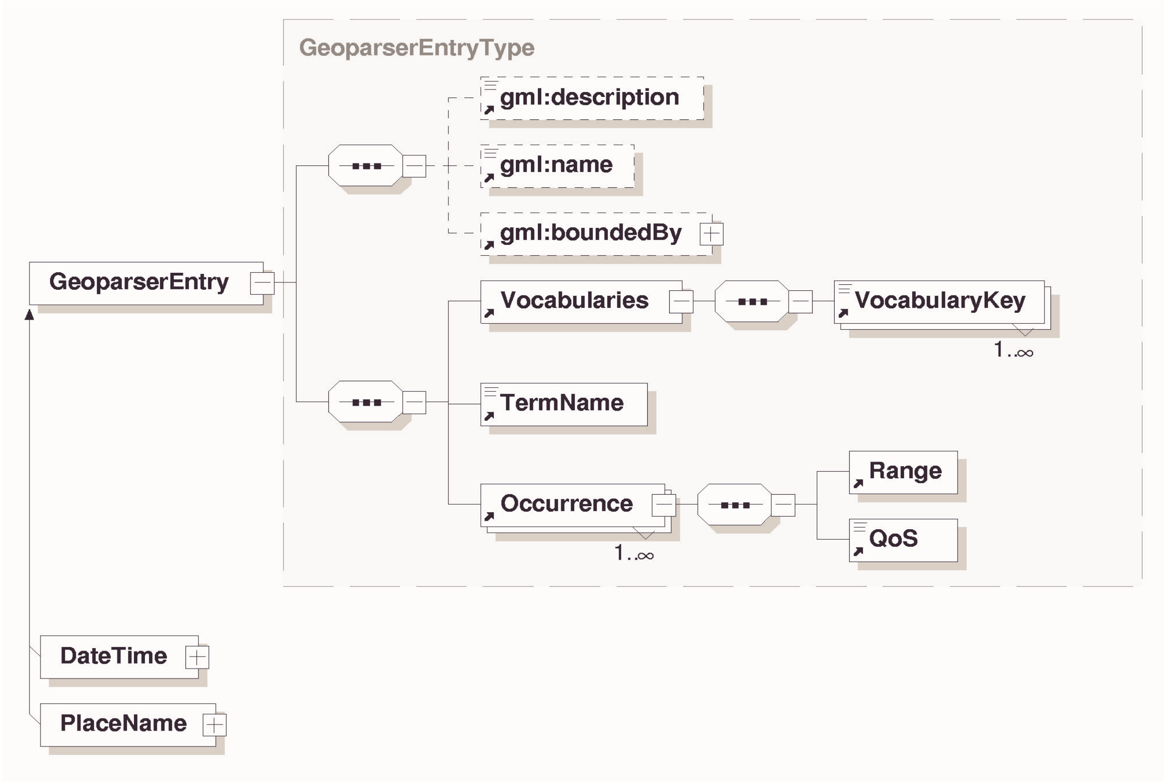
Response (Return Values)

Issue Name: [Update GetFeature response schema. (GFSPP, 30 Jan 01)] **Issue Description:** [] **Resolution:** [Closed. (jl, 2 Feb 01)]

The general form of the GeoparseResult which is returned from a GetFeature request is shown in the following diagram.



An important component of the output result is a set of pointers to the matched portions of the input. Previous versions of this specification erroneously attempted to use XPointer to reference the matched portions of the input. This is actually not feasible, and has been dropped. Hence compatibility with previous versions is maintained by using a new Range element for the pointers, as shown in the following diagram.



Future versions of this specification may define more GeoparserEntry feature types.

The response from the GetFeature request conforms to the following schema:

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema targetNamespace="http://www.opengis.net/gp"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
xmlns="http://www.opengis.net/gp"
xmlns:gml="http://www.opengis.net/gml" elementFormDefault="qualified">

  <!-- ============================================================

       Includes and Imports

    ================================================================
-->

  <xsd:import namespace="http://www.opengis.net/gml"
schemaLocation="C:\GFSPilot\docs\geoparser\schemas\gml\feature.xsd"/>

  <xsd:include schemaLocation="../gp/Vocabulary.xsd"/>

  <!-- ============================================================

       Global elements and attributes

    ================================================================
-->
```

```
  <xsd:element name="PlaceName" type="GeoparserEntryType"
substitutionGroup="GeoparserEntry"/>

  <xsd:element name="DateTime" type="GeoparserEntryType"
substitutionGroup="GeoparserEntry"/>

  <xsd:element name="GeoparserEntry" type="GeoparserEntryType"/>

  <!-- ================================================================

  Root element

  ================================================================ -->

  <xsd:element name="GeoparseResult">

    <xsd:complexType>

      <xsd:choice>

        <xsd:element ref="ServiceExceptionReport"/>

        <xsd:element ref="EntryCollection"/>

      </xsd:choice>

    </xsd:complexType>

  </xsd:element>

  <xsd:element name="ServiceExceptionReport">

    <xsd:complexType>

      <xsd:sequence minOccurs="0" maxOccurs="unbounded">

        <xsd:element name="ServiceException" type="xsd:string"/>

      </xsd:sequence>

      <xsd:attribute name="version" type="xsd:string" use="required"/>

    </xsd:complexType>

  </xsd:element>

  <xsd:element name="EntryCollection">

    <xsd:complexType>

      <xsd:sequence>

        <xsd:element ref="GeoparserEntry" maxOccurs="unbounded"/>

      </xsd:sequence>

    </xsd:complexType>

  </xsd:element>

  <!-- ================================================================

  Types

  ================================================================ -->

  <xsd:complexType name="GeoparserEntryType">

    <xsd:complexContent>

      <xsd:extension base="gml:AbstractFeatureType">
```

```
        <xsd:sequence>
          <xsd:element ref="Vocabularies"/>
          <xsd:element ref="TermName"/>
          <xsd:element name="Value" type="xsd:string"/>
          <xsd:element ref="Occurrence" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="Occurrence">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Range"/>
        <xsd:element ref="QoS"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="QoS" type="xsd:nonNegativeInteger"/>
  <xsd:element name="Range">
    <xsd:complexType>
      <xsd:attribute name="href" type="xsd:uriReference"
use="optional"/>
      <xsd:attribute name="start" type="xsd:nonNegativeInteger"
use="required"/>
      <xsd:attribute name="end" type="xsd:positiveInteger"
use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Vocabularies">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="VocabularyKey" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Exceptions

Issue Name: [Exceptions not defined. (GFSPP, 30 Jan 01)] **Issue Description:** [There is no common way to handle exceptions yet in OGC Web services land….] **Resolution:** [Open. Jeff Lansing to list exceptions, when general OGC approach to exceptions is determined (BSM/GSM issue). (GFSPP, 30 Jan 01)]

ExceptionName	Description ....

2.3. Interface *<DescribeFeatureType>*

The DescribeFeatureType interface is identical to the same interface in the WFS. Because the Transaction interface allows feature types to change dynamically, for example, by adding a new vocabulary to the PlaceName feature type, the DescribeFeatureType interface has to reflect the current state of the Geoparser Service.

Request

The function of the <describeFeatureType> interface is to provide a client the means to request a definition of any feature type that a particular Geoparser Service can provide. The description that is generated will define how a Geoparser Service expects a client application to express the state of a feature to be created. In other words, the result of a <describeFeatureType> request is an XML feature schema definition.

A request is composed of a list of names of feature types that are to be described.

The DescribeFeatureType interface is defined by the DescribeFeatureType schema at www.opengis.net/namespaces/wfs/DescribeFeatureTypeRequest.xsd.

repository. (GFSPP, 30 Jan 01)]

Issue Description: [inline the schema or reference it?]

Resolution:[Closed. (jl, 29 Mar 01)]

Response (Return Values)

The response to the DescribeFeatureType request conforms to the schema for XML-Schema. Here we provide an example of what such a schema might look like, for a Geoparser Service that supports place names.

Issue Name: [Vocabulary metadata is embedded into the DescribeFeatureType response. (GFSPP, 30 Jan 01)]

Issue Description: [Document that this is what is going on here or, alternatively, define a new interface to return vocabulary info. Seems to be general agreement that the approach shown below is correct.]

Resolution: [Closed. (jl, 2 Feb 01)]

```
<?xml version="1.0" encoding="UTF-8"?>

<xsd:schema targetNamespace="http://www.opengis.net/gp"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
xmlns="http://www.opengis.net/gp"
xmlns:gml="http://www.opengis.net/gml"
elementFormDefault="qualified">

  <!-- ==============================================================

       Includes and Imports


============================================================== -
->

  <xsd:import namespace="http://www.opengis.net/gml"
schemaLocation="..\gml\feature.xsd"/>

  <!-- ==============================================================

       Root element


============================================================== -
->

  <xsd:element name="PlaceName" type="PlaceNameType"/>
```

```
  <!-- ======================================================================
       Notice that the current state of the Geoparser can be
described by enumerating

  its values, as follows:

====================================================================== -
->
  <xsd:element name="VocabularyKey">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="Oslo Street Names"/>
        <xsd:enumeration value="Bergen Street Names"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:element>
  <!-- ======================================================================
  Types
  ================================================================= -->
  <xsd:complexType name="PlaceNameType">
    <xsd:complexContent>
      <xsd:extension base="gml:AbstractFeatureType">
        <xsd:sequence>
          <xsd:element ref="Vocabularies"/>
          <xsd:element ref="TermName"/>
          <xsd:element ref="Occurrence" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:element name="Occurrence">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Range"/>
        <xsd:element ref="QoS"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

```
  <xsd:element name="QoS" type="xsd:nonNegativeInteger"/>
  <xsd:element name="Range">
    <xsd:complexType>
      <xsd:attribute name="href" type="xsd:uriReference"
use="optional"/>
      <xsd:attribute name="start" type="xsd:nonNegativeInteger"
use="required"/>
      <xsd:attribute name="end" type="xsd:positiveInteger"
use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Vocabularies">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Vocabulary" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Vocabulary">
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="gml:AbstractFeatureType">
          <xsd:sequence>
            <xsd:element ref="VocabularyKey"/>
            <xsd:element ref="TermNames" minOccurs="0"/>
          </xsd:sequence>
          <xsd:attribute name="charset" type="xsd:string"
use="required"/>
          <xsd:attribute name="href" type="xsd:uriReference"
use="optional"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="TermNames">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="TermName" maxOccurs="unbounded"/>
```

```
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="TermName" type="xsd:string"/>
</xsd:schema>
```

Notice that the particular schema returned in response to a DescribeFeatureType request can be more specific than the part of the schema for the GetFeature request that describes that same feature.

Using a schema such as the above, a client could construct a filter using

```
<PropertyName>
   Occurrence.QoS
 </PropertyName>
```

such that the Geoparser Service would only return perfect matches. Or a client could construct a filter that used the canonical format of the date/time entries, together with a specified temporal interval, in order to limit the results from a subsequent call to GetFeature to lie within a certain time interval of interest.

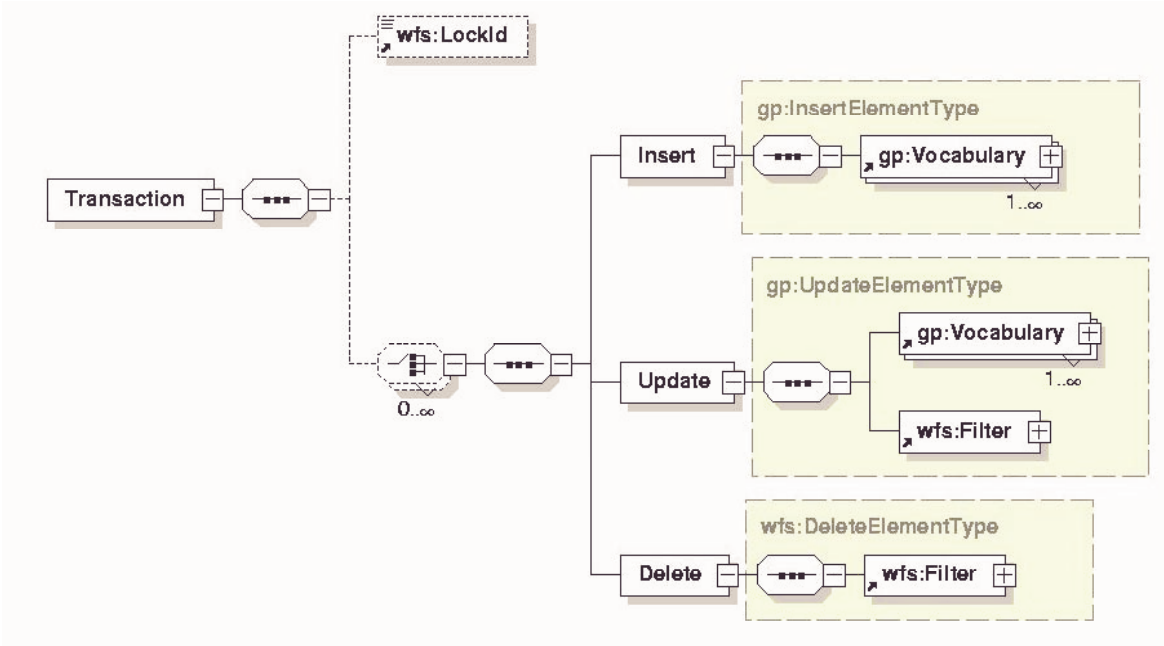
Exceptions

TBD.

ExceptionName	Description ....

2.4. Interface *<Transaction>*

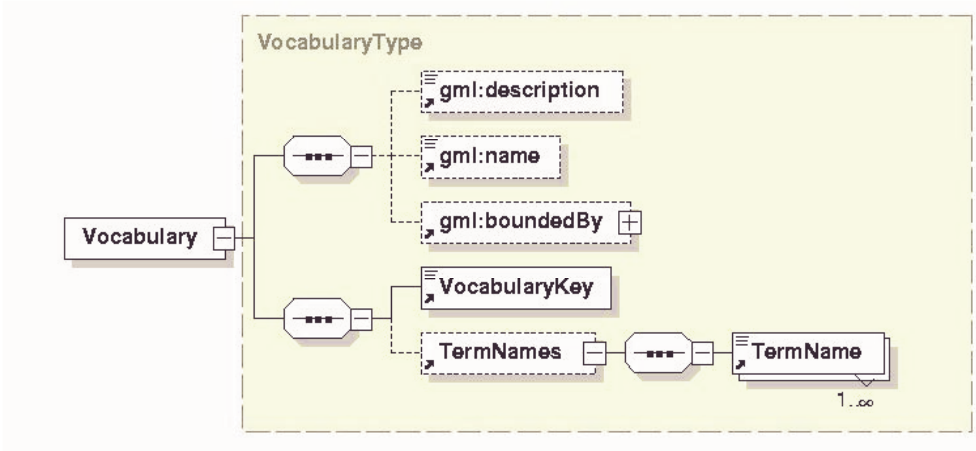
This interface is optional. The Transaction interface is a slight restriction of the same Transaction interface for WFS.



As shown in the above diagram, the restriction is that instead of allowing any feature type to be inserted, updated, or deleted, only Vocabularies that derive from AbstractFeatureType, may be used. As with the standard OGC Transaction interface, the LockId is required for updates and deletes.

A Vocabulary feature may be sent in a Transaction request in either of two forms, a “push” form, or a “pull” form. In the “push” form, all of the terms in the vocabulary that are involved in the transaction are included in the request. In the “pull” form the terms are not part of the request, but are instead referenced by a URL in the request. Thus in the following diagram, the TermNames element is seen to be optional; it is present in the request in the “push” case and it is located at the other end of the URL (and absent in the request) in the “pull” case.

(Note that in the “pull” case a client would not be able to acquire a lock on the vocabulary until after the “pull” operation was completed.)



Examples of the two forms of Vocabulary for a Transaction request are as follows. Note that the vocabulary key is always part of the request, so that any possible conflict conditions can be detected at the time of the request.

A “push” example would look like this:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Vocabulary xmlns="http://www.opengis.net/gp"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/gp ../gp/Vocabulary.xsd" charset="latin1">
  <VocabularyKey>West Bank Names</VocabularyKey>
  <TermNames>
    <TermName>'Abd al Qadir al Kaylani</TermName>
    <!-- thousands of these -->
    <TermName>Zuhur al Balqa'</TermName>
  </TermNames>
</Vocabulary>
```

and the corresponding “pull” example would looks like:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Vocabulary xmlns="http://www.opengis.net/gp"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/gp ../gp/Vocabulary.xsd"
href="zeuss/data/westbank/terms.xml" charset="latin1">
  <VocabularyKey>West Bank Names</VocabularyKey>
</Vocabulary>
```

These examples conform to the following schema for a Geoparser vocabulary:

```xml
<xsd:element name="TermName" type="xsd:string"/>
  <!-- ============================================================
  Root element
  ============================================================ -->
  <xsd:element name="Vocabulary" type="VocabularyType"/>
```

```xml
<!-- ==============================================================
Types
=============================================================== -->
<xsd:complexType name="VocabularyType">
  <xsd:complexContent>
    <xsd:extension base="gml:AbstractFeatureType">
      <xsd:sequence>
        <xsd:element ref="VocabularyKey"/>
        <xsd:element ref="TermNames" minOccurs="0">
          <xsd:annotation>
            <xsd:documentation>Must be present if href is missing.</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="charset" type="xsd:string" use="required">
        <xsd:annotation>
          <xsd:documentation>Must be a valid IANA charset parameter value.</xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
      <xsd:attribute name="href" type="xsd:uriReference" use="optional">
        <xsd:annotation>
          <xsd:documentation>Must be present if TermNames is missing.</xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:element name="TermNames">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="TermName" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```

The VocabularyKey element in the Vocabulary is a character string, which serves as the unique identifier of the vocabulary. If a vocabulary with a given key has already been inserted, the next insert will fail.

The Geoparser Service must expect that the content of the TermName elements will be XML encoded, which means that at a minimum ampersands will be encoded as “&”, and less than characters will be encoded as “<”. Additionally characters greater than 0x7f could be encoded using “&#” as a prefix and “;” as a suffix, depending on the “charset” attribute, as described in Character Model for the World Wide Web 1.0.

Issue Name: **[**Transaction Interface**, (GC, 11/27/00)].**

Interface, the following would need to be changed to fit into my views:

Transaction Request

- **Name**
- Abstract
- Keywords
- Domain
- Language
- Special Conditions
- Location (e.g., URL)
- **Format**
  - Field List (Must include mandatory fields: Unique Identifier, Human Readable Feature …)
  - Field Separators (e.g., Tab …)
  - Hierarchy (Y/N)
    - Number of Levels (e.g., 4)
    - Nomenclature (e.g., XML tree, Relational …)
  - Vocabulary
    - Size
      - Rows (e.g., 100,000)
      - Columns (e.g., 10)
      - Max Bytes per Row (e.g., 10)
    - Language (e.g., English, Norwegian, Unicode …)
- Authorisation
  - User Name and Password (?)
  - Syntax Name of Gazetteer (As supplied by the authorising authority)
- Administrative Details
  - Contact
    - **Person**
    - Organisation
    - Email (as the Response will be sent to this address)
    - …
  - Load Date (lodged)

Expiration Date (e.g., 2000/11/28)

Action: Rob and Jeff to define vocabulary schema. Start with minimum/required set of elements. Alternatively, work on making interface more extensible to handle arbitrary vocabulary schemas. Want to handle more complicated vocabularies without breaking the current working vocabulary schema.

Resolution: [Closed. (jl, 2 Feb 01)]

Request

The schema for the Transaction request describes how to provide a list of terms for the Geoparser Service to use in its processing of GetFeature requests.

Request Syntax

The request form is the following:

```
<?xml version="1.0" encoding="UTF-8"?>

<schema targetNamespace="http://www.opengis.net/gp"
xmlns:wfs="http://www.opengis.net/wfs" xmlns:gp="http://www.opengis.net/gp
xmlns="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified"

  <!-- ============================================================

       Includes and Imports

    ================================================================ --

  <import namespace="http://www.opengis.net/wfs"
schemaLocation="http://www.opengis.net/namespaces/wfs/TransactionRequest.x
d"

  <include schemaLocation="../gp/Vocabulary.xsd"/>

  <!-- ============================================================

       Global elements and attributes

    ================================================================ --

  <!-- ============================================================

  Root element

  ============================================================== -->

  <element name="Transaction">

    <complexType>

      <complexContent>
```

```
      <restriction base="wfs:TransactionType">
        <sequence>
          <element ref="wfs:LockId" minOccurs="0"/>
          <choice minOccurs="0" maxOccurs="unbounded">
            <sequence>
              <element name="Insert" type="gp:InsertElementType"/>
              <element name="Update" type="gp:UpdateElementType"/>
              <element name="Delete" type="wfs:DeleteElementType"/>
            </sequence>
          </choice>
        </sequence>
      </restriction>
    </complexContent>
  </complexType>
</element>
<!-- ==============================================================
Types
=============================================================== -->
<complexType name="InsertElementType">
  <complexContent>
    <restriction base="wfs:InsertElementType">
      <sequence>
        <element ref="gp:Vocabulary" maxOccurs="unbounded"/>
      </sequence>
    </restriction>
  </complexContent>
</complexType>
<complexType name="UpdateElementType">
  <complexContent>
    <restriction base="wfs:UpdateElementType">
      <sequence>
        <element ref="gp:Vocabulary" maxOccurs="unbounded"/>
        <element ref="wfs:Filter"/>
      </sequence>
    </restriction>
  </complexContent>
```

```
  </complexType>
</schema>
```

Response (Return Values)

The response to the Transaction request indicates either the success or failure of the transaction, as described in the Transaction interface (see WFS spec). The VocabularyKey element in the Vocabulary serves as the unique identifier of the vocabulary. If a vocabulary with a given key has already been inserted, the next insert will fail.

Exceptions

TBD.

ExceptionName	Description ....

2.4. Interface *<LockFeature>*

This interface is required if the Transaction interface supports an Update or a Delete operation. The Lock Feature interface is the same the OGC standard interface. By acquiring a lock on a vocabulary, a client can insure that another client cannot update or delete that vocabulary until such time as the lock expires or has been released.

References

For the latest version of any OGC specification, please consult the list of OpenGIS Specifications available at http://www.opengis.org.

Normative references

1. **Geography Markup Language (GML) v2.0.** Available online at: **<**http://feature.opengis.org/members/archive/arch01/01-029.pdf**>**
2. **Web Feature Server Specification.** OGC Draft Candidate Implementation Specification available online at: **<**http://feature.opengis.org/members/archive/arch01/01-023r1.pdf**>**
3. **Filter Encoding Specification.** OGC Draft Candidate Implementation Specification available online at: <http://feature.opengis.org/members/archive/arch01/01-033.pdf>
4. **Gazetteer Service Specification.** OGC Draft Candidate Implementation Specification available online at: <http://feature.opengis.org/members/archive/arch01/01-036.pdf>
5. **Geocoder Service Specification**. OGC Draft Candidate Implementation Specification available online at: <http://feature.opengis.org/members/archive/arch01/01-026r1.pdf>
6. **Location Organizer Folder**. OGC Draft Candidate Implementation Specification available online at: <http://feature.opengis.org/members/archive/arch01/01-037.pdf>
7. Whiteside, A, and J. Bobbit. 2000. ***Recommended Definition Data for Coordinate Reference Systems and Coordinate Transformations***. OGC Project Document 00-040r7.
8. **Basic Service Model V0.0.8.** OGC Discussion Paper available online at: <http://feature.opengis.org/members/archive/arch01/01-022r1.pdf>

9. **W3C Candidate Recommendation for XML Schema (24 October 2000).** Available online at: <http://www.w3.org/TR/2000/CR-xmlschema-0-20001024/
10. **Character Model for the World Wide Web 1.0**, available online at: http://www.w3.org/TR/charmod

Glossary

Term	Definition A

Appendix A: Examples of Geoparser Service GetFeature Requests and Responses

Issue Name: [Add new examples. (GFSPP, 30 Jan 01)] **Issue Description:** [Jeff has 8 new examples] **Resolution:** [Closed. (jl. 30 Mar 01)]

Some simple examples of using the elements and attributes, which make up a GetFeature request follow.

The cases where the Resource element has an “href” attribute are based on the following two sample cables. The text cable (called msg1.txt) is this:

```
RETURN-PATH: <WATCHER@OBSERVANT.ORG>
RECIEVED: FROM MARS(DEIMOS.PLANETS.ORG [192.178.24.11]) BY
MAIL.ARG.ORG(8.9.3/8.8.7) WITH SMTP ID TAAA00811; WED, 15 NOV 2000 10:12:41
-0000
MESSAGE-ID: <4.2.2.20000204154155.00A53980@192.178.24.12>
X-SENDER: HENRYFARMER@192.178.24.12 (UNVERIFIED)
X-MAILER: LOTUS NOTES RELEASE 5.0.1B SEPTEMBER 30, 1999
X-PRIORITY: 1(HIGH)
DATE: WED, 15 NOV 2000 10:12:41 -0000
TO: TM@SECURITY.ORG
FROM: HENRY FARMER <HENRYFARMER@192.178.24.12>
SUBJECT: O.S. MOVEMENT UPDATE
MIME-VERSION: 1.0
CONTENT-TYPE: TEXT/PLAIN;CHARSET="US-ASCII"; FORMAT-FLOWED

O.S. HAS BEEN SIGHTED IN LONDON. THE FIRST SIGHTING WAS AT 10:05 AM TUESDAY,
NOVEMBER 10TH AT TRAFAGLAR SQUARE. HE WAS OBSERVED MOVING SLOWLY DOWN STRAND IN
THE DIRECTION OF COVENT GARDEN. AT 11:15 am HE WAS OBSERVED GOING INTO A SMALL
PUB IN COVENT GARDEN CALLED "THE LAMB AND FLAG."

---
```

The xml version of the cable (called msg1a.xml) is this:


```
<CABLE>
 <RETURN-PATH>&lt;WATCHER@OBSERVANT.ORG&gt;</RETURN-PATH>
 <RECIEVED>
     <FROM>MARS(DEIMOS.PLANETS.ORG [192.178.24.11])</FROM>
     <BY>MAIL.ARG.ORG(8.9.3/8.8.7) WITH SMTP ID TAAA00811</BY>
     <ON>WED, 15 NOV 2000 10:12:41 -0000</ON>
```

```
</RECIEVED>
<MESSAGE-ID>4.2.2.20000204154155.00A53980@192.178.24.12</MESSAGE-ID>
<X-SENDER>HENRYFARMER@192.178.24.12 (UNVERIFIED)</X-SENDER>
<X-MAILER>LOTUS NOTES RELEASE 5.0.1B SEPTEMBER 30, 1999</X-MAILER>
<X-PRIORITY>1(HIGH)</X-PRIORITY>
<DATE>WED, 15 NOV 2000 10:12:41 -0000</DATE>
<TO>TM@SECURITY.ORG</TO>
<FROM>HENRY FARMER &lt;HENRYFARMER@192.178.24.12&gt;</FROM>
<SUBJECT>O.S. MOVEMENT UPDATE</SUBJECT>
<MIME VERSION="1.0" CONTENT-TYPE="TEXT/PLAIN" CHARSET="US-ASCII"
FORMAT="FLOWED"/>
<MESSAGE>
O.S. HAS BEEN SIGHTED IN LONDON. THE FIRST SIGHTING WAS AT 10:05 AM TUESDAY,
NOVEMBER 10TH AT TRAFAGLAR SQUARE. HE WAS OBSERVED MOVING SLOWLY DOWN
STRAND IN
THE DIRECTION OF COVENT GARDEN. AT 11:15 am HE WAS OBSERVED GOING INTO A SMALL
PUB IN COVENT GARDEN CALLED "THE <bold>LAMB</bold> <italic>AND</italic> FLAG".
</MESSAGE>
</CABLE>
```

Note that “<” and “>” in the text portions of the message have been escaped to “<” and “>”. Standard XML parsing of this resource will convert them back to their original form.

Example 1 shows a case where the resource is a plain text file referenced by a URL. The request would take this form:

```
<?xml version="1.0" encoding="UTF-8"?>
<GetFeature xmlns="http://www.opengis.net/gp" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/gp ../gp/GetFeatureRequest.xsd
                    http://www.opengis.net/wfs ../wfs/GetFeatureRequest.xsd"
wfs:outputFormat="GML2">
 <wfs:Query wfs:typeName="PlaceName"/>
 <wfs:Query wfs:typeName="DateTime"/>
 <Resource href="http://localhost:8100/geoparser/cables/msg1.txt" mime="text/plain"/>
</GetFeature>
```

And the response to such a request would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<GeoparseResult xmlns="http://www.opengis.net/gp" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-
instance" xsi:schemaLocation="http://www.opengis.net/gp ../gp/GeoparseResult.xsd">
 <EntryCollection>
  <PlaceName>
   <Vocabularies>
    <VocabularyKey>Testbed Place Names</VocabularyKey>
   </Vocabularies>
   <TermName>LONDON</TermName>
   <Occurrence>
    <Range href="http://localhost:8100/geoparser/cables/msg1.txt" start="616" end="622"/>
    <QoS>100</QoS>
   </Occurrence>
  </PlaceName>
  <PlaceName>
   <Vocabularies>
    <VocabularyKey>Testbed Place Names</VocabularyKey>
   </Vocabularies>
   <TermName>TRAFAGLAR SQUARE</TermName>
   <Occurrence>
    <Range href="http://localhost:8100/geoparser/cables/msg1.txt" start="687" end="703"/>
```

```
      <QoS>100</QoS>
    </Occurrence>
  </PlaceName>
  <PlaceName>
    <Vocabularies>
      <VocabularyKey>Testbed Place Names</VocabularyKey>
    </Vocabularies>
    <TermName>STRAND</TermName>
    <Occurrence>
      <Range href="http://localhost:8100/geoparser/cables/msg1.txt" start="740" end="746"/>
      <QoS>100</QoS>
    </Occurrence>
  </PlaceName>
  <PlaceName>
    <Vocabularies>
      <VocabularyKey>Testbed Place Names</VocabularyKey>
    </Vocabularies>
    <TermName>COVENT GARDEN</TermName>
    <Occurrence>
      <Range href="http://localhost:8100/geoparser/cables/msg1.txt" start="769" end="782"/>
      <QoS>100</QoS>
    </Occurrence>
    <Occurrence>
      <Range href="http://localhost:8100/geoparser/cables/msg1.txt" start="840" end="853"/>
      <QoS>100</QoS>
    </Occurrence>
  </PlaceName>
  <PlaceName>
    <Vocabularies>
      <VocabularyKey>Testbed Place Names</VocabularyKey>
    </Vocabularies>
    <TermName>THE LAMB AND FLAG</TermName>
    <Occurrence>
      <Range href="http://localhost:8100/geoparser/cables/msg1.txt" start="862" end="879"/>
      <QoS>100</QoS>
    </Occurrence>
  </PlaceName>
  <DateTime>
    <Vocabularies>
      <VocabularyKey>Default</VocabularyKey>
    </Vocabularies>
    <TermName>2000-11-15T10:12:41</TermName>
    <Occurrence>
      <Range href="http://localhost:8100/geoparser/cables/msg1.txt" start="154" end="174"/>
      <QoS>100</QoS>
    </Occurrence>
    <Occurrence>
      <Range href="http://localhost:8100/geoparser/cables/msg1.txt" start="382" end="402"/>
      <QoS>100</QoS>
    </Occurrence>
  </DateTime>
  <DateTime>
    <Vocabularies>
      <VocabularyKey>Default</VocabularyKey>
    </Vocabularies>
    <TermName>1999-09-30T</TermName>
    <Occurrence>
      <Range href="http://localhost:8100/geoparser/cables/msg1.txt" start="330" end="348"/>
      <QoS>100</QoS>
    </Occurrence>
  </DateTime>
  <DateTime>
    <Vocabularies>
      <VocabularyKey>Default</VocabularyKey>
    </Vocabularies>
    <TermName>--11-10T10:05</TermName>
```

```xml
    <Occurrence>
      <Range href="http://localhost:8100/geoparser/cables/msg1.txt" start="650" end="683"/>
      <QoS>100</QoS>
    </Occurrence>
  </DateTime>
  <DateTime>
    <Vocabularies>
      <VocabularyKey>Default</VocabularyKey>
    </Vocabularies>
    <TermName>----T11:15</TermName>
    <Occurrence>
```


Example 2 shows a case where the resource is an xml file referenced by a URL. The request would take this form:


```xml
<?xml version="1.0" encoding="UTF-8"?>
<GetFeature xmlns="http://www.opengis.net/gp" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/gp ../gp/GetFeatureRequest.xsd
                    http://www.opengis.net/wfs ../wfs/GetFeatureRequest.xsd"
wfs:outputFormat="GML2">
 <wfs:Query wfs:typeName="PlaceName"/>
 <wfs:Query wfs:typeName="DateTime"/>
 <Resource href="http://localhost:8100/geoparser/cables/msg1a.xml" mime="text/xml"/>
</GetFeature>
```


And the response to such a request would look like this:


```xml
<?xml version="1.0" encoding="UTF-8"?>
<GeoparseResult xmlns="http://www.opengis.net/gp" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-
instance" xsi:schemaLocation="http://www.opengis.net/gp ../gp/GeoparseResult.xsd">
 <EntryCollection>
  <PlaceName>
    <Vocabularies>
      <VocabularyKey>Testbed Place Names</VocabularyKey>
    </Vocabularies>
    <TermName>LONDON</TermName>
    <Occurrence>
      <Range href="http://localhost:8100/geoparser/cables/msg1a.xml" start="423" end="429"/>
      <QoS>100</QoS>
    </Occurrence>
  </PlaceName>
  <PlaceName>
    <Vocabularies>
      <VocabularyKey>Testbed Place Names</VocabularyKey>
    </Vocabularies>
    <TermName>TRAFAGLAR SQUARE</TermName>
    <Occurrence>
      <Range href="http://localhost:8100/geoparser/cables/msg1a.xml" start="493" end="509"/>
      <QoS>100</QoS>
    </Occurrence>
  </PlaceName>
  <PlaceName>
    <Vocabularies>
      <VocabularyKey>Testbed Place Names</VocabularyKey>
    </Vocabularies>
    <TermName>STRAND</TermName>
    <Occurrence>
      <Range href="http://localhost:8100/geoparser/cables/msg1a.xml" start="546" end="552"/>
      <QoS>100</QoS>
    </Occurrence>
  </PlaceName>
```

```xml
<PlaceName>
  <Vocabularies>
    <VocabularyKey>Testbed Place Names</VocabularyKey>
  </Vocabularies>
  <TermName>COVENT GARDEN</TermName>
  <Occurrence>
    <Range href="http://localhost:8100/geoparser/cables/msg1a.xml" start="574" end="587"/>
    <QoS>100</QoS>
  </Occurrence>
  <Occurrence>
    <Range href="http://localhost:8100/geoparser/cables/msg1a.xml" start="644" end="657"/>
    <QoS>100</QoS>
  </Occurrence>
</PlaceName>
<PlaceName>
  <Vocabularies>
    <VocabularyKey>Testbed Place Names</VocabularyKey>
  </Vocabularies>
  <TermName>THE LAMB AND FLAG</TermName>
  <Occurrence>
    <Range href="http://localhost:8100/geoparser/cables/msg1a.xml" start="666" end="683"/>
    <QoS>100</QoS>
  </Occurrence>
</PlaceName>
<DateTime>
  <Vocabularies>
    <VocabularyKey>Default</VocabularyKey>
  </Vocabularies>
  <TermName>2000-11-15T10:12:41</TermName>
  <Occurrence>
    <Range href="http://localhost:8100/geoparser/cables/msg1a.xml" start="121" end="141"/>
    <QoS>100</QoS>
  </Occurrence>
  <Occurrence>
    <Range href="http://localhost:8100/geoparser/cables/msg1a.xml" start="291" end="311"/>
    <QoS>100</QoS>
  </Occurrence>
</DateTime>
<DateTime>
  <Vocabularies>
    <VocabularyKey>Default</VocabularyKey>
  </Vocabularies>
  <TermName>1999-09-30T</TermName>
  <Occurrence>
    <Range href="http://localhost:8100/geoparser/cables/msg1a.xml" start="259" end="277"/>
    <QoS>100</QoS>
  </Occurrence>
</DateTime>
<DateTime>
  <Vocabularies>
    <VocabularyKey>Default</VocabularyKey>
  </Vocabularies>
  <TermName>--11-10T10:05</TermName>
  <Occurrence>
    <Range href="http://localhost:8100/geoparser/cables/msg1a.xml" start="457" end="489"/>
    <QoS>100</QoS>
  </Occurrence>
</DateTime>
<DateTime>
  <Vocabularies>
    <VocabularyKey>Default</VocabularyKey>
  </Vocabularies>
  <TermName>----T11:15</TermName>
  <Occurrence>
    <Range href="http://localhost:8100/geoparser/cables/msg1a.xml" start="592" end="600"/>
    <QoS>100</QoS>
```

```
      </Occurrence>
    </DateTime>
  </EntryCollection>
</GeoparseResult>
```

Example 3 shows a case where the resource is a plain text file embedded in the request. The request would take this form:


```
<?xml version="1.0" encoding="UTF-8"?>
<GetFeature xmlns="http://www.opengis.net/gp"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/gp ../gp/GetFeatureRequest.xsd
                    http://www.opengis.net/wfs ../wfs/GetFeatureRequest.xsd"
wfs:outputFormat="GML2">
  <wfs:Query wfs:typeName="PlaceName"/>
  <wfs:Query wfs:typeName="DateTime"/>
  <Resource mime="text/plain">
    <Contents>MEET ME AT THE LAMB AND FLAG AT 10:30 PM.  P.S. NOT THE LAMB AND FLAG IN
OXFORD, THE ONE NEAR COVENT GARDEN.</Contents>
  </Resource>
</GetFeature>
```


And the response to such a request would look like this:


```
<?xml version="1.0" encoding="UTF-8"?>
<GeoparseResult xmlns="http://www.opengis.net/gp" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-
instance" xsi:schemaLocation="http://www.opengis.net/gp ../gp/GeoparseResult.xsd">
  <EntryCollection>
    <PlaceName>
      <Vocabularies>
        <VocabularyKey>Testbed Place Names</VocabularyKey>
      </Vocabularies>
      <TermName>THE LAMB AND FLAG</TermName>
      <Occurrence>
        <Range start="11" end="28"/>
        <QoS>100</QoS>
      </Occurrence>
      <Occurrence>
        <Range start="52" end="69"/>
        <QoS>100</QoS>
      </Occurrence>
    </PlaceName>
    <PlaceName>
      <Vocabularies>
        <VocabularyKey>Testbed Place Names</VocabularyKey>
      </Vocabularies>
      <TermName>COVENT GARDEN</TermName>
      <Occurrence>
        <Range start="94" end="107"/>
        <QoS>100</QoS>
      </Occurrence>
    </PlaceName>
    <DateTime>
      <Vocabularies>
        <VocabularyKey>Default</VocabularyKey>
      </Vocabularies>
      <TermName>----T22:30</TermName>
      <Occurrence>
        <Range start="32" end="40"/>
```

```xml
      <QoS>100</QoS>
    </Occurrence>
  </DateTime>
 </EntryCollection>
</GeoparseResult>
```

The “start” and “end attributes are zero-based offsets into the content of an input resource which has type text/plain, and into the string-value of the root element of an input resource which has type text/xml or application/xml. In both cases the offsets are based on treating characters as 16-bit entities. The offsets are considered to point to the inter-character gaps; thus offset 0 points to the gap before the first character, offset 1 points to the gap between the first and second characters, etc.

The first match in this result has start offset 11; this points to the gap between the space before the first ‘THE’ in the input, and the ‘T’. Similarly, the end offset of 28 points to the gap between the ‘G’ of ‘FLAG’ and the space that follows it.

Example 4 shows a case where the resource is an xml file embedded in the request. The request would take this form:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<GetFeature xmlns="http://www.opengis.net/gp"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/gp ../gp/GetFeatureRequest.xsd
                    http://www.opengis.net/wfs ../wfs/GetFeatureRequest.xsd"
wfs:outputFormat="GML2">
 <wfs:Query wfs:typeName="PlaceName"/>
 <wfs:Query wfs:typeName="DateTime"/>
 <Resource mime="text/xml">
  <Contents><![CDATA[<note>MEET ME AT THE LAMB AND FLAG AT 10:30 PM.  P.S.
<bold>NOT</bold> THE LAMB AND FLAG IN OXFORD, THE ONE NEAR COVENT
GARDEN.</note>]]></Contents>
 </Resource>
</GetFeature>
```

And the response to such a request would look like this:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<GeoparseResult xmlns="http://www.opengis.net/gp" xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-
instance" xsi:schemaLocation="http://www.opengis.net/gp ../gp/GeoparseResult.xsd">
 <EntryCollection>
  <PlaceName>
   <Vocabularies>
    <VocabularyKey>Testbed Place Names</VocabularyKey>
   </Vocabularies>
   <TermName>THE LAMB AND FLAG</TermName>
   <Occurrence>
    <Range start="11" end="28"/>
    <QoS>100</QoS>
   </Occurrence>
   <Occurrence>
    <Range start="52" end="69"/>
```

```xml
      <QoS>100</QoS>
    </Occurrence>
   </PlaceName>
   <PlaceName>
    <Vocabularies>
      <VocabularyKey>Testbed Place Names</VocabularyKey>
    </Vocabularies>
    <TermName>COVENT GARDEN</TermName>
    <Occurrence>
      <Range start="94" end="107"/>
      <QoS>100</QoS>
    </Occurrence>
   </PlaceName>
   <DateTime>
    <Vocabularies>
      <VocabularyKey>Default</VocabularyKey>
    </Vocabularies>
    <TermName>----T22:30</TermName>
    <Occurrence>
      <Range start="32" end="40"/>
      <QoS>100</QoS>
    </Occurrence>
   </DateTime>
  </EntryCollection>
</GeoparseResult>
```


A variant of Example 4 would be to have internal general entities in the embedded xml; the following request would produce the same results as shown above.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<GetFeature xmlns="http://www.opengis.net/gp" xmlns:wfs="http://www.opengis.net/wfs"
xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/gp ../gp/GetFeatureRequest.xsd
        http://www.opengis.net/wfs ../wfs/GetFeatureRequest.xsd"
wfs:outputFormat="GML2">
  <wfs:Query wfs:typeName="PlaceName"/>
  <wfs:Query wfs:typeName="DateTime"/>
  <Resource mime="text/xml">
   <Contents><![CDATA[ <!DOCTYPE note [ <!ENTITY pub "LAMB AND FLAG"> ]>
                    <note>MEET ME AT THE &amp;pub; AT 10:30 PM.
      P.S. <bold>NOT</bold> THE &amp;pub; IN OXFORD, THE ONE NEAR COVENT
GARDEN.</note>]]>
   </Contents>
  </Resource>
</GetFeature>
```

Note that it is necessary to escape the embedded entity references because the initial parsing tries to expand them before they are defined.

Appendix B: HTTP GET Request Semantics

This section describes how to invoke the Geoparser Service interfaces using HTTP GET. This means that parameters consist of name-value pairs in the form of "name=value" and the pairs are separated by the ‘&’ character. The rules for URI encoding apply; for example, spaces must be encoded as ‘%20’s. These rules can be found at http://www.w3.org/TR/xptr#uri-escaping, for example.

Issue Name: [Reference to HTTP get. (GFSPP, 30 Jan 01)]

Issue Description: [We will reference the BSM/GSM spec, eventually.]

Resolution: [Open. (aaa,date)]

The following table describes the parameters required by all Geoparser Service requests:

URL Component	***Description***
http:/server_address/path	URL prefix of Geoparser Service.
VER	Request version.
REQUEST	Name of request. One of “DESCRIBEFEATURETYPE”, “TRANSACTION”, “GETFEATURE”, “GETCAPABILITIES”.
Additional parameters	As described in this section.

GETCAPABILITIES Interface

Request:

URL Component	***Description***
http:/server_address/path/	URL prefix of Geoparser Service.

VER=0.1.3	Request version. Required.
REQUEST= GETCAPABILITIES	Name of request. Required.

TRANSACTION Interface

Request:

URL Component	***Description***
http:/server_address/path/	URL prefix of Geoparser Service.
VER=0.1.3	Request version. Required.
REQUEST= TRANSACTION	Name of request. Required.
OPERATION=Operation.Name	The name of the operation. Must be one of “INSERT”, “UPDATE”, DELETE”, “LOCK”
TYPENAME=feature_type	The name of the feature type to operate on.
DESCRIPTION=Vocabulary.Description	Used with insert and update. This becomes the key to the vocabulary on insertion.
NAME=list_of_names	A comma delimited list of names to insert into, or merge with, a vocabulary.

DESCRIBEFEATURETYPE Interface

Request:

URL Component	***Description***
http:/server_address/path/	URL prefix of Geoparser Service.
VER=0.1.3	Request version. Required.
REQUEST=DESCRIBEFEATURETYPE	Name of request. Required.

TYPENAME=feature_type_list	A comma separated list of feature types to describe. Required.

GETFEATURE Interface

Request:

URL Component	***Description***
http:/server_address/path/	URL prefix of Geoparser Service.
VER=0.1.3	Request version. Required.
REQUEST=GETFEATURE	Name of request. Required.
RESOURCE=resource_url	URL location of the resource to be parsed.
MIME=mime_type	Mime type of the resource.
PROPERTYNAME=property_name_list	A list of properties must be specified for each feature type that is being queried. The list of property names is comma separated. Each individual list of properties is enclosed in parentheses. A '*' character can be used to fetch all properties. There is a 1:1 mapping between each element is a TYPENAME list and the PROPERTYNAME list. Required.
TYPENAME=feature_type_name	A list of feature type names to query. Optional. No default.
FILTER=uri_encoded_filter_xml_string	A filter specification describes a set of features to operate upon. The filter is defined above. The filter must be valid for all feature

	types specified in the TYPENAME parameter. Optional. No default. Prerequisite: TYPENAME parameter.

Using this draft specification, we have:

http://www.myfavoriteGeoparser Service.com/servlet/geoparse?VER=0.1.2&REQUEST=GETFEATURE&PROPERTYNAME=(*)&MIME=cable&RESOURCE=http://www.agency.gov/cables/cable1234.