OpenGIS Consortium Discussion Paper 01-026r1

" Geocoder Service Draft Candidate Implementation Specification 0.7.6"

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: Geocoder Service Specification

OGC DOCUMENT: OGC- 01-026r1

VERSION 0.7.6

DATE: *28 March, 2001*

TYPE: OGC-IP Draft Candidate Implementation Specification, Discussion Paper

This version:

http://feature.opengis.org/members/archive/arch01/01-026r1.pdf

Previous version:

http://feature.opengis.org/members/archive/arch01/01-026.pdf

Editor:

Serge Margoulies, IONIC Software, serge.margoulies@ionicsoft.com

Contributors:

Sandra Johnson, MapInfo, Sandra_Johnson@mapinfo.com
Rob Atkinson, Social Change Online, rob@socialchange.net.au
Bernard Snyers, IONIC Software, bernard.snyers@ionicsoft.com
John Davidson, OGC IP Team Architect, georef@erols.com
Harry Niedzwiadek, OGC IP Team Architect, harry1@erols.com
Carl Reed, OGC IP Team Architect, creediii@mindpsring.com
Ron Lake, Galdos, rlake@galdosinc.com
Jeff Lansing, DTAI, jeff@polexis.com
Marwa Mabrouk, ESRI , mmabrouk@esri.com
Benoit Borlee, IONIC Software, benoit.borlee@ionicsoft.com
David Warren, Cquay
David Danko, US NIMA
Paul Daisey, US Census

Abstract

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

Status of this document

This document is a draft candidate specification, Request for Comment (RFC), for OGC Geocoder Service. It represents “work in progress” and should be treated accordingly. This version of the specification supersedes all previous Geocoder Service documents.

The revision history is summarized below:

- 0.7.6 – Third draft candidate implementation specification, following GFSPP 2001. First revision of the TC Discussion Paper.
- 0.7.5 – Second draft candidate implementation specification.
- 0.7.41 –Updated with Sandra and John comments.
- 0.7.4 –Updated by Serge, following latest meetings in DC and understanding in GFSPP.
- 0.7.2 – Some editorial by David Warren.
- 0.7.1 – First draft of Discussion Paper.
- 0.6.2 – Version that included the combined specifications for Geocoder and Gazetteer Services, as implemented for the GFS Testbed, completed on November 17, 2000.
- Earlier versions were draft candidate specifications leading up to 0.6.2.

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 priority. (Initials, Date)]
>
> **Issue Description:** [Issue Description.]
>
> **Resolution:** [Insert Resolution Details and History. (Initials, Date)]

Table of contents

1. Introduction

A **Geocoder Service** is a network-accessible service that transforms a description of a feature location, such as a place name, street address or postal code, into a normalized description of the location, which includes a coordinate geometry. In other words, the Geocoder Service receives the description of a feature location as input and provides a normalized address with geometry as output.

The feature location descriptions are any words, codes or terms that describe the features, and that are well-known to the Geocoder Service, such as a street addressing or postal coding scheme. This service will determine the geometries for one or more features, given their associated well-known feature location descriptions, which are specified to the service at run-time, through a query.

The feature location descriptions spelled out in a geocoding query request may be determined in any of a number of ways, such as by user input in a form, by a remote application calling the Geocoder Service, or by a Geoparser Service that has processed a text resource, like an email message or news clipping, where the geoparser has pinpointed applicable feature location descriptions in the text resource.

The Geocoder Service defined herein assumes that text string inputs are in a suitable input format. This does not mean that the strings to be geocoded actually need to exist in the vocabulary associated with the service, or that they will not be further processed by the Geocoder Service, for example, to search for similar names. A Geocoder Service might optionally search for similar terms (synonyms), through an embedded thesauri or pattern matching capability.

Each instance of a Geocoder Service has an associated “vocabulary” of well-known location descriptions. For example, a Geocoder Service might only be applicable for geocoding street addresses in the United States, or for business specific geocoding, like IP addresses, grid names, etc.

A Geocoder Service returns a feature (with geometries and other properties) or a feature collection. These are the features that were specified through the request filter. The returned features are expressed in GML, which conforms to XML Schema, in a known OGC Spatial Reference System. The geometries are normalized using the GML types and the attributes are normalized using “Geocoding Types”, as defined in the geocoding Schema.

This specification facilitates interoperability by defining normalized Geocoding Types. These types allow strong typing of the Geocoding Service inputs and outputs. This guarantees, for example, that whatever the language or terms used by the geocoder, another application is able to semantically understand the result.

The Geocoder Service implements several capabilities, as summarized below and detailed in Section 2.

1. The ***GeocodingEntryType***. Defines the required and optional properties for the "feature" types supported by the Geocoder Service. These types may be actual feature types or they may pertain to the properties of features, e.g., one might wish to geocode a "street address", which is a property of a feature of type "street". Geocoder extends "GeocodingEntryType" to express the QoS (Quality of Service).

2. The **Quality of Service** (QoS) is returned for each feature to express the quality of geocoder results (e.g., what it found). For example, this can return a value between 0.0 and 1.0 to express how good the results match the request. This is useful to sort and rank the result, say, if one desires to return the five best results.

3. **Namespace**. The Geocoding Types are a list of available well-known types used in the normalization to ensure the semantic interoperability of the service. The types are defined in a XML Schema and have a preferred namespace named "geocoder: ". The Schema is available on the OGC Web site. To ensure interoperability, it is necessary to define the types by giving a name, a description (semantic), examples, and the type definition in XML Schema. For the geometry, the use of GML is mandatory.

4. The "**Type of Match Operation** " is a way to express what kind of geocoding the geocoder should perform. For example, if the user inputs a stream of features with multiple information types, the geocoding request should specify the nature of the type of geocoding that is desired (e.g. by IPAddress, not by ZIPCode, nor by some other geocoding capability that the Geocoder Service might be capable of performing). For some implementations, this will be implicit, and for others it will be explicit. There are several ways to do this, as we will see below, depending on the implementation.

5. Some **constraints** can be expressed in the user's request. The most common constraint is the "Exact" match, when the user expresses that the input parameters are known precisely and have to be matched exactly, or if he wants a "loose" or "fuzzy" search. Sometimes, the user wants to request "exact" match on a field, like PostalCode and a "loose" match on StreetName. Other constraints could be geographic, asking for a match inside a box. There are different ways to do this, as we will see below.

6. The OGC ***GetCapabilities*** Interface. This interface is the method for any OGC service to describe what it offers to a client. In this interface, it is used to expose supported `GeocodingEntryTypes` and other capabilities. This includes the definition of the geographic bounds for the geocoding area, the SRS' that is supported, and the specific geocoding capabilities.

7. The OGC ***DescribeFeatureType*** interface. This interface returns the schema for desired FeatureTypes, extending `GeocodingEntryType`.

8. The interfaces to support the **Geocoding Request**. There are two types of interfaces that directly support the geocoding process. Both interfaces solve the same problem with different levels of granularity or syntax, but generally, they have the same semantics. The geocoding process is always based on a database search and specialized algorithms to find the best match according to search parameters. The interfaces are:

    a) The OGC ***GetFeature*** interface**.** The GetFeature interface is the first method that a client can use to express a geocoding request. The feature types that can be geocoded can be discovered by the GetCapabilities / DescribeFeatureType mechanism. In this interface, the "type of match" operations are expressed by the Feature Type and the Filter.

    Here is an example to help understand the nature of the GetFeature interface: If there is a Geocoder Service exposing a Feature Type "USAddress", it is possible to write a filter for this Feature Type that looks like: "*where Street like '123 Main Street ' and City=' New York ' and ' Country=' US ' "*.
    The geocoder could find features like « North Main Street, 123 » and « S. Main Street - 123 » in New York, US, depending on the ability of the geocoder and the filter parameters.

    b) The ***GeocodeFeature*** interface. This interface provides the means to geocode a feature collection (one or more features) using a Geocoder Service. The schema for the feature collection is derived from the `GeocodingEntryType` schema. The service automatically updates the input feature collection with the results of geocoding operations, adding properties, such as the geometry. In this interface, the type of match operations are explicit in the parameter part of the request and are expressed as strings like: "Street" or "PostCode" or "Organization" or "Landmark", or some other specific type of match expressed in the capabilities.

The interfaces for both geocoding approaches are defined in detail in Section 2. Both approaches are equally valid, with the choice of interface depending on the user requirements, and possibly the underlying implementation.

We can envision that the first interface might be best suited for complex queries and geocoding from a Web page form. The second is designed to geocode a collection, like your customer database, in one simple call.

This specification is harmonized with the syntax and semantics of other OGC interfaces (Normative References), and it will be made compliant with the draft candidate implementation specification for Basic Service Model.

1.1. Use Cases (Informative)

There are many types of geocoding. These include the processes of matching and normalizing:

- Place name or landmark to geometry
- Street address to geometry
- Point, direction and distance to geometry
- Start point and distance along a linear feature to geometry
- IP address to geometry
- Phone number to geometry
- Business specific locations or vocabularies or map numbers or grid numbers to geometry.
- The OGC Gazetteer Service can also be thought of as a type of geocoding, but it is more specialized in terms of handling Gazetteer requirements (i.e., searching feature hierarchies and discovering feature types).

Some simple use cases are presented here as examples of how Geocoding Services might be employed.

Use Case 1 - Address Matching from a String

In this use case, a set of text strings containing addresses are passed to the Geocoder Service, and a Feature Collection with applicable features with geometries is returned.

Use Case 2 - Address Normalization

```
<StreetNo>184</StreetNo>
<Street>Milton Street</Street>
<CornerOf>Rose Street</CornerOf>
<Town>Annandale</Town>
<Postcode>15037</Postcode>
<State>NSW</State>
<LocationOf>
      <Point>
          <coordinates>152,-33</coordinates>
      </Point>
</LocationOf>
<GeocodeType>StreetNumberRange</MatchType>
<GeocodeNote>Unambiguous address, calculated location from
```

```
        street number</GeocodeNote>
<GeocodePrecision units="meters">100</GeocodePrecision>
```

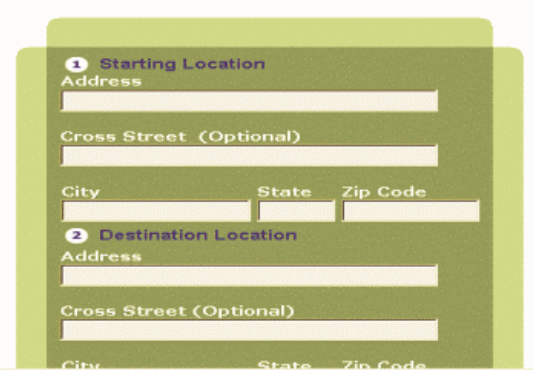
In this example, a feature containing properties that make up a structured address is passed to the Geocoder Service and the service responds with a feature containing a **normalized** address and coordinates. In this case only those elements that can be matched are returned as feature properties, and a status message and code are included. This allows a Geocoder Service to handle the common situations where, for example, address ranges for blocks are used to approximate exact locations in place of actual street frontage locations. Some of the returned information, like precision, are part of the Quality of Service (QoS) that the geocoder service provides.

Use Case 3 – Customer Database Use Case

A company has a database with a list of its customers and addresses. They want to geocode their database and have a geometry attached to the addresses. This will then be used to display the customers on a map or find the customers that are 100km around their new subsidiary in Italy.

Use Case 4 – Address Verification

A Web site or a customer service application often needs to have a valid address. This is required for many reasons. The simplest being accurate billing or shipping information. More advanced and urgent needs would include a 911-response application. The Web page or application would present the end user with input fields to enter their address.



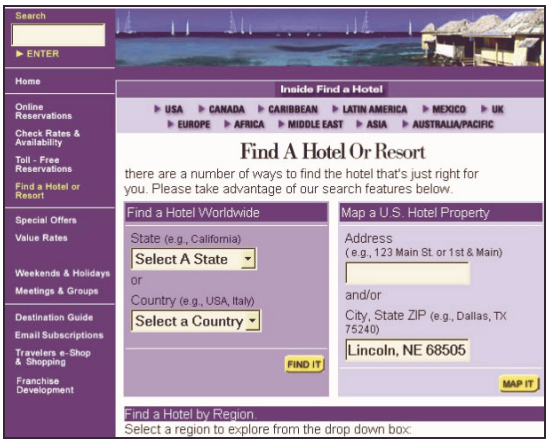
This is a sample input screen.

The response would be either Address OK, Invalid address, or if the application wants to assist. State that the address is invalid, this is the closest match, is it correct?

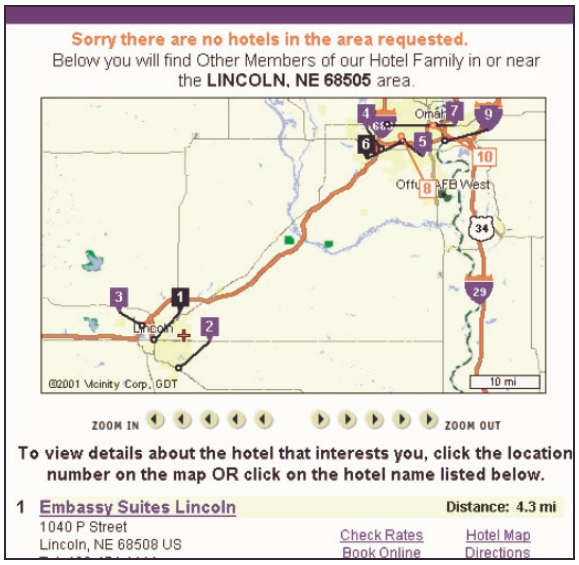
See Appendix C – Use Case 4 for the request and response for this use case. The important issue here is that the response include the 'cleaned' address.

Use Case 5 – Map Locations

Many web applications use an address to define what map should be displayed to the user. An example would be the end user enters an address in a form like:



And the resulting is a map such as:



See Appendix C – Use Case 5 for the request and response for this use case. The important issue here is that the response is a point feature that can easily be passed on to a mapping server.

1.2. Architectural Constraints (Informative)

Geocoding Types

For this version of the specification, geocoding is limited to the capabilities of the underlying geocoding engines. Appendix A contains a list of types that are commonly used to normalize addresses, like postalCode, StreetName, Country, etc…

Interface Compatibility

According to the OGC Basic Service Model, the Geocoder Service must support a <GetCapabilities> request, which when called upon, responds with a list of supported well-known `GeocodingEntryTypes` that have been defined for the service, as shown in Appendix A. Client applications must use the <DescribeFeatureType> interface to discover the structure (Schema) for these types.

Two Geocoding Approaches

Filter based geocoding. This approach defines geocoding through the <GetFeature> interface**,** involving features with geometries that are stored in a virtual store. It is used to specify the feature type and properties that make up the geocoding request, in the form of a structured and powerful query. The <GetFeature> interface uses a query (filter) syntax defined by the OGC Filter Encoding Specification to express the geocode request. The results of a <GetFeature> request are returned as a GML-encoded feature collection extending `GeocodingEntryType`. This interface takes as input a Feature Type (extending GeocodingEntryType) listed in the <GetCapabilities> response and having a schema defined by the <DescribeFeatureType> response. The <GetFeature> interface encapsulates the use of special geocoding algorithm operations in the filter operators.

This approach does not mean that the interface only supports exact match. When the "=" is used on a property in the filter, it means that this value is known exactly. When the "like" is used, it means that the geocoder can do loose or fussy search on this term. In the following example: "*where Street like '123 Main Street ' and City=' New York' and ' Country=' US' "* means that the geocoder could find features like « North Main Street, 123 » and « S. Main Street - 123 » for the Street but an exact match is wanted on "New York" and "US".

This means that this interface can hide complex algorithm of the implementation behind the operators.

Collection based geocoding. The <GeocodeFeature> interface accesses the Geocoding Service by using a feature collection (of one or more features) as input. This defines the input values to the Geocoder as values of the feature's properties. The details of the underlying implementation approaches are also hidden behind this interface and can be identical to those used to implement the <GetFeature> interface.

The <GeocodeFeature> operation accepts as input a `GeocodingEntryType` Feature Collection and a set of directives (Parameters) for controlling geocoding operations. The <GeocodeFeature> request operates on features (of type `GeocodingEntryType`) found in the input feature collection, updating designated property values of these features (e.g., their geometry properties), before returning the updated feature collection back to the client application. The result is a feature collection with updated property values (mainly feature geometry).

The distinction between the two interfaces is in the input. The <GetFeature> interface uses values in the Filter (analogous to the SQL WHERE clause) to define the input values for the Geocoding process. The operators in the Filter determine the directives, such as a direct or fuzzy search. The <GeocodeFeature> interface uses values in the input feature collection to define the input values for the Geocoding process. The directives to the process are stated explicitly.

Four interfaces

In summary, four interfaces apply to this version of Geocoder Service:

- GetCapabilities
- DescribeFeatureType
- GetFeature
- GeocodeFeature

A Geocoder Service must implement GetCapabilities and DescribeFeatureType and at least one of GetFeature or GeocodeFeature. They are defined in Section 2.

Query Constraints

Some Geocoder Service implementations are able to support spatial search constraints. Both interfaces can handle spatial searches. The <GetFeature> interface uses the OGC Filter Encoding to specify a spatial search.

Terms Addressable Via a URI: If links to remote features or relations (Geolinks, which are based upon XLINK) are returned in the result, they must be compliant with GML 2.

2. Interface Definitions (Normative)

The following interfaces are part of Geocoder Service:

- <GetCapabilities>
- <DescribeFeatureType>
- <GetFeature>
- <GeocodeFeature>

A Geocoder Service must implement GetCapabilities and DescribeFeatureType and at least one of GetFeature or GeocodeFeature interfaces. The OGC transactional interface and the lock interface are optional interfaces that are not required. Specific implementations of geocoder may support all or some of the interfaces. The GetCapabilities interface is used to specify which interfaces are implemented.

2.1. Interface *<GetCapabilities>*

Request

This interface is used to request a capabilities document from a Geocoder Service. (It is expected that a primary common reference for capabilities will be the Basic Service Model). The capabilities will describe the service type (Geocoder) and its supported interfaces.

Request Syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.opengis.net/namespaces/wfs"
xmlns:wfs="http://www.opengis.net/namespaces/wfs"
xmlns="http://www.w3.org/2000/10/XMLSchema"
elementFormDefault="qualified">
   <annotation>
      <appinfo>GetCapabilitiesRequest.xsd v0.2 2001-02</appinfo>
      <documentation xml:lang="en">WFS interface schema. Copyright (c)
2001 OGC, All Rights Reserved.</documentation>
   </annotation>
   <!-- ================================================================
   Includes and Imports
   ================================================================ -->
   <!-- ================================================================
   Global elements and attributes
   ================================================================ -->
   <!-- ================================================================
   Root element
   ================================================================ -->
   <element name="GetCapabilities" type="wfs:GetCapabilitiesType"/>
   <!-- ================================================================
   Types
```

```
   ========================================================== -->
   <complexType name="GetCapabilitiesType">
      <attribute name="version" type="CDATA" use="required"/>
   </complexType>
</schema>
```

Response (Return Values)

Issue Name: Need capabilities expressed in XML Schema. (JVD, 19March2001)]

Issue Description: All interfaces (request and response schema) have been specified in XML Schema except GetCapabilities and GeocodeFeature. The content and structure of the GetCapabilities response for Geocoder Service remains to be finalized. It is dependent on progress of the Basic Service Model. For now the DTD form of the response schema will suffice.

Resolution: [TBD. (Initials, Date)]

Issue Name: Capabilities gaps. (SergeM, 02Feb2001)]

Issue Description: For both interfaces, we need to advertise some things like if we can do exact or fuzzy searches, street level versus postal code level?, street intersections, etc?, or if some elements are queryable and some are not, etc.

Resolution: [TBD. (Initials, Date)]

The following is the GetCapabilities DTD response for Geocoder Service.

```
<!--
   ************************************************************************
   *   Capabilities RESPONSE                                              *
   ************************************************************************
   -->
<!—NOTE: comments in this Document Type Definition impose additional
             constraints beyond those codified in the DTD syntax.  A
             conformant Geocoder Service must provide Capabilities XML
             that: (1) validates against the DTD and (2) does not violate
             the constraints stated in comments herein. -->

<!-- The parent element of the Capabilities document includes as children
        a Service element with general information about the service, a
        Capability element with specific information about the kinds of
```

```
        functionality offered by the service and a FeatureTypeList element
        defining the list of all feature types available from this service. -->
<!ELEMENT GEC_Capabilities (Service, Capability, FeatureTypeList) >

<!-- The version attribute specifies the specification revision to which this
        DTD applies.  Its format is one, two or three integers separated by
        periods: "x", or "x.y", or "x.y.z", with the most significant number
        appearing first.
        Future revisions are guaranteed to be numbered in monotonically
        increasing fashion, though gaps may appear in the sequence.  -->
<!-- The updateSequence attribute is a sequence number for managing
        propagation of the contents of this document.  For example, if a
        Geocoder Service adds some data feature types it can increment the
        update sequence to inform catalog servers that their previously
        cached versions are now stale.
        The format is a positive integer.
-->
<!ATTLIST GEC_Capabilities version CDATA #FIXED "0.0.1"
                           updateSequence CDATA "0">

 <!-- The SCHEMALANGUAGES entity defines the schema languages that a Geocoder
Service is capable of using to describe the schema of a feature.  This
        entity can be redefined by individual services to include other schema
        languages but XMLSCHEMA must always be defined. -->
```

```
<!ENTITY % SCHEMALANGUAGES "XMLSCHEMA" >
<!ELEMENT XMLSCHEMA EMPTY>

<!-- The RESULTFORMATS entity defines the output formats that the Geocoder
        Service can generate.  The mandatory format "GML2" must
        always be available.  This entity can be redefined by individual
        services to include other formats. -->
<!ENTITY % RESULTFORMATS "GML2">
<!ELEMENT GML2>

<!-- The ServiceTypes entity is used to distinguish
     Geocoder implementations in the 'Name' element of the Service
     metadata. -->
<!ENTITY % ServiceTypes " OGCGeocoder " >

<!ELEMENT OGCGeocoder EMPTY>

<!-- The Service element provides metadata for the service as a whole. -->
<!ELEMENT Service (Name, Title, Abstract?, Keywords?,
                       OnlineResource, Fees?, AccessConstraints?) >

<!-- The service name. -->

<!ELEMENT Name (%ServiceTypes;) >
```

```
<!-- A human-readable title to briefly identify this service in menus. -->
<!ELEMENT Title (#PCDATA) >
<!-- A descriptive narrative for more information about this service. -->
<!ELEMENT Abstract (#PCDATA) >
<!-- Short words to help catalog searching.  Currently, no controlled
        vocabulary has been defined. -->
<!ELEMENT Keywords (#PCDATA) >

<!-- The top-level HTTP URL of this service.  Typically the URL of a "home
        page" for the service.  See also the onlineResource attributes of
        <DCPType> children, below.  Currently, no non-HTTP platforms have been
        specified. -->
<!ELEMENT OnlineResource (#PCDATA)>

<!-- Elements indicating what fees or access constraints are imposed.
        The reserved keyword "none" indicates no constraint exists. -->
<!ELEMENT Fees (#PCDATA)>
<!ELEMENT AccessConstraints (#PCDATA)>

<!-- A Capability lists available request types, how exceptions may be
        reported, and whether any vendor-specific capabilities are defined.
        It also lists all the feature types available from this Geocoder
        Service. -->
<!ELEMENT Capability
```

```
                (Request, GeocodeCapabilities, VendorSpecificCapabilities?) >
   <!-- Available Geocoder request types are listed here. At least one of
           the values is required, but more than one may be given. -->
   <!ELEMENT Request (GetCapabilities |
                         DescribeFeatureType |
         GeocodeFeature |
                     GetFeature
                      )+ >
   <!ELEMENT GetCapabilities (DCPType+)>
   <!ELEMENT DescribeFeatureType (SchemaDescriptionLanguage,DCPType+)>
   <!ELEMENT SchemaDescriptionLanguage (%SCHEMALANGUAGES)+>
   <!ELEMENT GeocodeFeature (DCPType+)>
   <!ELEMENT GetFeature (ResultFormat,DCPType+)>
   <!ELEMENT ResultFormat (%RESULTFORMATS)+>


   <!-- Available Distributed Computing Platforms (DCPs) are
           listed here.  At present, only HTTP is defined. -->
   <!ELEMENT DCPType (HTTP) >
   <!-- Available HTTP request methods. -->
   <!ELEMENT HTTP (Get | Post)+ >
   <!-- HTTP request methods.  The onlineResource attribute indicates the URL
        prefix for HTTP GET requests (everything before the question mark and
        query string: http://hostname[:port]/path/scriptname); for HTTP POST
```

```
        requests, onlineResource is the complete URL.  The HTTP GET syntax
        for Map, Capabilities and FeatureInfo requests has been well defined
        and is described in the OGC Web Map Server Interface Specification.
        The POST formalism, wherein GetMap arguments are encoded in XML and
        POSTed to the server, has not yet been fully developed. -->
    <!ELEMENT Get EMPTY>
    <!ATTLIST Get onlineResource CDATA #REQUIRED>
    <!ELEMENT Post EMPTY>
    <!ATTLIST Post onlineResource CDATA #REQUIRED>


    <!-- The optional VendorSpecificCapabilities element lists any capabilities
            unique to a particular service.  Because the information is not known
            a priori, it cannot be constrained by this particular DTD.  A
            vendor-specific DTD fragment must be supplied at the start of the XML
            capabilities document
    -->
    <!ELEMENT GeocodeCapabilities (SRS, LatLonBoundingBox, Match+) >

    <!-- The types of match operations supported by the <GeocodeFeature>
    interface -->
    <!ELEMENT Match (Street|PostCode|Organization|Landmark|Intersection) #REQUIRED>)
    <!ELEMENT Street EMPTY>
    <!ELEMENT PostCode EMPTY>
```

```
<!ELEMENT Organization EMPTY>
<!ELEMENT Landmark EMPTY>
<!ELEMENT Intersection EMPTY>
<!--
    DEFINE THIS ELEMENT AS NEEDED IN YOUR XML
    <!ELEMENT VendorSpecificCapabilities (your stuff here) >
   -->

<!-- A list of feature types available from this service.  The following
        table specified the number and source of the various elements that
        are available for describing a feature type.

   Element         number comments
   =======         ====== ========
   FeatureType       1    this is the name of the feature type

   Title             1    an optional meaningful title for the
                          feature type (e.g. " Roads" for
                          ROADL_1M")

   Abstract          0/1  optional; no default

   Keywords          0/1  optional; no default
```

```
      SRS                1    the SRS that should be used when specifying
                              the state of the feature

      LatLonBoundingBox  1    exactly one is required; default from parent

      MetadataURL        0+   optional; no default

      Operations         1    a list of available operations
      -->
   <!ELEMENT FeatureTypeList (Operations?,FeatureType+)>

   <!ELEMENT Operations (Query)+>
   <!ELEMENT FeatureType (Name,Title?,Abstract?,Keywords?,
                              SRS,—Operations?,LatLonBoundingBox,
                              MetadataURL*)>
   <!ELEMENT Name (#PCDATA)>
   <!ELEMENT SRS (#PCDATA)>
   <!-- The LatLonBoundingBox attributes indicate the edges of the enclosing
           rectangle in latitude/longitude decimal degrees (as in SRS EPSG:4326
           [WGS1984 lat/lon]).  LatLonBoundingBox MUST be supplied regardless of
           what SRS the server may support, but it MAY be approximate if EPSG:4326
           is not supported.  Its purpose is to facilitate geographic searches
           without requiring coordinate transformations by the search engine. -->
   <!ELEMENT LatLonBoundingBox EMPTY>
```

```
<!ATTLIST LatLonBoundingBox
              minx CDATA #REQUIRED
              miny CDATA #REQUIRED
              maxx CDATA #REQUIRED
              maxy CDATA #REQUIRED>

<!-- A Geocoder Service MAY use zero or more MetadataURL elements to offer
        detailed, standardized metadata about the data underneath a
        particular feature type. The type attribute indicates the standard
        to which the metadata complies; the format attribute indicates how
        the metadata is structured.  Two types are defined at present:
        'TC211' = ISO TC211 19115; 'FGDC' = FGDC CSDGM. -->
<!ELEMENT MetadataURL (#PCDATA) >
<!ATTLIST MetadataURL
              type ( TC211 | FGDC ) #REQUIRED
              format ( XML | SGML | TXT ) #REQUIRED>
```

Exceptions

None.

Extensions to the Basic Service Model (BSM)<GetCapabilities> for Geocoder Services

A Geocoder Service defines a well-defined set of supported feature types, called **`GeocodingEntryTypes`**. Any feature returned by a Geocoder Service will be of type **`GeocodingEntryType`** or extend **`GeocodingEntryType`** by extension.

According to the BSM, a `<GetCapabilities>` request will return a list of **`GeocodingEntryType`** names supported by any given implementation of the Geocoder Service.

At this time, the "**service name**" in the capabilities is used to designate the implementation as a Geocoder Service. This will be updated in the future when a mechanism will be clarified in BSM because this prevents from having a service instance implementing more then one interface.

```
<Service>
  <Name>OGCGeocoder</Name>
  <Title>Oslo Geocoder Server</Title>
  <Abstract>Oslo Street Server </Abstract>
  <OnlineResource>http://TESTER/OSLO?</OnlineResource>
/Service>
```

2.2. Interface *<GetFeature>* for filter based Geocoder Service

Request

This interface supports geocoding operations for filter based Geocoder Service. Functionally, it is the same <GetFeature> interface that is used by other OGC specifications like WFS and the Gazetteer Service. The main difference comes from the **semantic** given to the interface and to the operators, as described below.

Request Syntax

A query request for the Geocoder Service is described by the <GetFeature> element. The definition of a <GetFeature> request is defined in the following XML Schema and in the OGC BSM [Ref7] specifications.

Issue Name: Use of XML Schema to extend BSM abstract interfaces (JohnD, 18Mar2001)]

Issue Description: Reference to WFS <GetFeature> interface in schema below is not correct. The Geocoder <GetFeature> request should instead inherit (extend), as required, from a BSM-defined abstract interface. In its most basic form, it is simply a concrete class of the BSM abstract class.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.opengis.net/namespaces/wfs"
xmlns:wfs="http://www.opengis.net/namespaces/wfs"
xmlns="http://www.w3.org/2000/10/XMLSchema"
elementFormDefault="qualified">
   <annotation>
      <appinfo>GetFeatureRequest.xsd v0.2 2001-02</appinfo>
      <documentation xml:lang="en">WFS interface schema. Copyright (c)
2001 OGC, All Rights Reserved.</documentation>
   </annotation>
   <!-- ======================================================
   Includes and Imports
   ====================================================== -->
   <include
schemaLocation="http://www.opengis.net/namespaces/wfs/FilterRequest.x
sd"/>
   <!-- ======================================================
   Global elements and attributes
   ====================================================== -->
   <element name="Query" type="wfs:QueryType"/>
   <!-- ======================================================
   Root element
   ====================================================== -->
   <element name="GetFeature" type="wfs:GetFeatureType"/>
```

```
   <element name="GetFeatureWithLock"
type="wfs:GetFeatureWithLockType"/>
   <!-- ================================================================
   Types
   ================================================================= -->
   <complexType name="GetFeatureType">
      <sequence>
         <element ref="wfs:Query" maxOccurs="unbounded"/>
         <element ref="wfs:Filter" minOccurs="0"/>
      </sequence>
      <attribute name="outputFormat" use="default" value="GML2"/>
      <attribute ref="wfs:handle" use="default" value=""/>
      <attribute ref="wfs:maxFeatures" use="default" value=""/>
   </complexType>
   <complexType name="GetFeatureWithLockType">
      <sequence>
         <element ref="wfs:Query" maxOccurs="unbounded"/>
         <element ref="wfs:Filter" minOccurs="0"/>
      </sequence>
      <attribute name="outputFormat" use="default" value="GML2"/>
      <attribute ref="wfs:handle" use="default" value=""/>
      <attribute ref="wfs:maxFeatures" use="default" value=""/>
   </complexType>
   <complexType name="QueryType">
      <sequence>
         <element ref="wfs:PropertyName" minOccurs="0"
maxOccurs="unbounded"/>
         <element ref="wfs:Filter" minOccurs="0"/>
      </sequence>
      <attribute ref="wfs:handle" use="default" value=""/>
      <attribute ref="wfs:typeName" use="required"/>
   </complexType>
   <simpleType name="OutputFormatType">
      <restriction base="string">
         <enumeration value="GML2"/>
         <enumeration value="CDATA"/>
      </restriction>
   </simpleType>
</schema>
```

Type Definitions

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 tag. The tag defines which feature type to query, what properties to retrieve and what constraints (spatial or non-spatial) to apply to those properties.

Filter

The Filter is used to constrain a query. Both spatial and/or non-spatial constraints can be specified as described in the OGC Filter Encoding Specification [Ref3]

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 value is GML 2.0.

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. ".

Specific Semantic for geocoding

This Geocoder specification uses the OGC Filter [Ref3] to express a query and provides additional Geocoder-specific semantics described here:

a) The concept of "**EditFields**" found in the <GeocodeFeature> interface, expressing that input values can be updated or not, is always implicitly set to **YES** in this interface due to its semantic.

b) In the filter, there is a "**PropertyIsEqualTo**" operator, which will mean an "**ExactMatch"** request for this property. There is a "**PropertyIsLike**" operator, which means that the geocoder has the right to make any kind of calculation, pattern matching, misspelling, or other fussy algorithm to search. If some property is considered to be absolutely correct, the "PropertyIsEqualTo" is used, and the service is designed to fail if there is no match.

c) The concept of "**relaxToPostcode**" has the following meaning: If the request fails, it is valid to make another request that will only search on the postcode and give back the coordinates (centroid) of this postcode. At this time, with the <getFeature> interface, it is needed to check that the request failed and issue a second request limited to the postcode. This could be encapsulated in the interface to do it in one request, but the way to do this is not yet defined and reserved for a future version.

Response (Return Values)

Refer to Appendix A for the normative specification of the Geocoder Service <GetFeature> response schema.

Exceptions

None.

2.3 Interface <*GeocodeFeature*> for Collection based Geocoder Service

Request

The function of the <GeocodeFeature> interface is to provide a client the means to geocode a feature collection (minimum 1 feature). The input feature collection can contain one or more features. The schema for the feature collection is derived from the type "**`GeocodingEntryType`**" schema (Appendix A).

The service automatically updates the input feature collection with the results of geocoding operations, adding output properties, such as the geometry. Optionally, the service may update input property values, such as the street address, but only if requested ("EditFields = no" by default). The feature collection that is returned is the same as that which was input, with updated properties, and it is possible that an input feature may be duplicated and returned with possible candidate responses.

There are at most **`MaxFeatures`** returned for each feature. Note that these candidates usually result from ambiguous input, such as "123 Main" where there are both "123 S. Main" and "123 N. Main" possibilities, but no "123 Main". If **`MaxFeatures`** =1, then only exact matches are returned.

The **`<GeocodeFeature>`** request is composed of features and geocoding directives. The geocoding directives define which of the advertised capabilities the Geocoder Service is to use for the request.

Request Syntax

The definition of a <**`GeocodeFeature`**> request is defined as a multi-MIME part HTTP POST request. The first part contains a GML Feature Collection in a part called "REQUESTDATA". The parameters for SRS, MaxFeatures, RequestMatch, etc are in the "REQUEST" part. .

Issue Name: GeocodeFeature request in XML Schema. (JohnD, 19March2001)]

Issue Description: All interfaces (request and response schema) have been specified in XML Schema except GetCapabilities and GeocodeFeature. The content and structure of the GeocodeFeature request for Geocoder Service remains to be finalized. For now the DTD form of the request schema will suffice.

An XML Schema for the REQUEST will be written here instead of this DTD. Also, adhoc clarification will be added.

Convert DTD below to XML Schema.

Resolution: [Insert Resolution Details and History (initials, date)]

```
<!-- ************************************************************************
-->
        <!—geocoding request data    -->
        <!--
**********************************************************************   --
>
        <!ELEMENT GeocodeRequest (AddressMatchConstraints,
ResponseConstraints)>
                <!ATTLIST GeocodeRequest MatchSequence (address |
postalcode | address_postalcode) "address_postalcode">
        <!ELEMENT AddressMatchConstraints ((MaxFeatures?, MaxRanges?,
CornerOffset?, StreetOffset?)>
        <!ELEMENT MaxFeatures (#PCDATA)>
        <!ELEMENT MaxRanges (#PCDATA)>
        <!ELEMENT CornerOffset (#PCDATA)>
             <!ATTLIST CornerOffset Units (mt | ft | mi | km) #REQUIRED>
        <!ELEMENT StreetOffset (#PCDATA)>
              <!ATTLIST StreetOffset Units (mt | ft | mi | km) #REQUIRED>
        <!ELEMENT USPSRules EMPTY>
        <!ELEMENT CustomRules (AcceptFirstOfMultiple,
AcceptCloseMatchesOnly, RelaxHouse, RelaxStreet, RelaxPostalCode,
StreetMatchOnly, SearchRadiusExtension?)>
        <!ELEMENT AcceptFirstOfMultiple (#PCDATA)>
        <!ELEMENT AcceptCloseMatchesOnly (#PCDATA)>
        <!ELEMENT RelaxHouse (#PCDATA)>
        <!ELEMENT RelaxStreet (#PCDATA)>
        <!ELEMENT RelaxPostalCode (#PCDATA)>
        <!ELEMENT StreetMatchOnly (#PCDATA)>
        <!ELEMENT SearchRadiusExtension (#PCDATA)>
                <!ATTLIST SearchRadiusExtension Units (mt | ft | mi | km)
#REQUIRED>
                <!ATTLIST SearchRadiusExtension LimitSearch (true |
false) #REQUIRED>
        <!ELEMENT ResponseConstraints SRS>
             <!ATTLIST ResponseConstraints EditFields (YES | NO) “NO”>
```

Issue Name: Undocumented request elements. (JVD, 19March2001)]

Issue Description: The following elements defined in the above DTD do not have an associated description of their type and semantic:

1. MatchSequence
2. AddressMatchConstraints
3. MaxRanges

4. CornerOffset
5. StreetOffset
6. USPSRules
7. CustomRules
8. AcceptFirstOfMultipe
9. AcceptCloseMatchesOnly
10. RelaxHouse
11. RelaxStreet
12. RelaxPostalCode
13. SearchRadiusExtension
14. ResponseConstraints

Resolution:

Type Definitions

GeocodeFeature

The <GeocodeFeature> element can be used to geocode one or more features in a single request. The results of all geocoding requests are concatenated to produce the result set.

FeatureCollection

A feature collection is defined in the GML 2.0 Specification. All features in the collection are of a feature type derived from the `GeocodingEntryType`.

Attributes

Response (Return Values)

Refer to Appendix A for the normative specification of the Geocoder Service <GeocodeFeature> response schema.

Exceptions

None.

2.4. Interface *<DescribeFeatureType>*

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 Geocoder Service can provide. The description that is generated will define how a Geocoder 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 a GML-compliant feature definition expressed using XML Schema.

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

Request Syntax

The definition of the <DescribeFeatureType> request is defined in the following XML Schema and in the OGC BSM specifications.

Issue Name: Use of XML Schema to extend BSM abstract interfaces (JohnD, 18Mar2001)]

Issue Description: Reference to WFS <DescribeFeature> interface in schema below is not correct. The Geocoder <DescribeFeatureType> request should instead inherit (extend), as required, from a BSM-defined abstract interface. In its most basic form, it is simply a concrete class of the BSM abstract class.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.opengis.net/namespaces/wfs"
xmlns:wfs="http://www.opengis.net/namespaces/wfs"
xmlns="http://www.w3.org/2000/10/XMLSchema"
elementFormDefault="qualified">
   <annotation>
      <appinfo>DescribeFeatureTypeRequest.xsd v0.2 2001-02</appinfo>
      <documentation xml:lang="en">WFS interface schema. Copyright
(c) 2001 OGC, All Rights Reserved.</documentation>
   </annotation>
   <!-- ======================================================
   Includes and Imports
   ====================================================== -->
   <!-- ======================================================
   Global elements and attributes
   ====================================================== -->
   <!-- ======================================================
   Root element
   ====================================================== -->
   <element name="DescribeFeatureType"
type="wfs:DescribeFeatureTypeType"/>
   <!-- ======================================================
   Types
   ====================================================== -->
   <complexType name="DescribeFeatureTypeType">
      <sequence>
```

```
        <element name="TypeName" type="string" minOccurs="0"
maxOccurs="unbounded"/>
      </sequence>
      <attribute name="outputFormat" type="wfs:OutputFormatType"
use="default" value="XMLSCHEMA"/>
   </complexType>
   <simpleType name="OutputFormatType">
      <restriction base="string">
         <enumeration value="XMLSCHEMA"/>
         <enumeration value="CDATA"/>
      </restriction>
   </simpleType>
</schema>
```

Type Definitions

TypeName

Used to list the set of feature types to describe. Currently, this specification defines the `GeocodingEntryType`, as listed in Appendix B.

Attributes

`outputFormat`

Used to indicate the schema description language that should be used to describe a feature schema. The only mandated format is XMLSCHEMA.

Response (Return Values)

The Geocoder Service only supports feature types that are or extend `GeocodeEntryType` as specified in Appendix B. This schema conforms to the requirements expressed in W3C XML Schema and GML specifications.

The `GeocodeEntryType` must use normalized types (properties) from the Geocode Schema (namespace `geocoder`) defined in Appendix A. Returning a `GeocodeEntryType` element of type USZipCode should either be of type `geocoder:PostCode` or have a property of this type. Doing this ensures interoperability and a common understanding of the meaning of Geocoder results.

Exceptions

None.

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: **<**http://feature.opengis.org/members/archive/arch01/01-029.pdf**>**

2. **Web Feature Server Specification.** OGC Draft Candidate Implementation Specification, available online: **<**http://feature.opengis.org/members/archive/arch01/01-023r1.pdf**>**

3. **Filter Encoding Specification.** OGC Draft Candidate Implementation Specification, available online: <http://feature.opengis.org/members/archive/arch01/01-033.pdf>

4. **Gazetteer Service Specification.** OGC Draft Candidate Implementation Specification, available online: <http://feature.opengis.org/members/archive/arch01/01-036.pdf>

5. **Geoparser Service Specification.** OGC Draft Candidate Implementation Specification, available online: <http://feature.opengis.org/members/archive/arch01/01-035.pdf>

6. Whiteside, A, and J. Bobbit. 2000. ***Recommended Definition Data for Coordinate Reference Systems and Coordinate Transformations***. OGC Project Document 00-040r7. <http://feature.opengis.org/members/archive/arch00/01-040r7.pdf>

7. **Basic Service Model V0.0.8.** OGC Discussion Paper**,** available online: <http://feature.opengis.org/members/archive/arch01/01-022r1.pdf>

8. **W3C Candidate Recommendation for XML Schema (24 October 2000),** available online: <http://www.w3.org/TR/2000/CR-xmlschema-0-20001024/>

Glossary

Term	Definition
Geocode:	The namespace for the geocode Schema
GeocodingEntryType	The Schema of the base geocoding feature type

Appendix A: Geocoding Types and Geocoding Schema (Normative)

A Geocoding Type is a Universal Type that can be understood by any application, developer, or reader of the specification. It is the common international agreement and the first common basis for interoperability. It has the following parts:

- A name
- A definition in a formal language
- A description and semantic: a description of this type and all information to understand his meaning.
- A responsible standards body
- A version
- An example

Some of the types may have been proposed in other specifications and all our research could not isolate common agreement on this. The types and their semantics will be updated or synchronized in the future, if necessary.

The descriptions below and the definitions in the .xsd file should be consistent. When they are not, this document (not the .xsd) file takes precedent.

TYPE: POSTCODE

GEOCODING TYPE	POSTCODE
Name	PostCodeType
Definition	<xsd:simpleType name="postCodeType"> <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	A postal code or zip code
Semantic	Postal Code – An area encompassing a section of a street, a collection of streets, an establishment, a structure, a group of post office boxes, and so forth, which is assigned the same code by the Postal Service of the country. E.g., Code Postal,

	Zip Code
Examples	03124
RespBody	OGC
Version	1.0

For example, if someone writes:

NOT GOOD:

```
<xsd:simpleType name="usZipCodeType"
        <restriction base="xsd:string">
            <xsd:pattern value="([0-9]{5}|[0-9]{5}-[0-9]{4})"/>
        </restriction>
</xsd:simpleType>
```

This is not understandable as a postal code by an application that is looking for a Postal Code, and so, we lack the minimal worldwide interoperability. But if you write:

GOOD :

```
<xsd:simpleType name="usZipCodeType"
        <restriction base=" geocode:postCodeType ">
            <xsd:pattern value="([0-9]{5}|[0-9]{5}-[0-9]{4})"/>
        </restriction>
</xsd:simpleType>
<xsd:simpleType name=" beCPType "
        <restriction base="geocode:postCodeType">
            <xsd:pattern value"([0-9]{4})"/>
        </restriction>
</xsd:simpleType>
```

then, any application that knows what to do with a “postCodeType” would be able to handle both US and Belgium Zip codes.

List of Geocoding Types

The geocoding simple types are defined (initially) as these:

- Organization
- Street Address
- Unstructured street address
- Street Locators (Number, Building, CornerOf)
- Municipal Subdivision
- Municipality
- Post Code, Government Information
- Country Subdivision, Country, Region
- Place names and landmarks
- PersonName, Telephone, Fax, IPAddress, and hoursOfService

TYPE: ORGANIZATION

GEOCODING TYPE	ORGANIZATION
Name	OrganizationType
Definition	<xsd:simpleType name="organizationType"> <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	The name of an organization or company.
Semantic	The name of an organization or company; the name or title under which a company transacts business. The name of a partnership of two or more persons that is not recognized as a legal person distinct from its members. The name of a business unit or enterprise. It can be a company, Government bureau, Name of a School, etc…
Examples	Oracle, OpenGIS Consortium, Bureau of Census, FBI, NATO, etc
RespBody	OGC
Version	1.0

TYPE: PERSON NAME

GEOCODING TYPE	PERSON NAME
Name	PersonNameType
Definition	<xsd:simpleType name="personNameType"> <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	The name of a person.
Semantic	The name of a person
Examples	Mr. John Smith, Mrs. Jane Robinson, …
RespBody	OGC
Version	1.0

TYPE: STREET

GEOCODING TYPE	STREET
Name	StreetType
Definition	<xsd:simpleType name="streetType"> <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	The street address or name of the street.
Semantic	The street address or name, place name, etc…
Examples	5th Avenue, Avenue de l'Observatoire, Oracle Drive, Johnston Street
RespBody	OGC
Version	1.0

Sometimes, the number (streetNumber) is embedded in the street.

TYPE: UNSTRUCTURED STREET

GEOCODING TYPE	UNSTRUCTURED STREET
Name	UnstructuredStreetType
Definition	<xsd:simpleType name="unstructuredStreetType"> <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	An unstructured street address.
Semantic	The street address in unstructured form. Might include information in non-structured, non-normalized form. Some Geocoding Services will handle this case. This should be used only if really required.
Examples	150 Troy-Schenectady Road Fenimore Trace Apts 1-L
RespBody	OGC
Version	1.0

TYPE: MUNICIPAL SUBDIVISION

GEOCODING TYPE	MUNICIPAL SUBDIVISION
Name	MunicipalSubdivisionType
Definition	<xsd:simpleType name="municipalSubdivisionType"> <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	The name of an administrative or cultural division of a municipality; a distinct part of a city.
Semantic	The name of a division of a municipality – district, precinct, section, sector, area, neighborhood, vicinity, etc.
Examples	NW Washington, Little Italy, Le carre
RespBody	OGC
Version	1.0

TYPE: MUNICIPALITY

GEOCODING TYPE	MUNICIPALITY
Name	MunicipalityType
Definition	<xsd:simpleType name="municipalityType"> <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	The name of a city or municipality.
Semantic	The name of a City - an inhabited place, a town, village, etc. (e.g., Urban, burghal, municipal).
Examples	Washington, Liège, Paris, Annandale
RespBody	OGC
Version	1.0

TYPE: POSTCODE

GEOCODING TYPE	POSTCODE
Name	PostCodeType
Definition	<xsd:simpleType name="postCodeType"> <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	A postal code or zip code.
Semantic	An area encompassing a section of a street, a collection of streets, an establishment, a structure, a group of post office boxes, and so forth, which is assigned the same code by the Postal Service of the country. E.g., Code Postal, Zip Code, …
Examples	03124, 2037
RespBody	OGC
Version	1.0

TYPE: GOVERNEMENT INFORMATION

GEOCODING TYPE	GOVERNMENT INFORMATION
Name	GovernmentInformationType
Definition	<xsd:simpleType name="governmentInformationType"> <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	A government Information number.
Semantic	A government Information number.
Examples	51013 is for VA (51) Arlington (013)
RespBody	OGC
Version	1.0

TYPE: COUNTRY SUBDIVISION

GEOCODING TYPE	COUNTRY SUBDIVISION
Name	CountrySubdivisionType
Definition	<xsd:simpleType name="countrySubdivisionType"> <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	The name of a country subdivision like a state / region / département (France).
Semantic	The name of a country subdivision or state. A country subdivision is one of the constituent units of a country (nation). Synonym: State, province, Département, lander, etc…
Examples	California, Indre, Westfalie, nsw,
RespBody	OGC
Version	1.0

TYPE: COUNTRY

GEOCODING TYPE	COUNTRY
Name	CountryType
Definition	<xsd:simpleType name="countryType"> <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	The name of a country.
Semantic	The name of a country. A country is the land of a person's birth, residence, or citizenship. It's a political state, nation or its territory.
Examples	France, Belgium, Japan, Brazil, Australia
RespBody	OGC
Version	1.0

TYPE: REGION

GEOCODING TYPE	REGION
Name	RegionType
Definition	<xsd:simpleType name="regionType"> <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	One of the great divisions of land on the globe defined by cultural or physical geography.
Semantic	One of the great divisions of land on the globe defined by cultural or physical geography.
Examples	Asia, Europe, Australia, Antarctica, Africa, North America, South America, Oceania, Middle East, Eastern Europe, Western Europe, Atlantic Ocean, Pacific Ocean, Indian Ocean, etc.
RespBody	OGC
Version	1.0

TYPE: LANDMARK

GEOCODING TYPE	LANDMARK
Name	LandmarkType
Definition	<xsd:simpleType name="landmarkType"> <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	The name of the landmark.
Semantic	A historical or cultural landmark is any site (including significant trees or other plant life located thereon), building, or structure, of particular historic or cultural significance, such as historic structures or sites in which the broad cultural, political, economic, or social history of the nation, State, or community is reflected or exemplified, or which are identified with historic personages or with important events in the main currents of national, State, or local history, or which embody the distinguishing characteristics of an architectural-type specimen, inherently valuable for a study of a period style or method of construction, or a notable work of a master builder, designer, or architect whose individual genius influences his age.
Examples	GoldenGate Bridge,
RespBody	OGC
Version	1.0

TYPE: STREET LOCATOR

A location in a street can be expressed in multiple ways. They are all "Street locators".

GEOCODING TYPE	STREET LOCATOR TYPE
Name	StreetLocatorType

Definition	<xsd:simpleType name="streetLocatorType" abstract="true" <restriction base="xsd:string"> </restriction> </xsd:simpleType>
Description	The abstract street locator.
Semantic	The abstract street locator is the type of all street locators.
Examples	/
RespBody	OGC
Version	1.0

GEOCODING TYPE	STREET NUMBER TYPE
Name	StreetNumberType
Definition	<xsd:simpleType name="streetNumberType" <restriction base="streetLocatorType"> </restriction> </xsd:simpleType>
Description	The number of the location in the street.
Semantic	The number of the location in the street, or at the place name. It can be the house number (128, 4B, …). We need to define a logic for a list of values (37,42), a range (37-42), etc.
Examples	**745**, 5th Avenue; **128**, Avenue de l'Observatoire; **86-96**, main street
RespBody	OGC
Version	1.0

Sometimes, the street number is embedded in the street name.

GEOCODING TYPE	CORNER LOCATOR TYPE
Name	CornerLocatorType
Definition	<xsd:simpleType name="cornerLocatorType"

	<restriction base="streetLocatorType"> </restriction> </xsd:simpleType>
Description	The name of a street corner.
Semantic	The name of a street corner, or place name, indicating its position. (To be refined)
Examples	<CornerOf>Rose Street</CornerOf>
RespBody	OGC
Version	1.0

GEOCODING TYPE	BUILDING LOCATOR TYPE
Name	BuildingLocatorType
Definition	<xsd:simpleType name="buildingLocatorType"> <restriction base="streetLocatorType"> </restriction> </xsd:simpleType>
Description	The name of a building.
Semantic	The name of a building, indicating its position. (To be refined)
Examples	Empire State Building
RespBody	OGC
Version	1.0

GEOCODING TYPE	SUB BUILDING LOCATOR TYPE
Name	SubBuildingLocatorType
Definition	<xsd:simpleType name="subBuildingLocatorType"> <restriction base="streetLocatorType"> </restriction> </xsd:simpleType>
Description	A location in a building.
Semantic	This describes the location in a building. It can be

	the name of the floor in the building, the suite, the name of a ship, a studio, which gives information about the location inside a building, etc. This is also sometimes called "subbuilding".
Examples	1-L, 5-R, 27eme, flat 125, flat 2 third floor, unit 13, shop 6, studio 54, suite 226-233, second floor, estate office, sugar shack, south west wing (these examples come from real databases)
RespBody	OGC
Version	1.0

More streetLocatorTypes might be added in the future. Also, a GeocodingEntryType can have multiple streetLocatorTypes. For example:

```
<StreetNo>128</StreetNo>
<CornerOf>Rose Street</CornerOf>
<Building>Empire Johnson Building</Building>
```

List of Abbreviation Types

Abbreviations types will allows applications to expose abbreviations and express the type of abbreviations. For example, the GNS database contains country abbreviation. A GeocodingEntryType will expose one of its field of type “countryAbbreviationType” that will be well known and well understood by another application. Without this, we would have to present it as String, which would be meaningless, or as countryType, which would then need clarification.

TYPE: STATE ABBREVIATION

GEOCODING TYPE	STATE ABBREVIATION
Name	StateAbbreviationType
Definition	<xsd:simpleType name="stateAbbreviationType"> <restriction base="xsd:string” > </restriction> </xsd:simpleType>
Description	The abbreviation of a state.
Semantic	The abbreviation of a state. A state is country dependent and the pattern or enumeration will be different, but by this we will know that it’s a state abbreviation.
Examples	ca, vi, co,
RespBody	OGC
Version	1.0

TYPE: COUNTRY ABBREVIATION

GEOCODING TYPE	COUNTRY ABBREVIATION
Name	CountryAbbreviationType
Definition	<xsd:simpleType name="countryAbbreviationType"> <restriction base="xsd:string” > <xsd:pattern value="([0-9]{2})" /> </restriction> </xsd:simpleType>
Description	The abbreviation of a country.

Semantic	The abbreviation of a country is a 2 (to be determined) character representation / abbreviation of a Country.
Examples	Be, us, uk, nl, fr
RespBody	OGC
Version	1.0

Issue Name: [GEC-13, (SM, 10/25/00)]

Issue Description: [Should we provide a list for all countries? What about name in multiple languages (Belgium, Belgique, Belgie, Belgica, etc…)? How do we relate it to ISO?)]

Resolution: [Proposed: Need to decide on this. (HAN, 11/30/00)]

TYPE: PLACE NAME ABBREVIATION

GEOCODING TYPE	PLACENAME ABBREVIATION
Name	PlaceNameAbbreviationType
Definition	<xsd:simpleType name="placeNameAbbreviationType"> <restriction base="xsd:string" > </restriction> </xsd:simpleType>
Description	The abbreviation of a city or place name.
Semantic	The abbreviation of a city or place name.
Examples	‘Wiltz’ when full name is 'Canton de Wiltz' (this is often found, for example, in GNS database)
RespBody	OGC
Version	1.0

Quality Of Service Type
(Part still to be updated with BSM)

The quality of service (QOS) is a complex type made of multiple parts. It consists of:

- MATCHTYPE: e.g., StreetNumberRange
- RESULTCODE: returns a numeric non-match code that is vendor specific; for clients to investigate match results at a detailed level, if desired.
- NOTETYPE: e.g. Unambiguous address, calculated location from street number.
- PRECISIONTYPE: e.g. 100 meters
- ACCURACY: e.g.: 0.82 (82% probability of match between input and returned features).

GEOCODING TYPE	ACCURACY
Name	geocodeAccuracyType
Definition	<xsd:simpleType name="geocodeAccuracyType"> <restriction base="xsd:float" > </restriction> </xsd:simpleType>
Description	The accuracy of the service.
Semantic	A float value between 0 and 1 defining the accuracy (error probability) of the result.
Examples	0.82
RespBody	OGC
Version	1

GEOCODING TYPE	RESULTCODE
Name	GeocodeResultCodeType
Definition	<xsd:simpleType name="geocodeResultCodeType"> <restriction base="xsd:string" > </restriction> </xsd:simpleType>

Description	Numeric result information.
Semantic	A String that is service dependent.
Examples	E1235F
RespBody	OGC
Version	1

GEOCODING TYPE	GEOCODE NOTE
Name	GeocodeNoteType
Definition	<xsd:simpleType name=" geocodeNoteType " <restriction base="xsd:string" > </restriction> </xsd:simpleType>
Description	A note or comment on the process.
Semantic	A note or comment on the process. The field can be free.
Examples	Unambiguous address; calculated location from street number.
RespBody	OGC
Version	1.0

GEOCODING TYPE	GEOCODE PRECISION
Name	geocodePrecisionType
Definition	
Description	The precision of the result
Semantic	To be defined. This could be something more formal, units, semantic, etc…
Examples	
RespBody	OGC
Version	1.0

GEOCODING TYPE	MATCH
Name	geocodeMatchType

Definition	
Description	The type of matching
Semantic	To be defined
Examples	
RespBody	OGC
Version	1.0

GEOCODING TYPE	QUALITY OF SERVICE
Name	geocodingQOSType
Definition	<xsd:complexType name="geocodingQOSType"> <xsd:element name="accuracy" type=”geocodeAccuracyType” minOccurs="0" maxOccurs="1"/> <xsd:element name="precision" type=”geocodePrecisionType” minOccurs="0" maxOccurs="1"/> <xsd:element name="note" type=”geocodeNoteType” minOccurs="0" maxOccurs="1"/> <xsd:element name="accuracy" type=”geocodeMatchType” minOccurs="0" maxOccurs="1"/> </xsd:complexType>
Description	The quality of geocoding service
Semantic	The quality of geocoding service
Examples	
RespBody	OGC
Version	1.0

A representative schema for GeocodingQOSType is:

```
<xsd:complexType    name="geocodingQOSType "
  <xsd:element name="accuracy" type=” geocodeAccuracyType minOccurs="0"
```

```
                    maxOccurs="1"/>
  <xsd:element name="precision" type=" geocodePrecisionType " minOccurs="0"
                    maxOccurs="1"/>
  <xsd:element name="note" type=" geocodeNoteType " minOccurs="0"
                    maxOccurs="1"/>
  <xsd:element name="accuracy" type=" geocodeMatchType "  minOccurs="0"
                    maxOccurs="1"/>
</xsd:complexType>
```

List of other Geocoding Types

TYPE: TELEPHONE

GEOCODING TYPE	TELEPHONE
Name	telephoneType
Definition	<xsd:simpleType name="telephoneType"> <restriction base="xsd:string" > </restriction> </xsd:simpleType>
Description	The telephone number
Semantic	The telephone number
Examples	+3243640364
RespBody	OGC
Version	1.0

ED Question: is there a normalization Schema of phone to reference

TYPE: FAX

GEOCODING TYPE	FAX
Name	FaxType
Definition	<xsd:simpleType name="faxType"> <restriction base="telephoneType" > </restriction> </xsd:simpleType>
Description	The fax number
Semantic	The fax number
Examples	+3242534737
RespBody	OGC
Version	1.0

TYPE: NETWORK ADDRESS

GEOCODING TYPE	NETWORK ADDRESS
Name	NetworkAddressType
Definition	<xsd:simpleType name="networkAddressType"> <restriction base="xsd:string" > </restriction> </xsd:simpleType>
Description	A Network address
Semantic	A Network address.
Examples	
RespBody	OGC
Version	1.0

We could have subtypes like ip4addressType (e.g., 63.123.112.23) or ip6addressType (e.g., 123.251.63.123.112.23)

TYPE: Hours of Service

GEOCODING TYPE	Hours of Service
Name	HoursOfServiceType
Definition	<xsd:simpleType name="hoursOfServiceType"> <restriction base="xsd:string" > </restriction> </xsd:simpleType>
Description	Hours of service
Semantic	The hours of service.
Examples	For the moment it's a String, in conformance with GILS specification. It could be based on GML.
RespBody	OGC
Version	1.0

Geocoding Types (Normative):

The Geocoder XML Schema document shown (in part) here can be found in its entirety online at <http://www.opengis.net/namespaces/geocoder/geocoder.xsd>

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- geocode0.7.39.24oct.xsd  -->
<!--
   This schema contains the Type definition of the OGC Geocoding
specification
   - the namespace prefix is geocoder:
   - Description and semantics can be found in the OGC geocoding
specification
   - this version is maintained by Serge Margoulies, IONIC Software
   - send me your comments at sm@ionicsoft.com
-->
<xsd:schema
targetNamespace="http://www.opengis.net/namespaces/geocoder"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml"
xmlns="http://www.opengis.net/namespaces/geocoder"
xmlns:geocoder="http://www.opengis.net/namespaces/geocoder"
xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
elementFormDefault="qualified" version="0.1">
   <!-- import constructs from the GML Feature and Geometry schemas
-->
   <xsd:import namespace="http://www.opengis.net/gml"
```

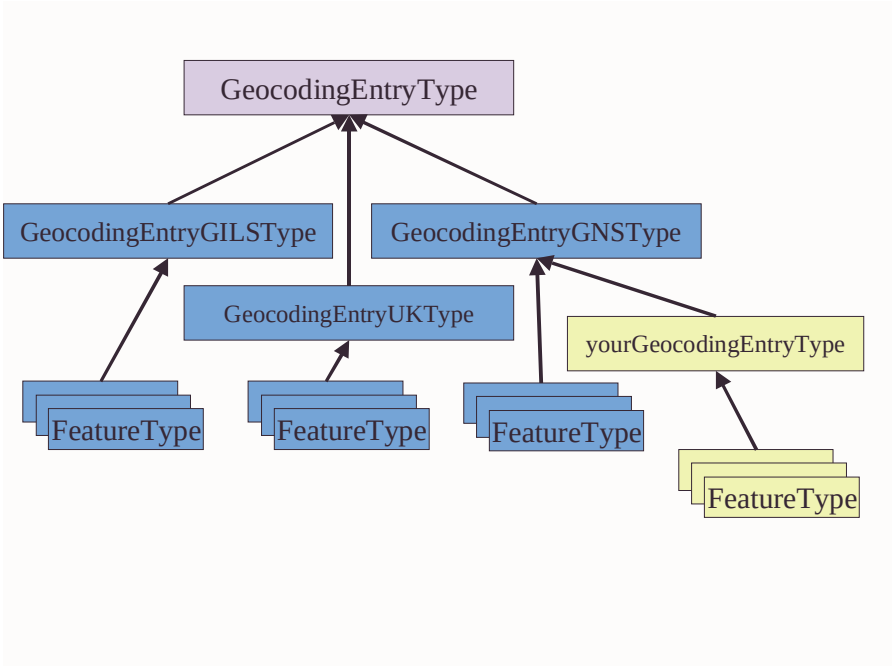
```
schemaLocation="http://www.opengis.net/namespaces/gml/core/feature
.xsd"/>
   <xsd:simpleType name="organizationType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="personNameType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="streetType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="unstructuredStreetType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="municipalSubdivisionType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="municipalityType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="postCodeType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="countrySubdivisionType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="countryType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="regionType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="landmarkType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="streetLocatorType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="streetNumberType">
      <xsd:restriction base="streetLocatorType"/>
   </xsd:simpleType>
   <xsd:simpleType name="cornerLocatorType">
      <xsd:restriction base="streetLocatorType"/>
   </xsd:simpleType>
   <xsd:simpleType name="buildingLocatorType">
      <xsd:restriction base="streetLocatorType"/>
   </xsd:simpleType>
   <xsd:simpleType name="subBuildingLocatorType">
      <xsd:restriction base="streetLocatorType"/>
   </xsd:simpleType>
   <xsd:simpleType name="stateAbbreviationType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="countryAbbreviationType">
      <xsd:restriction base="xsd:string">
         <xsd:pattern value="([0-9]{2})"/>
      </xsd:restriction>
```

```
   </xsd:simpleType>
   <xsd:simpleType name="placeNameAbbreviationType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="telephoneType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="faxType">
      <xsd:restriction base="telephoneType"/>
   </xsd:simpleType>
   <xsd:simpleType name="networkAddressType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="hoursOfServiceType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
...
```

Remark : The model is that GeocodingEntryTypes can have any number of different properties, depending on the server implementation. To discover these properties, for a given server, the client must issue a <DescribeFeatureType> request. One of the properties of all GeocodingEntryTypes returned by the Geocoder Service will be of type gml:geometry.

Appendix B: Support for GeocodingEntryType (Normative)

The concept of `GeocodingEntryType` is to provide the means for Geocoder Service to support extensible (specialized) feature type hierarchies, as illustrated below.



Schemas for `GeocodingEntryType` and extended `GeocodingEntryType`

The Geocoder XML Schema document shown (in part) here can be found in its entirety online at <http://www.opengis.net/namespaces/geocoder/geocoder.xsd>

```
…
   <xsd:complexType name="geocodingQOSType">
      <xsd:sequence>
         <xsd:element name="accuracy" type="geocodeAccuracyType"
minOccurs="0"/>
         <xsd:element name="note" type="geocodeNoteType"
minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:simpleType name="geocodeAccuracyType">
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <xsd:simpleType name="geocodeNoteType">
```

```
      <xsd:restriction base="xsd:string"/>
   </xsd:simpleType>
   <!--
   Here is the base FeatureType for geocoding: the
GeocodingEntryType.
-->
   <xsd:complexType name="GeocodingEntryType">
      <xsd:annotation>
         <xsd:documentation>
                     The base type for GeocodingType.
                </xsd:documentation>
      </xsd:annotation>
      <xsd:complexContent>
         <xsd:extension base="gml:AbstractFeatureType">
            <xsd:sequence>
               <xsd:element name="QOS"
type="geocoder:geocodingQOSType" minOccurs="0"/>
               <xsd:element ref="gml:_Geometry"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
…
```

All subtypes can be provided, as needed (Informative)

```
<xsd:complexType  name="myGeocodingEntryType " base="geocodingEntryType"
        <xsd:annotation>
            <xsd:documentation>
                A proposed type for geocoding FeatureType for OGC
             </xsd:documentation>
        </xsd:annotation>
        <xsd:element name="organization" type=”organizationType" minOccurs="0"
                maxOccurs="1"/>
        <xsd:element name="street" type=”streetType” minOccurs="0" maxOccurs="1"/>
        <xsd:element name="streetNum" type=”streetNumberType” minOccurs="0"
                maxOccurs="1"/>
        <xsd:element name="secondaryLocator" type=”BuildingNumberType”
                equivClass="streetLocatorType”
                minOccurs="0"  maxOccurs="1"/>
        <xsd:element name="city" type=” municipalityType ” minOccurs="0" maxOccurs="1"/>
        <xsd:element name="state" type=”stateType” minOccurs="0" maxOccurs="1"/>
        <xsd:element name="postCode" type=”postCodeType” minOccurs="0" maxOccurs="1"/>
        <xsd:element name="country" type=”countryType” minOccurs="0" maxOccurs="1"/>
</xsd:complexType>
```

GeocodingEntryGILSType (Informative)

There is an Application Profile For The USA **Government Information Locator Service** (GILS). This standard is FIPS PUB192 and provides a Schema for a point of Contact with mandatory, non-repeatable elements. This is a good use case to define the XMLSchema of the GILS FeatureType, which will extend “geocodingEntryType” and is named “GeocodingEntryGILSType”.

GILS Use Case: If we look at the GILS, 11 fields are mandatory: NAME, ORGANISATION, STREET ADDRESS, CITY, STATE, ZIP, COUNTRY, NETWORK ADDRESS, HOURS OF SERVICE, TELEPHONE, FAX. The XML Schema is shown below.

```
   <xsd:complexType name="GeocodingEntryGILSType">
     <xsd:annotation>
       <xsd:documentation>
            A proposed type for geocoding GILS FeatureType
       </xsd:documentation>
     </xsd:annotation>
     <xsd:complexContent>
       <xsd:extension base="geocodingEntryType">
         <xsd:sequence>
           <xsd:element name="Name"
type="geocoder:personNameType" minOccurs="0"/>
           <xsd:element name="Organization"
type="geocoder:organizationType" minOccurs="0"/>
           <xsd:element name="street" type="geocoder:streetType"
minOccurs="0"/>
           <xsd:element name="streetNum"
type="geocoder:streetNumberType" minOccurs="0"/>
           <xsd:element name="city" type="geocoder:cityType"
minOccurs="0"/>
           <xsd:element name="state" type="geocoder:stateType"
minOccurs="0"/>
           <xsd:element name="postCode"
type="geocoder:postCodeType" minOccurs="0"/>
           <xsd:element name="country"
type="geocoder:countryType" minOccurs="0"/>
           <xsd:element name="networkAddress"
type="geocoder:networkAddressType" minOccurs="0"/>
           <xsd:element name="hoursOfService"
type="geocoder:hoursOfService" minOccurs="0"/>
           <xsd:element name="telephone"
type="geocoder:telephoneType" minOccurs="0"/>
           <xsd:element name="fax" type="geocoder:faxType"
minOccurs="0"/>
         </xsd:sequence>
       </xsd:extension >
     </xsd:complexContent>
   </xsd:complexType>
```

Appendix C: Examples (Informative)

Serge is in the process of creating an example based on the running material from GFSPP.

Sandra Johnson has indicated that she has a use case to include, as well.

top contents glossary references

Appendix D: Implementation Hints (Informative)

GeocodingEntryType by Extension

In the text, you can find that the FeatureType should be of type "`GeocodingEntryType`". A Feature having a type "MyType" that extends "`GeocodingEntryType`" by extension or any sub type of "`GeocodingEntryType`" is a "`GeocodingEntryType`". So when we write that it must be a "`GeocodingEntryType`", it means that it can be any FeatureType that extends the "`GeocodingEntryType`".

Gazetteer or Geocoder

In case of a need of hierarchy in the queries, then the Gazetteer Service should be used. It is possible (recommended) to use the Normalization of the elements of the FeatureType pointed to by the gazetteer using the types defined in the `geocoder` namespace.

geocoder: Namespace

The namespace prefix for geocoder is: "`geocoder:`". In XML Schema, it is important that the URL defines the namespace, not it's name or it's physical location. Thus, the following namespace definition applies:

```
xmlns:geocoder="http://www.opengis.net/namespaces/geocoder
```

The `geocoder` namespace should be used to qualify all geocoder elements.

top contents glossary references

Appendix E: Comments or material for reflection (Informative)

Issue Name: [Need to express queryable fields in the DescribeFeatureType. (SM, 3/17/01)]

Issue Description: [There is a need in the DescribeFeatureType to express fields that are queryable and those that are not. This is more general then this specification. It is needed for the GetFeature interface, in general.]

Resolution: [Open. (aa,bb)]

Issue Name: [There is no placeName Type. (SM, 3/17/01)]

Issue Description: [Is it the same as Landmark?]

Resolution: [Open. (aa,bb)]

Issue Name: [There is no placeName Type, but there is a placeNameAbbreviation. (SM, 3/17/01)]

Issue Description: [We have to check this because this is found in GNS.]

Resolution: [Open. (aa,bb)]

top contents glossary references

Appendix F: Informative References

References:

1. **Application profile for the Government Information Locator Service (GILS).** http://www.itl.nist.gov/fipspubs/fip192.htm
2. **Common Address Matching and Geocoding Terms.** http://www.geocode.com/geocode.htm
3. **Coding Accuracy Support System (CASS).** http://www.usps.gov/ncsc/programs/cass.html

top contents glossary references