Physical Address and Geocoding Standards

Conceptual Model

Version 1.0

Prepared by

Michael Ross, DataBC
Enterprise Data Services

Mar 15, 2010

Document Control

Date	Author	Version	Change Reference
May 29, 2009	Michael Ross, GeoBC	Draft v0.3	Initial Release
June 16, 2009	Michael Ross, GeoBC	Draft v0.4	Incorporated internal review comments
June 24, 2009	Michael Ross, GeoBC	Draft v.0.5	Incorporated review comments by geocoder vendor (Refractions Research)
June 29, 2009	Michael Ross, GeoBC	Draft v0.6	Incorporated review comments from June 25 presentation to NRSIWG
Sep 30, 2009	Michael Ross, GeoBC	Draft v0.7	Incorporated comments from Aug 26 NRSIWG review meeting including VIHA and Elections BC reps; revised introduction and scope; added context section; added support for non-civic addresses; added high level Site class diagram; expanded definition of geocoder operation
Nov 23, 2009	Michael Ross, GeoBC	Draft v0.8	Reorganized document into information model and service model sections. Elaborated on application/geocoder interaction in section 3.2.5 Added geocoding examples, concordance, and business context Incorporated feedback from Ministry of Small Business and Revenue (K Hatch)
Nov 25, 2009	Michael Ross, GeoBC	Draft v0.81	Revised definition of physical address.
Feb 9, 2010	Michael Ross, GeoBC	Draft v0.9	Incorporated feedback from ASRB review; removed compliance schedule Fixed OCL definition of PhysicalAddress.SiteName; expanded definition of PhysicalAddress.addressString and added examples; revised queryAddress parameter description in section 3.4.2 Restructured CivicAccessPoint into AccessPoint, CivicAddressPoint, NonCivicAddressPoint Added retire date to Site and AccessPoint In section 2.1, added retire date for site and accessPoint; added fullSiteDescriptor In section 2.2.6, fixed problems in OCL with civic and non-civic address operations Added geocoding examples (section 3.6) Improved OCL of geocoder operations (section 3.4) In section 3.4.2, added more examples of unstructured addresses Made minor edits throughout

Date	Author	Version	Change Reference
Feb 19, 2010	Michael Ross, GeoBC	Draft v0.91	In section 2.1.3, simplified presentation of address string formats and examples In section 3.2.2 added some missing standardization steps In section 3.4.2, simplified presentation of queryAddress formats and examples
Feb 25, 2010	Michael Ross, GeoBC	Draft v0.92	Added the concept of a street intersection; sections 2 and 3 updated In section 1, redefined physical address to include both site and street intersection addresses. In section 2, renamed PhysicalAddress SiteAddress to support new definition of Physical Address. In sections 2.3.1, 2.3.2.2, 2.3.6.9, added fullName operation to Street
Mar 1, 2010	Michael Ross, GeoBC	Draft v0.93	Incorporated latest feedback from ASRB In section 1.5, expanded discussion of security and privacy issues and added note on legal disclaimer. In section 3.3, Added copyrightNotice, copyrightLicense, disclaimer, and privacyStatement attributes to Geocoder, SearchResults, and IntesectionSearchResults Made some minor edits throughout
Mar 10, 2010	Michael Ross, GeoBC	Draft v0.94	Incorporated latest feedback from ASRB In section 1.4, Out of Scope, added Address Data Management In section 1 Introduction, added section 1.7 Related Standards In section 3.4.1 changed setBack type from integer to Real In section 3.4.2 added SetBack:Real to parameter list
Mar 15, 2010	Michael Ross, GeoBC	V1.0	released
Aug 30, 2012	Michael Ross, DataBC	V1.0.1	Minor corrections only; updated gov't organization names

TABLE OF CONTENTS

1. INTRODUCTION

Geocoding is the process of determining the geographic position (coordinates) of a street intersection, house, building, etc., from its physical address. Government has long expressed a need for a single, authoritative, physical address registry and geocoding service. This standard is an important step toward establishing such services.

1.1 Terminology

Civic Address means a site address in a part of the province that is administered by a civic address authority and includes a site name, unit, civic number, street name, and province.

Delivery Address means an address that describes a location with sufficient detail to allow delivery of physical goods.

GeoBC is the GEOBC Branch in the Integrated Resource Operations Division of the Ministry of Forest, Lands and Resource Operations.

Locality means a municipality, subdivision, community, Indian reserve, named natural feature, regional district, or aboriginal lands. The primary locality of a site is assigned by the appropriate administrative authority, not Canada Post. Direct location is represented by points in a coordinate reference system.

Mailing Address is an address that describes a location that can receive posted mail. Canada Post is the authority for mailing addresses in Canada.

MAY means that it is optional; often there is a practice to do something, however it is not a requirement.

MUST means an absolute requirement of the specification.

Non-civic Address means a site address in a part of the province not administered by a civic address authority and includes a site name, unit, locality, and province but not civic number and street name.

Physical address means the relative and direct location of a site or street intersection on the earth. Direct location is represented by coordinates (e.g., latitude, longitude) in a coordinate reference system. Relative location is represented by a site or street intersection address.

Site Address means the relative and direct location of a site on the earth. Direct location is represented by coordinates (e.g., latitude, longitude) in a coordinate reference system. Relative location is represented by a civic or non-civic address.

SHOULD means that there may exist valid reasons in particular circumstances to use alternate methods, but the full implications MUST be understood and carefully weighed before choosing a different course.

Site means a constructed geographic feature in British Columbia with known coordinates (latitude/longitude) and a site address that is needed in the conduct of provincial business. Examples of constructed geographic features include houses, cabins, permanent campsites, mobile home and RV parks, units within houses and buildings, buildings within complexes, stores, malls, offices, industrial plants, bandshell, golf courses, hospitals, universities, recreation centres, places of worship, parks, municipal pumping stations, hydro sub-stations, wharves, airports, train stations, and landmarks.

Street Intersection Address means the relative and direct location of the place where two or more streets meet or cross. Direct location is represented by a point in a coordinate reference system. Relative location is represented by the names of all the streets that meet or cross at the intersection.

1.2 Applicability

This standard applies to the conceptual models of information systems within core government that provide or consume geocoding services.

1.3 Exemptions

Exemptions to this standard may be granted through the OCIO Architecture and Standards Branch. See http://www.cio.gov.bc.ca/legislation/standards/ for more information on the exemption process.

1.4 Scope

This standard defines a conceptual model of sites and street intersections in British Columbia, their associated physical addresses, and geocoding services.

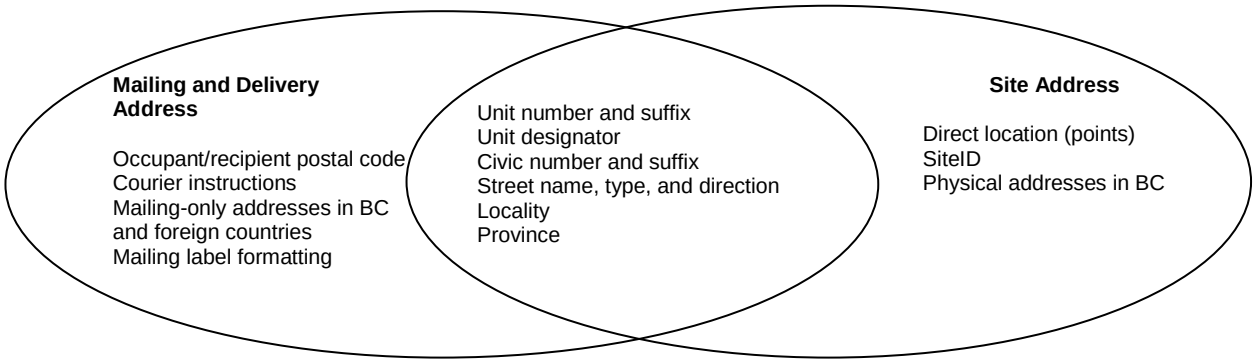
A geocoding service provides the following functions:

1. Address validation.
2. Address standardization.
3. Geocoding of both site and street intersection addresses.
4. Provision of unique site ids and street intersection ids for use in address maintenance by business applications.
5. Support for both real-time and batch operation (e.g., validation of a single address during form input and batch geocoding of one million addresses).

The following are out of scope:

1. Mailing and delivery address standards and requirements. This excludes:
    a. Certain address elements such as site occupants and postal code
    b. Mailing-only address types (e.g., lock box, rural route, and general delivery),
    c. Mailing label formatting information
    d. Courier instructions
    e. Assignment of locality by Canada Post. Since this standard is about geocoding, not mail delivery, the locality of a site is that defined by the appropriate administrative authority. For example, an address in the municipality of Esquimalt will be assigned a locality of Esquimalt whereas Canada Post would assign a locality of Victoria.
2. Site classification and usage. This tends to be business specific and sensitive in many cases.
3. Identification of data sources for addresses, road network, and locality extents (i.e., boundaries).
4. Driving directions between two civic addresses. No routing functions are defined.
5. Civic address assignment and signage guidelines. This is the responsibility of each municipality.
6. Address data management and related access control. Only read access in support of geocoding is in scope.

The following diagram illustrates the differences between mailing/delivery address and site address:



The conceptual model consists of an information model that defines the classes and associations needed to represent sites, street intersections, and physical addresses; and a service model for geocoding.

All models are defined in the Unified Modelling Language 2.0 [R5]. Operations are defined in UML Object Constraint Language 2.0 notation [R6].

All geometric data types used in this standard (e.g., Point, Polygon) conform to the OGC Simple Feature Access specification [R4].

All time and date types conform to the BC Date and Time Standard [R18].

1.5 Security, Privacy, and Legal Considerations

This conceptual model can be used as the basis for a secure or public geocoding service implementation. Ideally, a public geocoder will be implemented for general use and secure geocoders will be implemented to suit specialized business needs. It is the responsibility of the geocoding service implementer to complete the required Privacy Impact Assessment and Security Threat and Risk Assessment.

Although there are no personal attributes in the conceptual model, a geocoding service that aspires to public access must not store any personal or sensitive information. Diligence is required to prevent this information from seeping in, particularly into the site name and narrative location attributes. For example, a site name of "Secret Military Installation 33412" reveals its sensitive classification and a narrative location that includes "follow the creek east two miles past Roy Roger's Double R Bar Ranch" reveals personal information.

Given that some data used by a compliant geocoder may not be accurate, it is important to include an appropriate legal disclaimer with all geocoding service responses. See section 3.3 for details.

The conceptual model includes privacy statement, disclaimer, and copyright attributes as part of the Geocoder class definition (see section 3.3 for details).

1.6 Context

1.6.1 Business Context

Most government data has a spatial component which is usually an address. Address data is usually not represented spatially, denying many business areas the valuable techniques of geographic visualization and analysis. Address data is also not managed centrally which has led to duplication of effort, varying address data quality, and difficulty in sharing addresses across government.

Standardizing physical addresses and geocoding services enables common understanding and is a necessary step towards a government-wide distribution hub for all physical addresses in BC. Benefits of such a hub include the following:

1. Improved service delivery (e.g., fewer ambulances dispatched to the wrong address, fewer people not notified of emergency evacuations and oil and gas activity, fewer eligible voters not registered).

2. Increased revenue through more accurate application of tax laws (e.g., property tax assessment and administration).

3. Reduced cost of service delivery through more accurate geographic analysis (e.g., fewer non-residents approved for resident-only services, better fraud detection, fewer new facilities serving smaller than anticipated catchment populations).

4. Reduced cost of address management through the elimination of duplicate data, duplicate systems, and duplicate maintenance.

5. Increased sharing of address data across government.

1.6.2 Standards Context

The Enterprise Geocoding Conceptual Model was derived from the original AddressBC conceptual data model [R14], the Canada Post Addressing Guidelines [R1], the BC Enterprise Civic Geocoder User Guide [R16], and the address model published by James Rumbaugh [R11]. The following address models and guidelines were also consulted:

- BC Mailing and Delivery Address standard [R7]
- Nova Scotia Civic Address Guide[R3]
- URISA Street Address Data Standard[R13]
- United States Postal Addressing Standards[R9]
- UK Spatial Datasets For Geographical Referencing B7666-2006[R10]
- ESRI Address Data Model for the City of Calgary[R12]
- ISO TC211 address standards proposals[R8]
- ISO 19112 – Geographic information – Spatial Referencing by geographic identifiers [R20]
- OpenGIS Location Service (OpenLS) Implementation Standards [R15]

What distinguishes the geocoding conceptual model from the Address BC Data Model and the BC Mailing and Delivery Address Standard?

1. The geocoding conceptual model is focused on the determination of the location (coordinates) of physical addresses. Address elements are based on Canada Post standards where applicable.

2. The Address BC Data Model is focused on the management of civic addresses. Address elements are based on URISA (United States Postal Service) standards.

3. The BC Mailing and Delivery Address standard defines the address needed to get mail and parcels delivered, how to display and validate an address in an application, and how to print address labels. Address elements are based on Canada Post standards.

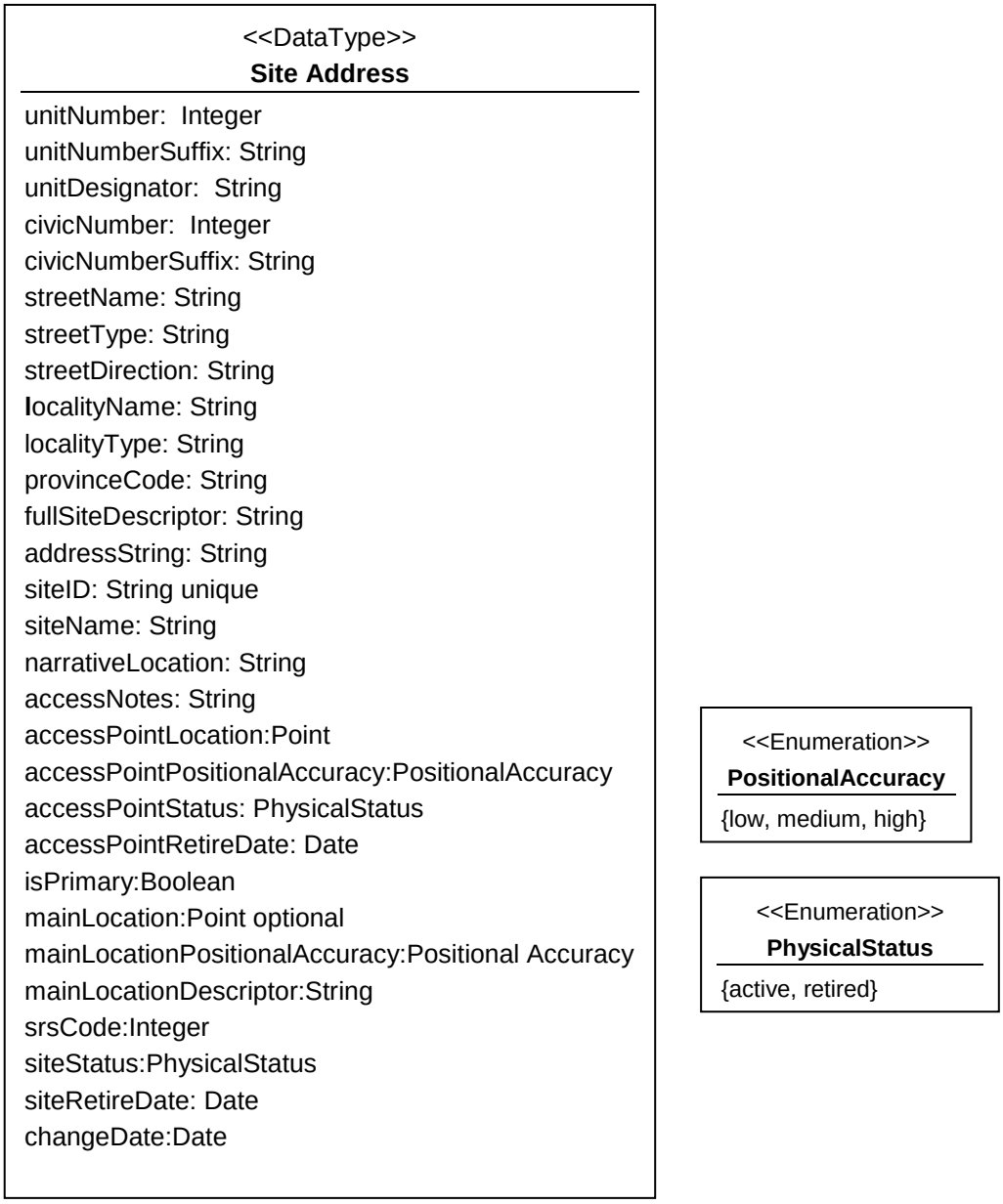
1.7 Related Standards

The Physical Address and Geocoding Interoperability Standards, to be released in fiscal 2010/2011, will define the required and optional web service APIs of a compliant geocoder.

2. INFORMATION MODEL

2.1 Site Address

2.1.1 DataType Diagram



2.1.2 DataType Definition

A **SiteAddress** represents the relative and direct location of a site on the earth. Direct location is represented by points in a coordinate reference system. Relative location is represented by a civic or non-civic address.

A civic address is a site address that includes a civic number, street name, and municipality. Civic numbers and street names are created and assigned by a municipality for use in emergency response, policing, garbage collection, enumeration, tax collection, and other municipal services. Here is an example of a civic address assignment guideline:

> "Structures with more than one principal use or occupancy should have a separate civic number for each primary outside entrance." [R4]

While interest in extending civic addressing into rural areas is growing, many rural sites in BC still have non-civic addresses. A non-civic address is represented by a site name, locality, and province and includes most landmarks, log cabins located on a named mountain, and residences on named islands without communities. The locality of a non-civic address can be an unincorporated subdivision or community, Indian reserve, named natural feature, regional district, or aboriginal lands.

Site Address is modelled as a DataType (or structured literal). A data type is a type that represents a set of immutable values (constants). A value is immutable and has no identity other than itself. A DataType can be simple as in an integer or string type, or composite as in a complex number or date.

All site addresses are assumed to be located in British Columbia.

2.1.3 Attribute Definitions

unitNumber is the number of the unit, suite, or apartment within a house or building.

unitNumberSuffix usually takes the form of a letter (e.g., as in 1A) or a fraction (as in 1½).

unitDesignator is the type of unit(e.g., APT, SUITE, and UNIT) and is abbreviated if such an abbreviation is defined. The set of all unit designators MUST include those defined in [R1] by Canada Post and is the responsibility of GEOBC.

civicNumber is the official number assigned to a site by a municipality.

civicNumberSuffix usually takes the form of a letter (e.g., as in 103A) or a fraction (as in 103½).

streetName is the official name of the street recognized by a municipality (e.g., *Douglas* in 1175 Douglas Street). A *streetName* that starts with a directional is not abbreviated (e.g., North Park, not N Park)

streetType is the type of street as assigned by a municipality (e.g., the *ST* in 1175 DOUGLAS ST) and is abbreviated if such an abbreviation exists. The set of all street types MUST include those defined in [R1] by Canada Post and is the responsibility of GEOBC.

streetDirection is the abbreviated compass direction as defined in [R1] by Canada Post (e.g., *W* in 101 Burnside RD W) and BC civic addressing authorities . The complete list is C, E, N, NE, NW, SE, SW, and W.

There is no street predirectional in SiteAddress for several reasons:

1. It makes it easier for geocoding service clients. They know that a predirectional should always be included in the *streetName* parameter since there is no predirectional parameter. For example, given a street named "North Park", a geocoder client knows to put " **North Park**", not "**Park**" into the *streetName* parameter of a geocoding request.

2. It introduces unnecessary difficulty into the address matching process. Given a request to geocode a structured address that contains a street predirectional of "**North**" and a street name of "**Park**", a geocoder will not match these elements to a street with no predirectional and the name "**North Park**".

3. It is rarely used by municipalities. Restructuring the few streets that have a predirectional is a simple, unambiguous task for a geocoding system.

4. It is not compliant with Canada Post standards.

localityName is the name of the municipality, community, Indian reservation, subdivision, regional district, aboriginal lands, or natural feature the site is located in. Since this standard is about geocoding, not mail delivery, the locality of a civic address is that defined by the civic address authority, not Canada Post. A locality name that starts with a directional is not abbreviated (e.g., North Vancouver, not N Vancouver). Spelling of localities that are place names or natural feature names MUST match that published by the BC Geographical Names Information System [R2].

localityType can be a municipality, community, Indian reservation, subdivision, regional district, aboriginal lands, forward sortation area, or natural feature.

provinceCode is the ISO 3166-2 Sub-Country Code [R19] for British Columbia, which is BC.

addressString is the address as a single string with comma-separated names. addressString MUST be in one of the following formats (a term in square brackets is optional, a term in square brackets followed by an asterisk means zero or more times):

```
Format 1. [[unitDesignator unitNumber[unitNumberSuffix]] [siteName],]*
            civicNumber[civicNumberSuffix] streetName streetType [streetDirection],
            localityName, provinceCode
```

```
Format 2. [[unitDesignator unitNumber[unitNumberSuffix]] [siteName],]*
            localityName, provinceCode

Format 3. streetName streetType [streetDirection], localityName, provinceCode

Format 4. localityName, provinceCode
```

Here are some examples:

1. A civic address with no unit number:

   ```
   1025 HAPPY VALLEY RD, METCHOSIN, BC
   420 GORGE RD E, VICTORIA, BC
   130A HILL ST, NELSON, BC
   ```

2. A civic address with a unit number:

   ```
   UNIT 1, 433 CEDAR RAPIDS BLVD, PEMBERTON, BC
   PAD 2, 1200 NORTH PARK RD, SHAWNIGAN LAKE, BC
   ```

3. A civic address with a simple site name:

   ```
   PORT ALICE HEALTH CENTRE, 1090 MARINE DRIVE, PORT ALICE, BC
   ROYAL ATHLETIC PARK, 1014 CALEDONIA AVE, VICTORIA, BC
   ```

4. A civic address with a unit within a named complex:

   ```
   PAD 2, HAPPY MOBILE HOME PARK, 1200 NORTH PARK RD, SHAWNIGAN LAKE, BC
   ROOM 103A, CLEARIHUE BUILDING, UNIVERSITY OF VICTORIA, 3800 FINNERTY RD, VICTORIA, BC
   ROOM 230, WEST BLOCK, ROYAL JUBILEE HOSPITAL, 1952 BAY ST, VICTORIA, BC
   ```

5. A civic address with a unit within a named complex and the unit has both a number and a name:

   ```
   Cabin C Heron, Happy Fishing Resort, Whopper, BC
   ```

6. A civic address with a unit within a unit of a named complex:

   ```
   PAD 11, TERMINAL 3, BC SPACEPORT, 1 MILKY WAY, STAR CITY, BC
   ```

7. A street within a locality:

   ```
   WILLOW DRIVE, 70 MILE HOUSE, BC
   HORSE LAKE ROAD, 100 MILE HOUSE, BC
   ```

8. A locality within the province:

   ```
   PEACE RIVER REGIONAL DISTRICT, BC
   100 MILE HOUSE, BC
   PYPER LAKE, BC
   V8T, BC
   ```

fullSiteDescriptor is that portion of *addressString* that precedes the civic number (in the case of a civic address) or the locality (in the case of a non-civic address).

The *fullSiteDescriptor* of the civic address:

```
PAD 11, TERMINAL 3, BC SPACEPORT, 1 MILKY WAY, STAR CITY, BC
```

is

```
PAD 11, TERMINAL 3, BC SPACEPORT
```

The *fullSiteDescriptor* of the non-civic address:

```
Cabin C Heron, Happy Fishing Resort, Whopper, BC
```

is

```
Cabin C Heron, Happy Fishing Resort
```

Site Address contains the following additional site properties:

A *siteID* is a unique and immutable identifier assigned to every site in BC. Using this siteID, a compliant geocoding system can serve as a master data hub of sites and their addresses for all clients systems within government. See section 3.2.5 for details.

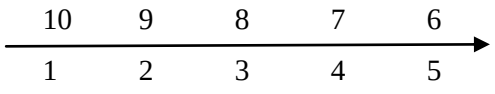
siteName is a string containing the name of the building, facility, or institution (e.g., Duck Building, Casa Del Mar, Crystal Garden, Bluebird House). A business name should only be used if it is permanently affixed to the site and the site has no other, more generic name. If a site is a unit within a complex, it may have a *sitename* in addition to a unitNumber and unitSuffix. *siteName* may be empty.

narrativeLocation contains written directions to access the site. The narrative should start at the closest known, named physical feature to the site (e.g. from Tlell, travel north on highway 16 to the big golden spruce tree on your left, hike west for about one kilometre).

accessNotes contain additional information that is helpful in determining the access location and any restrictions on mode of access (e.g., boat only, floatplane only).

accessPointLocation is the point along the street centreline that is even with the site's access lane or driveway if there is vehicle access into the site. If there is no vehicle access, the point is even with the site's front walk.

A street centreline is a line that is parallel to the street edges and located in the middle of the roadway. A street centreline exists even on streets that have no painted lines on them (e.g., a residential or unpaved street). A street centreline is oriented in the direction of increasing civic numbers. In the case of blocks that have the following type of continuous numbering:



the street centreline is oriented in the direction of the lowest block range, 1-5 in this example.

accessPointPositionalAccuracy is **high** if the point position was observed or measured using GPS, survey instruments, or photogrammetric methods; **medium** if point position was interpolated along a block face range by a geocoder or digitized off a paper map, web map or an orthophoto; and **low** if the access point was assigned the representative point of a street, locality, or province. For further details on positional accuracy, see section 3.2.4

accessPointStatus is status of the access point (**active**, or **retired**). A civic access point is usually only Retired if the associated site is renumbered, destroyed, or combined with another site. A non-civic access point is usually retired when its associated site is assigned a civic number or the site becomes unoccupied.

accessPointRetireDate is the date the access point was retired.

isPrimary is true if this is the primary (or official) access point of the associated site; false otherwise.

mainLocation is the point geometry of the rooftop, front door, or centre of the property.

mainLocationPositionalAccuracy is **high** if the point position was observed or measured using GPS, survey instruments, or photogrammetric methods; **medium** if point position was interpolated along a block face range by a geocoder or digitized off a paper map, web map or an orthophoto; and **low** if the access point was assigned the representative point of a street, locality, or province. For further details on positional accuracy, see section 3.2.4

mainLocationDescriptor describes the nature of the main location. Values include **rooftop, front door, centre,** and **insideBoundary**.

srsCode is the code [R19] of the spatial referencing system that *accessPoint* and *mainLocation* are defined in. Compliant geocoders must support, at minimum, lat/lon(4326), BC Albers(3005), and NAD83 UTM (26907-26911). Additional codes may be supported.

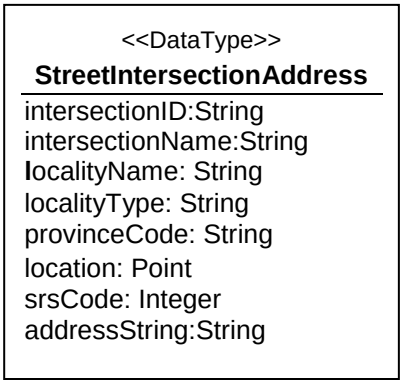
status is the status of the site (**active**, or **retired**). A site is usually **retired** when it is destroyed or combined with another site.

siteRetireDate is the date the site was retired.

changeDate is the date the site address was last changed.

2.2 Street Intersection Address

2.2.1 DataType Diagram



2.2.2 DataType Definition

A **StreetIntersectionAddress** represents the relative and direct location of the place where two or more streets meet or cross. Direct location is represented by a point in a coordinate reference system. Relative location is represented by the names of all the streets that meet or cross at the intersection.

Many provincial agencies have business events that happen at street intersections. For example, ICBC and BC Ambulance are interested in traffic accidents and these usually happen at intersections.

2.2.3 Attribute Definitions

intersectionID is a unique and immutable identifier assigned to every street intersection in the province.

intersectionName is the concatenation of the names of all streets that meet or cross at the intersection. Names appear in alphabetical order and are separated by the word “and” (e.g., “Blueforest Ave E and Elm St”). A typical street intersection involves two street names (e.g., “7th Ave and Union St”) but three or more names are possible (e.g., “Gorge Rd E and Gorge Rd W and Harriet Rd”, “Douglas St and Gorge Rd E and Government St and Hillside Ave”)

localityName is the name of the municipality, community, Indian reservation, subdivision, regional district, or aboriginal lands the street intersection is located in. Since this standard is about geocoding, not mail delivery, the locality of a street intersection is that defined by the civic address authority, not Canada Post. A locality name that starts with a directional is not abbreviated (e.g., North Vancouver, not N Vancouver).

localityType can be a municipality, community, Indian reservation, subdivision, regional district, or aboriginal lands.

provinceCode is the ISO 3166-2 Sub-Country Code [R19] for British Columbia, which is BC.

addressString is the address as a single string in the following format:

intersectionName, localityName, provinceCode

Some examples are: "7th Ave and Union St, New Denver, BC" and

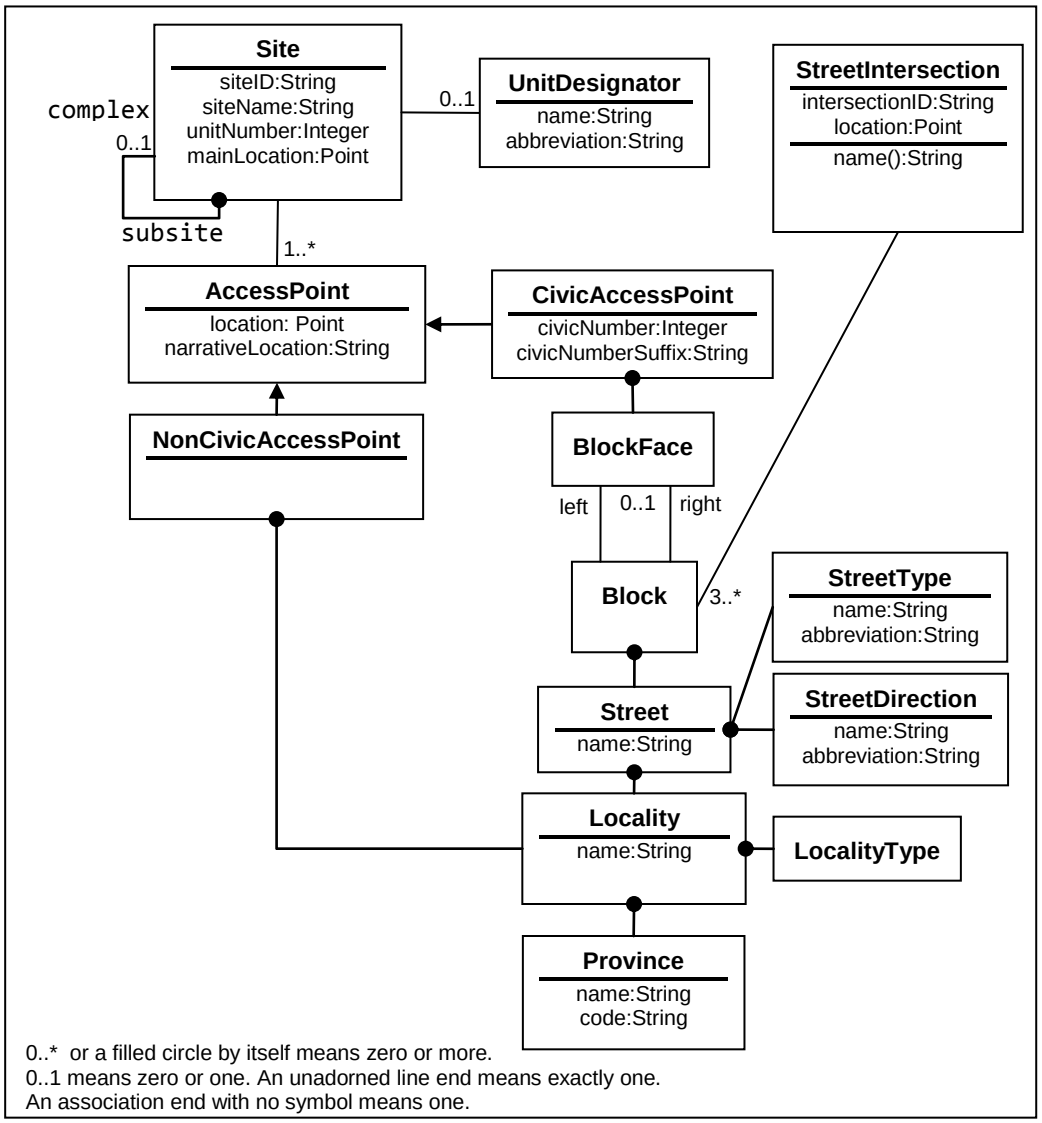
"Gorge Rd E and Gorge Rd W and Harriet Rd, Victoria, BC"

location is a point within the street intersection, preferably the spatial intersection of all road centrelines that meet or cross, if such a point exists; otherwise, a point known to be within the intersection boundary, preferably the centroid.

srsCode is the code [R19] of the spatial referencing system that *location* is defined in. Compliant geocoders must support, at minimum, lat/lon(4326), BC Albers(3005), and NAD83 UTM (26907-26911). Additional codes may be supported.

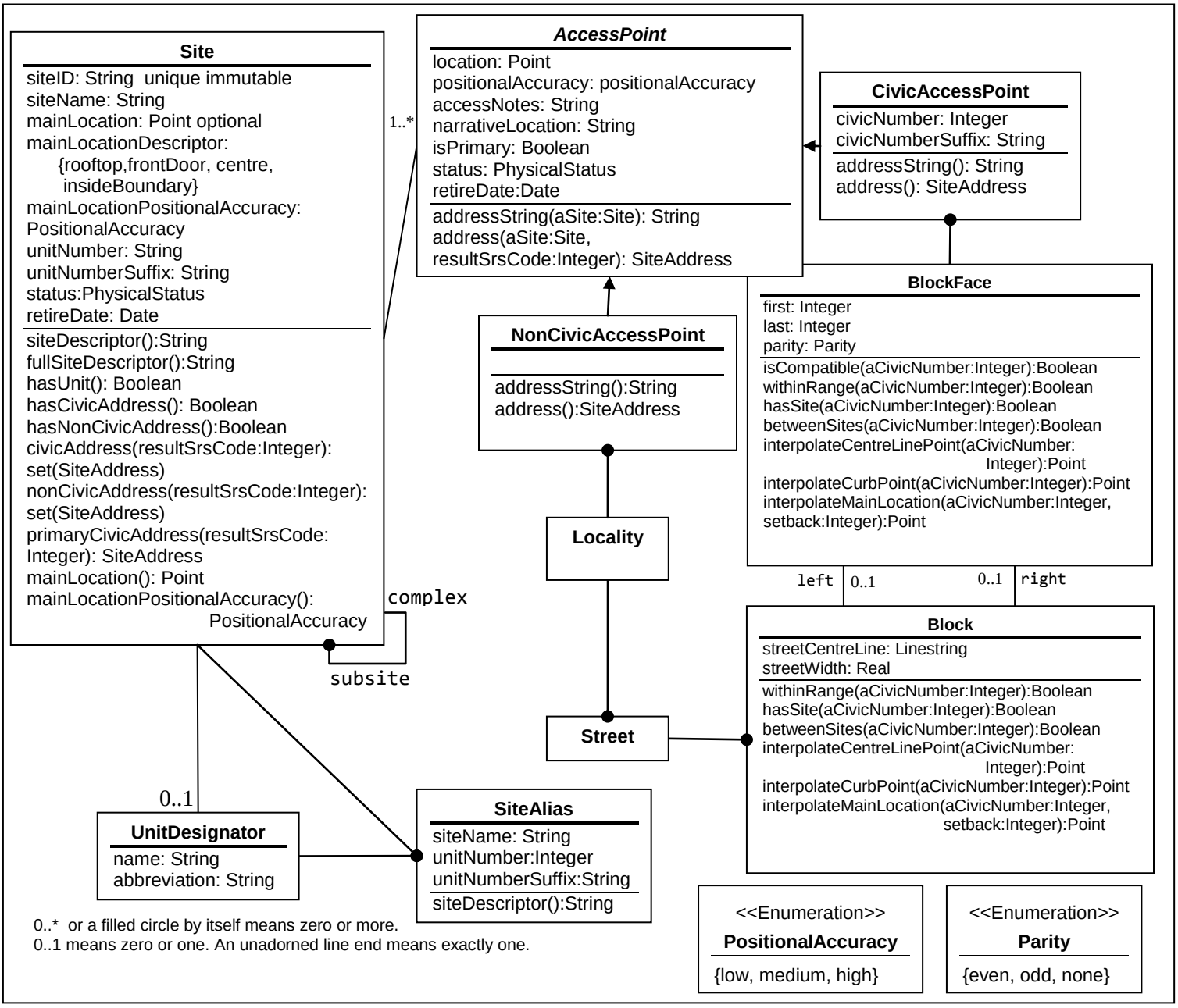
2.3 Site, Street Intersection, and Related Classes

2.3.1 Class Diagram – High Level

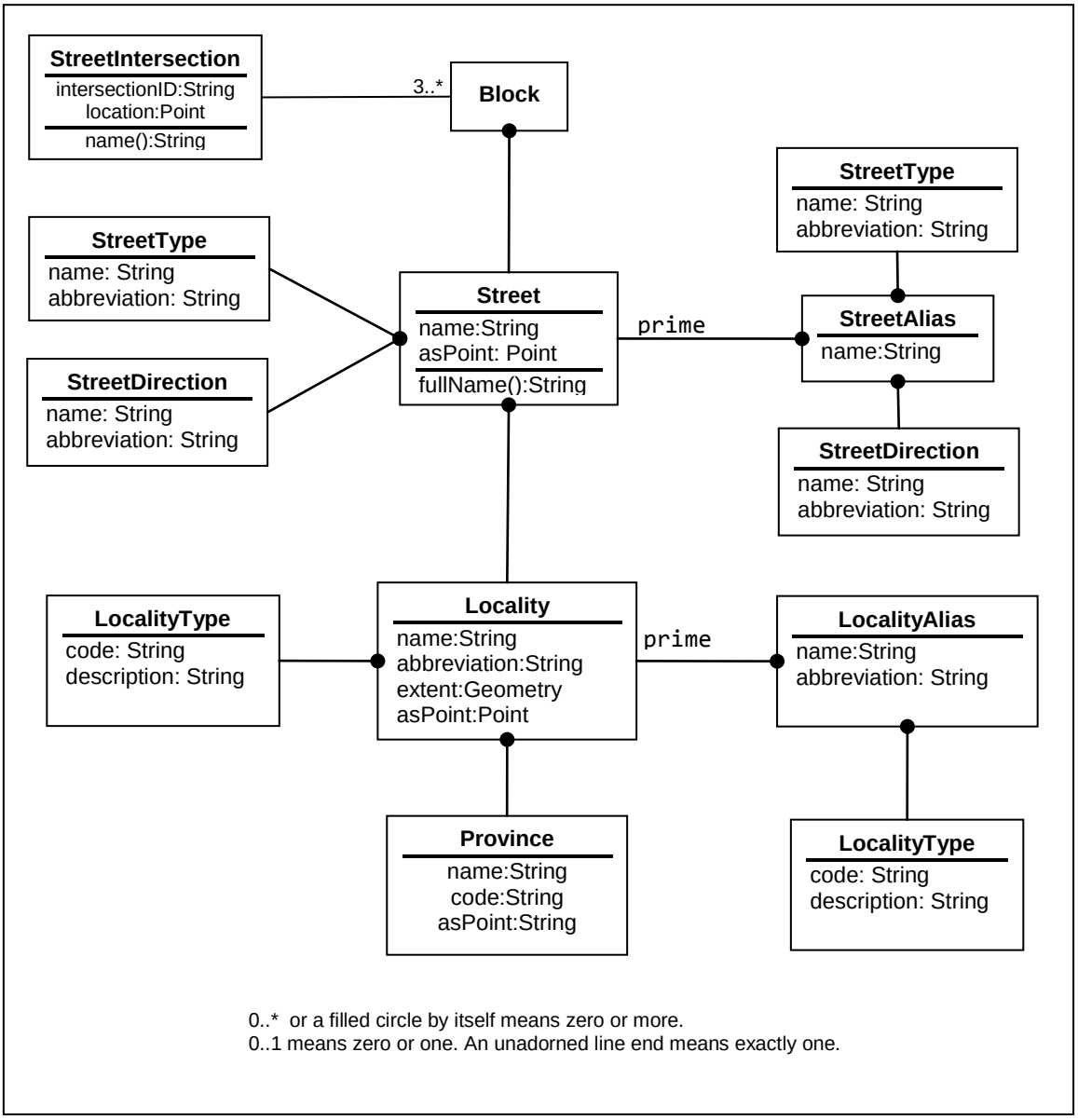


2.3.2 Class Diagrams – Detail

2.3.2.1 Site, AccessPoint, BlockFace, Block



2.3.2.2 Street, StreetIntersection, Locality



2.3.3 Class Definitions

A Site is a constructed geographic feature in British Columbia with known coordinates (latitude/longitude) and a physical address that is needed in the conduct of provincial business. Examples of constructed geographic features include houses, cabins, permanent campsites, mobile home and RV parks, units within houses and buildings, buildings within complexes, stores, malls, offices, industrial plants, bandshell, golf courses, hospitals, universities, recreation centres, places of worship, parks, municipal pumping stations, hydro sub-stations, wharves, airports, train stations, and landmarks.

A site can be assigned one or more AccessPoints, one of which is primary. A CivicAccessPoint associates a site with a street block face. A NonCivicAccessPoint associates a site with a natural feature or administrative area and is used where no civic number has been assigned. AccessPoints for a given site can change over time. Occasionally, the same civic number will be assigned to more than one site.

A site may have multiple SiteAliases that represent former or alternate site names or units. For example, a Site with a *siteName* of "Vancouver Island University" would have a SiteAlias with the *siteName* "Malaspina College".

A Street has one or more Blocks. A Block is bounded by two intersections and may have a left BlockFace, a right BlockFace, or both left and right BlockFaces.

StreetType is the set of all valid street types(e.g., ST, RD). This set MUST include those defined in [R1] by Canada Post and is the responsibility of GEOBC.

StreetDirection is the set of all valid street directions and their abbreviations as defined in [R1] by Canada Post and by BC civic address authorities. The complete list is: Central/C, East/E, North/N, Northeast/NE, Northwest/NW, South/S, Southeast/SE,Southwest/ SW, and West/W.

StreetAlias is a former or alternate, unofficial name, abbreviation, streetType, and streetDirection of a Street. A StreetAlias has no blocks of its own, sharing those of its prime.

StreetIntersection is the set of all street intersections in the province.

Unit Designator is the set of all valid types of unit(e.g., APT, SUITE, and UNIT). This set MUST include those defined in [R1] by Canada Post and is the responsibility of GEOBC.

Locality is the geographic area a site is located in or near. A locality has a localityType and any number of localityAliases.

LocalityType is the set of all valid locality types. The following types MUST be supported: municipality, community, subdivision, Indian Reservation, regional district, aboriginal lands, forward sortation area, and natural feature. Additional types, such as regional district electoral area, MAY be supported.

LocalityAlias is an alternate or former name of a Locality. A LocalityAlias has no Streets of its own, sharing those of its prime. It may have its own localityType as in a named neighbourhood acting as an alias for a municipality. For example, Victoria is a locality with a localityType of "municipality" and Fairfield is a localityAlias of Victoria with a localityType of "neighbourhood". A neighbouring populated place can also act as a localityAlias.

Province is the set of names and ISO 3166-2 Sub-country codes [R19] for the provinces and territories in Canada.

2.3.4 Attribute Definitions By Class

2.3.4.1 Site

siteID is a unique and immutable identifier assigned by a compliant geocoding system and can be used by all client systems to maintain the currency of their site addresses.

siteName is a string containing the name of the building, facility, or institution (e.g., Duck Building, Casa Del Mar, Crystal Garden, Bluebird House). A business name is only used if it is permanently affixed to the site and the site doesn't have a more generic name. If a site is a unit within a complex, it may have a sitename in addition to a unitNumber and unitSuffix. *siteName* may be empty.

mainLocation is the point geometry of the rooftop, front door, or centre of the property.

mainLocationDescriptor describes the nature of the main location. Values include **rooftop, frontDoor, centre,** and **insideBoundary**.

mainLocationPositionalAccuracy is **high** if the point position was observed or measured using GPS, survey instruments, or photogrammetric methods; **medium** if point position was interpolated along a block face range by a geocoder or digitized off a paper map, web map or an orthophoto; and **low** if the access point was assigned the representative point of a street, locality, or province. For further details on positional accuracy, see section 3.2.4

unitNumber is the number of the unit, suite, or apartment within a house or building.

unitNumberSuffix usually takes the form of a letter (e.g., as in 1A) or a fraction (as in 1½).

status is the status of the site (**active**, or **retired**). A site is usually **retired** when it is destroyed or combined with another site.

retireDate is the date the site was retired.

Access points and main locations of all sites must be in the same spatial referencing system.

2.3.4.2 SiteAlias

siteName is an alternate or former name of an associated Site.

unitNumber is an alternate or former unit number of an associated Site.

unitNumberSuffix is an alternate or former unit number suffix of an associated Site.

2.3.4.3 AccessPoint

location is the point along street centreline that is even with the site's access lane or driveway if there is vehicle access into the site. If there is no vehicle access, the point is even with the site's front walk.

positionalAccuracy is **high** if the point position was observed or measured using GPS, survey instruments, or photogrammetric methods; **medium** if point position was interpolated along a block face range by a geocoder or digitized off a paper map, web map or an orthophoto; and **low** if the access point was assigned the representative point of a street, locality, or province. For further details on positional accuracy, see section 3.2.4

accessNotes contain additional information that is helpful in determining the access location and any restrictions on mode of access (e.g., boat only, floatplane only).

narrativeLocation contains written directions to access the site. The narrative should start at the closest known, named physical feature to the site (e.g. from Tlell, travel north on highway 16 to the big golden spruce tree on your left, hike west for about one kilometre).

isPrimary is true if this is the primary access point of the associated site; false otherwise. If a site has both civic and non-civic address points, one of the civic access points is usually primary.

status is status of the access point (**active**, or **retired**). A CivicAccessPoint is usually only **retired** if the associated site is renumbered, destroyed, or combined with another site. A NonCivicAccessPoint is usually **retired** when its associated site is assigned a civic number or the site becomes unoccupied.

retireDate is the date the access point was retired.

Access points and main locations of all sites must be in the same spatial referencing system.

2.3.4.4 CivicAccessPoint

civicNumber is the official number assigned to a site by a municipality. More than one site can have the same civic number.

civicNumberSuffix usually takes the form of a letter (e.g., as in 103A) or a fraction (as in 103½).

2.3.4.5 BlockFace

first is the first civic number on the block face looking in the direction of the street centreline.

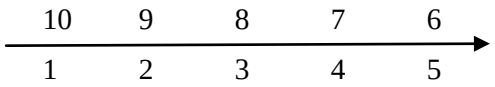
last is the last civic number on the block face looking in the direction of the street centreline.

Address ranges may be set to the theoretical values assigned by the civic addressing authority or set to the lowest and highest observed civic numbers on the block face.

parity defines the parity of the civic numbers on the block face: even, odd, or none. A parity of none means continuous numbering.

A block may have a *left* block face, a *right* block face, or both *left* and *right* block faces. In the case of a block on a divided highway, two blocks would be defined, one block with a left block face, and one block with a right block face.

In the case of a block that has the following type of continuous numbering:

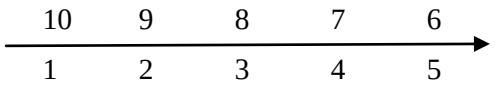


The block will have both left and right block faces. The left block face will have *first*=10, *last*=6, and *parity*= **none** while the right block face will have *first*=1, *last*= 5, and a *parity*=**none**

2.3.4.6 Block

streetWidth is the width of the street measured from edge to edge (or curb to curb) and in metres.

streetCentreLine is a line that is parallel to the street edges and located in the middle of the roadway. A street centreline exists even on streets that have no painted lines on them (e.g., a residential or unpaved street). A street centreline is oriented in the direction of increasing civic numbers. In the case of blocks that have the following type of continuous numbering:

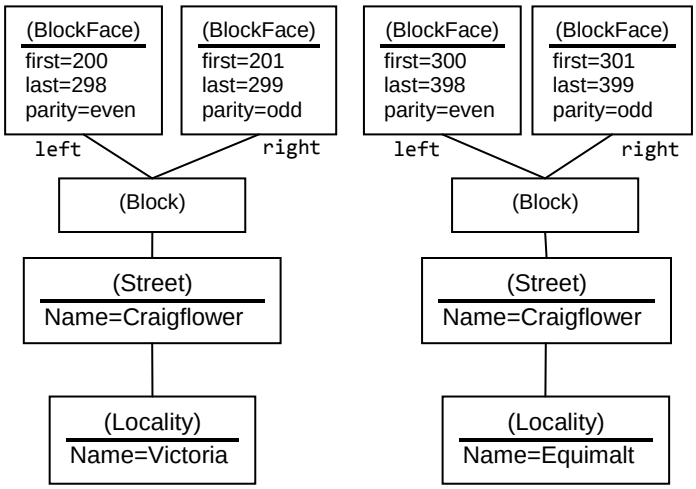


the centreline is oriented in the direction of the ascending civic numbers (e.g., one to five).

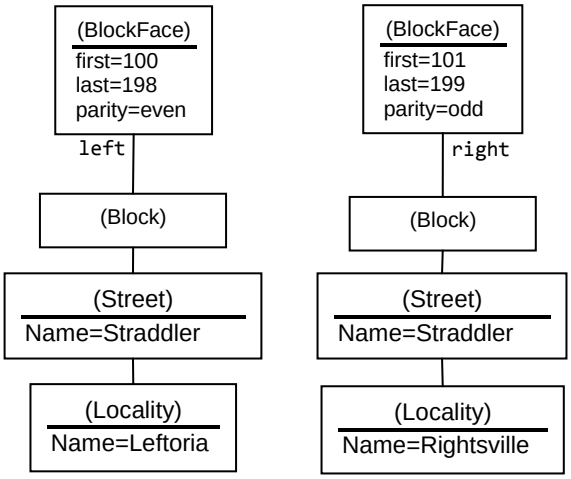
2.3.4.7 Street

A Street has a *name*, a representative point (*asPoint*), which is used when an address doesn't match to a site or block, a *streetType*, and a *streetDirection*.

A Street must be contained in a single municipality. If a street crosses a municipal boundary, it should be modelled as separate Street and Block objects in each municipality as follows:



If a street straddles a municipal boundary, it should be modelled so that the left block face is in the municipality on the left and the right block face is in the municipality on the right as shown in the following diagram:



2.3.4.8 StreetAlias

name is an alternate or former street name.

2.3.4.9 StreetIntersection

intersectionID is a unique and immutable identifier assigned to every street intersection in the province.

location is a point within the street intersection, preferably the spatial intersection of all road centrelines that meet or cross, if such a point exists; otherwise, a point known to be within the intersection boundary, preferably the centroid.

2.3.4.10 Locality

name is the name of the locality. A *name* that starts with a directional is not abbreviated (e.g., North Vancouver, not N Vancouver). Correct spelling of official populated place names and natural features is defined by the BC Geographical Names Information System [R3].

abbreviation is the abbreviated name of the locality.

extent is the geographic extent (i.e., boundary) of the locality. It is usually a polygon but may also be a linestring or, rarely, a point.

asPoint is a representative point within an areal or linear extent (e.g., the location of the municipal hall).

When a site isn't located near a street with address ranges, it may be located in or near a named, natural feature. If the natural feature is about the size of a populated place, it can serve as a locality for the site. For example, Piers Island and Elk Lake could serve as localities but not Vancouver Island and Williston Lake. Named landmarks and other cultural features can also serve as localities if they are small enough.

2.3.4.11 LocalityAlias

name is an alternate or former street name of a locality or the name of a neighbouring locality.

abbreviation is the abbreviation of *name*.

2.3.4.12 Province

A Province has a *name*, a *code* as defined in ISO-3166-2 Sub-Country Codes [R19], and a representative point (*asPoint*).

2.3.5 Relationship between Site Address and Site

A Physical Address contains a set of geographic identifiers that are derived, in a complex way, by the geocoder from Site and associated classes. The following table illustrates the basic correspondence:

Site Address Element	Site Element
siteID	site.siteID
siteName	site.siteName
unitNumber	site.unitNumber
unitNumberSuffix	site.unitNumberSuffix
unitDesignator	site.unitDesignator
civicNumber	site.civicAccessPoint.civicNumber
civicNumberSuffix	site.civicAccessPoint.civicNumberSuffix
streetName	site.civicAccessPoint.blockFace.block.street.name
streetType	site.civicAccessPoint.blockFace.block.street.streetType.abbreviation
streetDirection	site.civicAccessPoint.blockFace.block.street.streetDirection.abbreviation
locality	site.civicAccessPoint.blockFace.block.street.locality.name **OR** site.nonCivicAccessPoint.locality.name
province	site.civicAccessPoint.blockFace.block.street.locality.province.code **OR** site.nonCivicAccessPoint.locality.province.code

The precise derivation of site address is given in sections 2.3.6.3 – 2.3.6.5.

In short, a site address, as is commonly and incorrectly modelled as a single class, is a highly denormalized view of Site and its associates. Put another way, an address is the name of a site and not the site itself.

2.3.6 Operations by Class

Class operations are defined in Object Constraint Language 2.0 notation.

2.3.6.1 Site

```
context Site::hasUnit():Boolean
--- returns true if site has a unit number assigned
post: result = unitNumber>0


context Site::unitString(): String
post: result = if hasUnit() then
                  if unitDesignator.notEmpty() then
                     if unitDesignator.abbreviation<>”” then
                        unitDesignator.abbreviation
                     else
```

```
                        unitDesignator.name
                     endif
                  else
                     "UNIT"
                  endif
                  + unitNumber.asString + unitNumberSuffix + " "
               else
                 “”
               endif


context Site::siteDescriptor(): String
--- returns unit and site name as a single string
post: result = if hasUnit() then
                  if siteName=“” then
                     unitString()
                  else
                     unitString() + “ “ + siteName
                  endif
               else
                  siteName
               endif


context Site::fullSiteDescriptor(): String
--- returns fully qualified site descriptor
post: result = if complex->isEmpty() then
                  siteDescriptor()
               else
                  if siteDescriptor()=“” then
                     complex.fullSiteDescriptor()
                  else
                     if complex.fullSiteDescriptor()=“” then
                        siteDescriptor()
                     else
                         siteDescriptor() + “, “ + complex.fullSiteDescriptor()
                     endif
                  endif
               endif


context Site::hasCivicAddress():Boolean
--- returns true if site has a civic address
post: result = if not civicAccessPoint->isEmpty then
                  true
               elseif not complex.isEmpty() then
                   complex.hasCivicAddress()
               else
                 false
               endif


context Site::directCivicAddress(aSite:SITE, resultSrsCode:INTEGER):set(SiteAddress)
--- returns the set of all civic addresses directly associated with aSite
--- resultSrsCode specifies the code of the SpatialReferenceSystem that SiteAddresses
---           must be projected into

---  resultSrsCode must be the code of a supported spatial reference system
pre: spatialReferenceSystem->exists(s| s.code=resultSrsCode)
```

```
post:  civicAccessPoint->isEmpty implies result->isEmpty

post:  result->notEmpty implies
         result->forAll(pa| civicAccessPoint->exists(x| pa =
                            x.address(aSite:Site, geocoder.baseSRS.code, resultSrsCode)


context Site::indirectCivicAddress(aSite:Site, resultSrsCode:INTEGER):set(SiteAddress)
--- returns the set of all civic addresses indirectly
--- associated with aSite
--- resultSrsCode specifies the code of the SpatialReferenceSystem that SiteAddresses
---           must be projected into

---  resultSrsCode must be the code of a supported spatial reference system
pre: spatialReferenceSystem->exists(s| s.code=resultSrsCode)

pre: hasCivicAddress()

post: result = if not civicAccessPoint->isEmpty then
                  directCivicAddress(aSite)
               else
                   complex.indirectCivicAddress(aSite)
               endif


context Site::civicAddress(resultSrsCode:INTEGER):set(SiteAddress)
--- returns the set of all civic addresses either directly or indirectly
--- associated with a site
--- resultSrsCode specifies the code of the SpatialReferenceSystem that SiteAddresses
---           must be projected into

---  resultSrsCode must be the code of a supported spatial reference system
pre: spatialReferenceSystem->exists(s| s.code=resultSrsCode)

pre: hasCivicAddress()

post: result = if civicAccessPoint->notEmpty then
                  directCivicAddress(self, resultSrsCode)
               else
                   complex.indirectCivicAddress(self, resultSrsCode)
               endif


context Site::primaryAddress(resultSrsCode:INTEGER):SiteAddress
--- returns the primary address directly or indirectly associated with a site
--- resultSrsCode specifies the code of the SpatialReferenceSystem that SiteAddress
---           must be projected into

---  resultSrsCode must be the code of a supported spatial reference system
pre: spatialReferenceSystem->exists(s| s.code=resultSrsCode)

pre: hasCivicAddress()

post: result = if hasCivicAddress()then
                  civicAddress(resultSrsCode)->select(isPrimary)->asSequence->at(1)
               else
                   nonCivicAddress(resultSrsCode)->select(isPrimary)->asSequence->at(1)
               endif


context Site::hasNonCivicAddress():Boolean
```

```
--- returns true if site has a non-civic address
post: result = if not nonCivicAccessPoint->notEmpty then
                  true
               elseif not complex.isEmpty() then
                   complex.hasNonCivicAddress()
               else
                 false
               endif


context Site::directNonCivicAddress(aSite:SITE, resultSrsCode:INTEGER):set(SiteAddress)
--- returns the set of all non-civic addresses directly associated with aSite
--- resultSrsCode specifies the code of the SpatialReferenceSystem that SiteAddresses
---           must be projected into

---  resultSrsCode must be the code of a supported spatial reference system
pre: spatialReferenceSystem->exists(s| s.code=resultSrsCode)

post:  nonCivicAccessPoint->isEmpty implies result->isEmpty

post:  result->notEmpty implies
         result->forAll(pa| nonCivicAccessPoint->exists(x| pa =
                           x.address(aSite:Site, geocoder.baseSRS.code, resultSrsCode)


context Site::indirectNonCivicAddress(aSite:Site, resultSrsCode:INTEGER):set(SiteAddress)
--- returns the set of all non-civic addresses indirectly associated with aSite
--- resultSrsCode specifies the code of the SpatialReferenceSystem that SiteAddresses
---           must be projected into

---  resultSrsCode must be the code of a supported spatial reference system
pre: spatialReferenceSystem->exists(s| s.code=resultSrsCode)


pre: hasNonCivicAddress()
post: result = if not nonCivicAccessPoint->isEmpty then
                  directNonCivicAddress(aSite, resultSrsCode)
               else
                   complex.indirectNonCivicAddress(aSite, resultSrsCode)
               endif


context Site::nonCivicAddress(resultSrsCode:INTEGER):set(SiteAddress)
--- returns the set of all non-civic addresses either directly or indirectly
--- associated with a site
--- resultSrsCode specifies the code of the SpatialReferenceSystem that SiteAddresses
---           must be projected into

---  resultSrsCode must be the code of a supported spatial reference system
pre: spatialReferenceSystem->exists(s| s.code=resultSrsCode)

pre: hasCivicAddress()

post: result = if nonCivicAccessPoint->notEmpty then
                  directNonCivicAddress(self, resultSrsCode)
               else
                   complex.indirectNonCivicAddress(self, resultSrsCode)
               endif
```

```
context Site::mainLocation():Point
--- returns the front door or rooftop location directly or indirectly
--- associated with a site
post: result = if not mainLocation.isNull() then
                  mainLocation
               elseif not complex.isEmpty() then
                      complex.mainLocation()
               else
                  null
               endif


context Site::mainLocationPositionalAccuracy():Point
--- returns the front door or rooftop location directly or indirectly
--- associated with a site
post: result = if not mainLocation.isNull() then
                  mainLocationPositionalAccuracy
               elseif not complex.isEmpty() then
                      complex.mainLocationPositionalAccuracy()
               else
                  null
               endif



context Site::allSubSites():set(Site)
--- returns all subsites directly or indirectly associated with a given site
post: result = if self.subSite->notEmpty then
                  self.subSite->union(subSite->collect(s:Site| s.allSubSites))
               else
                  self.subSite
               endif
```

2.3.6.2 SiteAlias

```
context SiteAlias::hasUnit():Boolean
--- returns true if site has a unit number assigned
post: result = unitNumber>0


context SiteAlias::unitString(): String
post: result = if hasUnit() then
                   if unitDesignator.notEmpty() then
                      if unitDesignator.abbreviation<>”” then
                         unitDesignator.abbreviation
                      else
                         unitDesignator.name
                      endif
                   else
                      "UNIT"
                   endif
                   + unitNumber.asString + unitNumberSuffix + " "
                else
                  “”
                endif
```

```
context SiteAlias::siteDescriptor(): String
--- returns unit and site name as a single string
post: result = if hasUnit() then
                  if siteName="" then
                     unitString()
                  else
                     unitString() + " " + siteName
                  endif
               else
                  siteName
               endif
```

2.3.6.3 AccessPoint

```
context AccessPoint::addressString(aSite:Site):String
--- returns address as a single string
 pre: site=aSite or site.allSubsites()->includes(aSite)


context AccessPoint::address(aSite:Site, resultSrsCode:Integer):SiteAddress
--- returns the address associated with aSite
--- resultSrsCode specifies the code of the SpatialReferenceSystem that SiteAddresses
---           must be projected into

---  resultSrsCode must be the code of a supported spatial reference system
pre: spatialReferenceSystem->exists(s| s.code=resultSrsCode)

pre: site=aSite or site.allSubSites()->includes(aSite)

post:  result.unitNumber = aSite.unitNumber and
       result.unitNumberSuffix = aSite.unitNumberSuffix and
       result.unitDesignator = aSite.unitDesignator.abbreviation and
       result.fullSiteDescriptor = aSite.fullSiteDescriptor() and
       result.addressString = addressString(aSite) and
       result.siteID = aSite.siteID and
       result.siteName = aSite.siteName and
       result.narrativeLocation = narrativeLocation and
       result.accessNotes = accessNotes and
       result.accessPointLocation =
                     geocoder.reproject(location, geocoder.baseSRS.code, resultSrsCode)and
       result.accessPointPositionalAccuracy = positionalAccuracy and
       result.accessPointStatus = status and
       result.accessPointRetireDate = retireDate and
       result.isPrimary = isPrimary and
       result.mainLocation =
           geocoder.reproject(aSite.mainLocation(), geocoder.baseSRS.code, resultSrsCode) and
       result.mainLocationPositionalAccuracy = aSite.mainLocationPositionalAccuracy() and
       result.mainLocationDescriptor = aSite.mainLocationDescriptor and
       result.spatialReferencingSystem = resultSrsCode and
       result.siteStatus = aSite.status and
       result.siteRetireDate = aSite.retireDate
```

2.3.6.4 CivicAccessPoint

```
context CivicAccessPoint::addressString(aSite:Site):String
```

```
--- returns civic address as a single string

--- preconditions inherited from AccessPoint::addressString

post: result = if aSite.fullSiteDescriptor()<>"" then
                  aSite.fullSiteDescriptor() + ","
               endif
               + civicNumber + civicNumberSuffix + " "
               + blockFace.block.street.streetName + " "
               + blockFace.block.street.streetType.abbreviation + " "
               + blockFace.block.street.streetDirection.abbreviation + ", "
               + blockFace.block.street.locality.name + ", "
               + blockFace.block.street.locality.province.code


context CivicAccessPoint::address(aSite:Site, resultSrsCode:Integer):SiteAddress
--- returns the civic address associated with a aSite

--- preconditions inherited from AccessPoint::address

--- in addition those defined here, postconditions are also inherited from AccessPoint::address

post:  result.civicNumber = civicNumber and
       result.civicNumberSuffix = civicNumberSuffix and
       result.streetName = blockFace.block.street.name and
       result.streetType = blockFace.block.street.streetType.abbreviation and
       result.streetDirection = blockFace.block.street.streetDirection.abbreviation and
       result.localityName = blockFace.block.street.locality.name and
       result.localityType = blockFace.block.street.locality.localityType.description and
       result.provinceCode = blockFace.block.street.locality.province.code and
```

2.3.6.5 NonCivicAccessPoint

```
context NonCivicAccessPoint::addressString(aSite:Site):String
--- returns non-civic address as a single string

--- preconditions inherited from AccessPoint::addressString

post: result = if aSite.fullSiteDescriptor()<>"" then
                  aSite.fullSiteDescriptor() + ","
               endif
               + locality.name + ", "
               + locality.province.code


context NonCivicAccessPoint::address(aSite:Site, resultSrsCode:Integer):SiteAddress

--- preconditions inherited from AccessPoint::address

--- in addition to postconditions defined here, postconditions are inherited from
AccessPoint::address

post:  result.localityName = locality.name and
       result.localityType = locality.localityType.description and
       result.provinceCode = locality.province.code and
```

2.3.6.6 Integer

```
context Integer::isEven()
--- Returns true if number is even
post: result = (self = self/2*2)


context Integer::isOdd()
--- Returns true if number is odd
Post: result = not isEven()
```

2.3.6.7 BlockFace

```
context BlockFace::isCompatible(aCivicNumber:Integer):Boolean
--- Returns true if value has the same or no parity as the block face

--- civicNumber valid
pre: civicNumber>0

post: result = parity=none
               or (parity=Parity::odd and aCivicNumber.isOdd())
               or (parity=Parity::even and aCivicNumber.isEven())


context BlockFace::withinRange(aCivicNumber:Integer):Boolean
--- Returns true if aCivicNumber is within address range and of compatible parity

--- civicNumber valid
pre: civicNumber>0

post: result = isCompatible(aCivicNumber)
               and first<=aCivicNumber and aCivicNumber<=last


context BlockFace::hasSite(aCivicNumber:Integer):Boolean
--- returns true if site with aCivicNumber exists on the block face

--- civicNumber valid
pre: aCivicNumber>0

post: result = civicAccessPoint->exists(x | x = aCivicNumber)


context BlockFace::betweenSites(aCivicNumber:Integer):Boolean
--- returns true if aCivicNumber lies between two adjacent site access points on the block face

--- civicNumber valid
pre: civicNumber>0

post: let n:orderedSet(Integer) = civicAccessPoint.civicNumber->orderBy(civicNumber)
      result = isCompatible(aCivicNumber) and not hasSite(aCivicNumber:Integer)
               and n->exists(x|x <= aCivicNumber and n->exists(y| aCivicNumber <=y and x<y) )


context BlockFace::interpolateAlongCentreLine(aCivicNumber:Integer): Point
--- returns a point interpolated along the street centre line between two adjacent points
---    A and B, in the block. A and B can be site access points or block face end points

--- civicNumber valid
```

```
pre: civicNumber>0

--- aCivicNumber must lie on a block face
pre: withinRange(aCivicNumber)


post: not Point.isNull()

--- Result is offset perpendicularly from the street centreline by half the street width
---   to approximate a point along the curb or street edge separating the site's front walk,
---   access road,or driveway from the street surface.



context BlockFace::interpolateAlongCurb(aCivicNumber:Integer): Point
--- returns a point interpolated along the curb between two adjacent points
---   A and B in the block face. A and B can be site access points or block face end points

--- civicNumber valid
pre: civicNumber>0

--- aCivicNumber must lie on a block face
pre: withinRange(aCivicNumber)


post: not Point.isNull()

--- Result is offset perpendicularly from the street centreline by half the street width
---   to approximate a point along the curb or street edge separating the site's front walk,
---   access road, or driveway from the street surface


context BlockFace::interpolateMainLocation(aCivicNumber:Integer, setBack:Integer): Point
--- Returns a point interpolated along the curb between two adjacent points
---   A and B in the block face. A and B can be site access points or block face end points
--- Interpolated point is also set back from the curb.

--- civicNumber valid
pre: civicNumber>0

--- aCivicNumber must lie on a block face
pre: withinRange(aCivicNumber)


post: not Result.isNull()

--- Result is offset perpendicularly from the street centreline by half the street width
---   plus a setback (in metres) to approximate a site's main location


context Block::withinRange(aCivicNumber:Integer):Boolean
--- Returns true if aCivicNumber is within the block range of the
---   associated left or right block face

--- civicNumber valid
pre: civicNumber>0

post:  Result = if left.notEmpty() then left.withinRange(aCivicNumber) else false endif
                or
                if right.notEmpty() then right.withinRange(aCivicNumber) else false endif
```

2.3.6.8 Block

```
context Block::hasSite(aCivicNumber:Integer):Boolean
--- returns true if site with aCivicNumber exists on the associated left or right block face

--- civicNumber valid
pre: aCivicNumber>0

post:  Result = if left.notEmpty() then
                    left.civicAccessPoint->exists(x | x = aCivicNumber)
                  else
                    false
                  endif
                or
                  if right.notEmpty() then
                    right.civicAccessPoint->exists(x | x = aCivicNumber)
                  else
                    false
                  endif


context Block::betweenSites(aCivicNumber:Integer):Boolean
--- returns true if aCivicNumber lies between two adjacent site access points on the
---    associated left or right block face

--- civicNumber valid
pre: civicNumber>0

post: result =   if left.notEmpty() then left.betweenSites(aCivicNumber) else false endif
               or
                 if right.notEmpty() then right.betweenSites(aCivicNumber) else false endif


context Block::interpolateAlongCentreLine(aCivicNumber:Integer): Point
--- Returns a point interpolated along the street centre line using the left or right block face

--- civicNumber valid
pre: civicNumber>0

--- aCivicNumber must lie on a block face
pre: withinRange(aCivicNumber)

post: result =   if left.notEmpty() then
                   left.interpolateAlongCentreLine(aCivicNumber)
                 else
                   right.interpolateAlongCentreLine(aCivicNumber)
                 endif


context Block::interpolateAlongCurb(aCivicNumber:Integer): Point
--- Returns a point interpolated along the curb of the left or right block face.
---   left or right block face

--- civicNumber valid
pre: civicNumber>0

--- aCivicNumber must lie on a block face
pre: withinRange(aCivicNumber)
```

```
post: result =   if left.notEmpty() then
                    left.interpolateAlongCurb(aCivicNumber)
                 else
                    right.interpolateAlongCurb(aCivicNumber)
                 endif


context Block::interpolateMainLocation(aCivicNumber:Integer, setBack:Integer): Point
--- Returns a point interpolated along and set back from the curb of the
---   left or right block face

--- civicNumber valid
pre: civicNumber>0

--- aCivicNumber must lie on a block face
pre: withinRange(aCivicNumber)


post: result =   if left.notEmpty() then
                    left.interpolateMainLocation(aCivicNumber,setBack)
                 else
                    right.interpolateMainLocation(aCivicNumber,setBack)
                 endif
```

2.3.6.9 Street

```
context Street::fullName():String
--- returns the full name of a street including street type and directional

post:  Result = name + " " + streetType.abbreviation + " " + streetDirection.abbreviation
```

2.3.6.10 StreetIntersection

```
context StreetIntersection::name():String
```

--- returns the the concatenation of the names of all streets that meet or cross at the intersection.

--- Names appear in alphabetical order and are separated by the word "and" (e.g., "Blueforest Ave E and Elm St").

--- A typical street intersection involves two street names (e.g., "7th Ave and Union St") but three or more names are possible.

--- Some examples are "Gorge Rd E and Gorge Rd W and Harriet Rd" and "Douglas St and Gorge Rd E and Hillside Ave"

2.3.7 Constraints

1. A site must have a single, primary access point.

   ```
   context Site inv hasPrimaryAccessPoint:
   site->accessPoint->select(isPrimary)->size=1
   ```

2. A unit number must be zero or greater. A value of zero indicates that no unit number has been assigned (see hasUnit() in section 2.3.6.1).

   ```
   context Site inv validUnitNumber:
   ```

```
unitNumber >=0
```

3. A site with a unit designator must have an assigned unit number.

```
context Site inv unitDesignatorHasUnit:
unitDesignator->size>0 implies hasUnit()
```

4. A civic number must be positive.

```
Context CivicAccessPoint inv validCivicNumber:
civicNumber>0
```

5. A civic number must lie within its block face range.

```
Context CivicAccessPoint inv civicNumberWithinBlockFaceRange:
blockFace.withinRange(civicNumber)
```

6. A civic number must have the correct parity for the block face it is on.

```
Context CivicAccessPoint inv civicNumberHasCorrectParity
blockFace.isCompatible(civicNumber)
```

7. A civic number and civic number prefix must remain constant for the lifetime of its CivicAccessPoint. If a site receives a new civic number, the CivicAccessPoint with the current civic number should be retired and a new CivicAccessPoint created (see section 2.7.8 for an example).

8. A block face with an even or odd parity must be have a positive and non-decreasing address range

```
Context BlockFace inv validBlockFaceRange:
not parity=Parity::none implies 0< first and first <= last
```

9. All geometry must be kept in the same spatial referencing system.

2.4 Geometry

All class properties in this conceptual model must be kept in the same spatial referencing system. All geometric data types used in this standard (e.g., Point, Polygon), spatial predicates (e.g., isValid()), and spatial operations (e.g., intersects()) conform to the OGC Simple Feature Access specification [R4].

2.5 Change Tracking

All classes in this conceptual model must have changes tracked in sufficient detail to reconstruct the state of an object on a given date. The Date type used in this standard conforms to the Stored Date Basic format of the BC Date and Time Standard [R18]. The Timestamp type used here conforms to the Stored Date and Time Combined format in [R18]

2.6 Data Element Concordance

The following is a concordance table showing the relationships between data elements of SiteAddress defined in this standard and the BC Mailing Delivery Address Standard [R7], and AddressBC[R14].

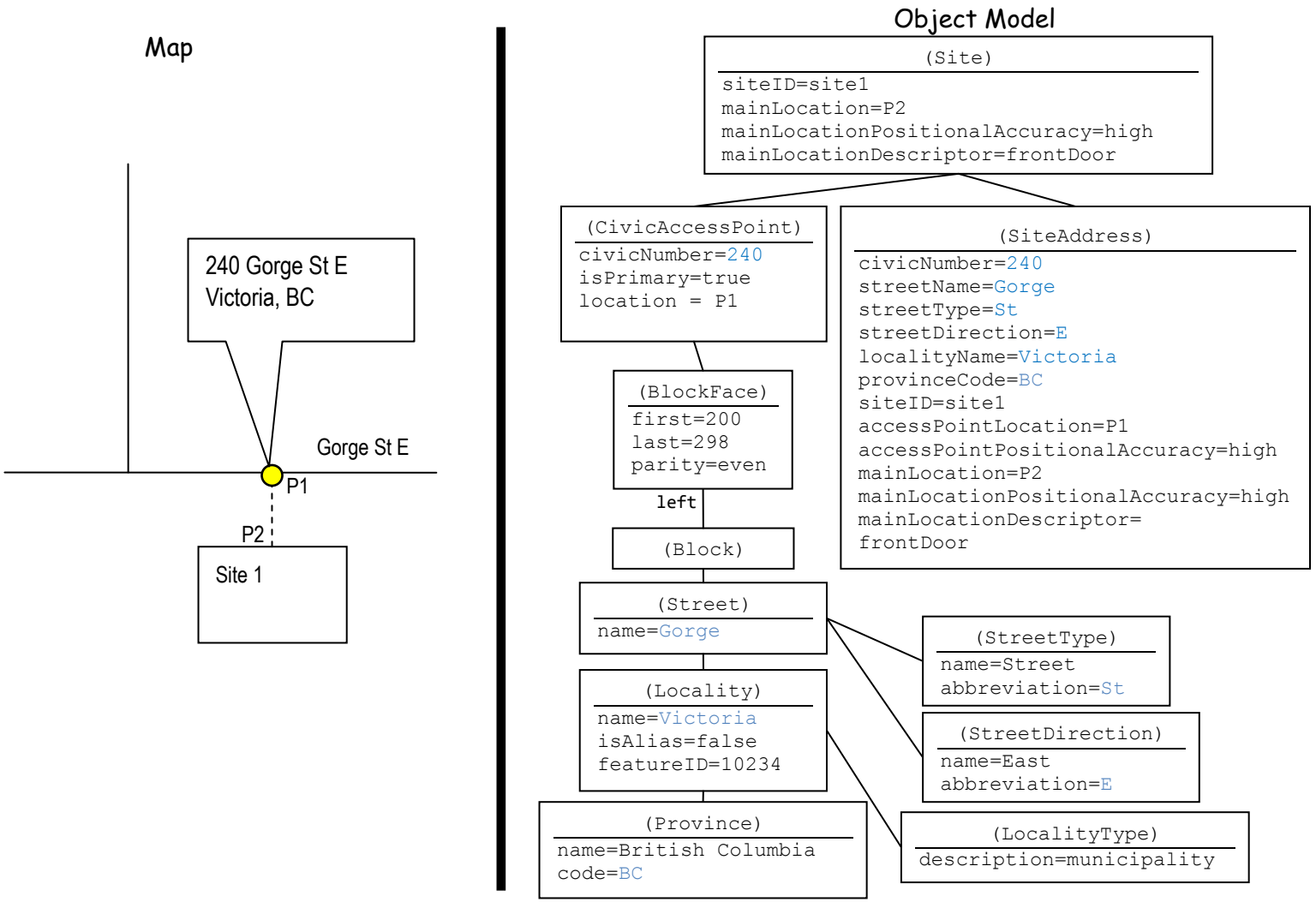
Site Address	Mailing and Delivery Address	AddressBC
unitNumber	Within Unit Identifier contains unitNumber + unitNumberSuffix	unitNumber
unitNumberSuffix	Within Unit Identifier contains unitNumber + unitNumberSuffix	unitNumberSuffix
unitDesignator	Within unit Designator	unitDesignator
N/A	N/A	Street Number Prefix
civicNumber	Civic number	Street Number
civicNumberSuffix	Civic number Suffix	Street Number Suffix
civicNumber + civicNumberSuffix	Civic number + Civic number suffix	Street Number + Street Number Suffix
streetName +streetType+streetDirection	Predirectional + Street name + Street type + Post-Directional	StreetName + Street Type + Street Dir Suffix
Incorporated into Street Name	Pre-directional	Incorporated into Street Name
streetName	Street name	Street Name
streetType	Street type	Street Type
streetDirection	Post-directional	Street Dir Suffix
locality (populated place for civic address, other locality types for non-civic address)	Municipality (any populated place)	Locality (municipality if appropriate; unincorporated area otherwise)
localityType (includes municipality, community, subdivision, Indian reservation, regional district, aboriginal lands, forward sortation area, and natural feature)	N/A	Muni ID not null means locality is a municipality; otherwise locality is an unincorporated area
province	Province	N/A (BC is assumed)
addressString	N/A	Full address
SiteID	N/A	Site ID
siteName	Building name	Building Name, Landmark Name
narrativeLocation	Detail Lines 1 – 4	Address Notes
accessNotes	Detail Lines 1 – 4	Alternate Access
accessPoint	N/A	Shape if Address Point Type set to “access”
accessPointPositionalAccuracy	N/A	N/A
mainLocation	N/A	Shape if Location Type set to mainLocation
mainLocationPositionalAccuracy	N/A	N/A
mainLocationDescriptor	N/A	Location Type
spatialReferencingSystem	N/A	N/A
isPrimary	N/A	N/A

A locality can be a forwardSortationArea which is first 3 characters of Postal Code	Postal Code	Postal Code
Incorporated into Street Name	Pre-directional	Incorporated into Street Name
N/A	Route Service Type	N/A
N/A	Delivery Installation Type	N/A
N/A	Delivery Installation Name	N/A
N/A	Delivery Installation Qualifier	N/A
N/A	Box Number	N/A
N/A	Lock Box Header	N/A
N/A	General Delivery Header	N/A
N/A	Route Service Number	N/A
N/A	Detail Lines 1 – 4	N/A
N/A	Delivery Instructions	N/A
N/A	International Postal Code	N/A
N/A	Country	N/A
N/A	US Pre-directional	N/A
N/A	US Post-directional	N/A
N/A	US Street Type	N/A
N/A	US WithinUnit Designator	N/A
N/A	US State	N/A
N/A	Zip Code	N/A
N/A	US Country	N/A
N/A	N/A	Geographic Domain
Locality.localityAlias and StreetAlias are used in address matching but are not part of Site Address	N/A	Locality Name Alias
N/A	N/A	Issuing Agency
N/A	N/A	Collection method
N/A	N/A	Location Type
N/A	N/A	Civic ID
N/A	N/A	Telus Community
locality	Municipality	Postal Community
narrativeLocation can include LKI ID	N/A	LKI (Line Kilometre Index) ID
locality	N/A	MoT Admin Area
locality	N/A	Regional District
N/A	N/A	Building Type ID
N/A	N/A	Location ID
siteStatus	N/A	Site is retired if Site Retired Date not null; active otherwise
accessPointStatus	N/A	Civic Address is retired if Civic Address Retired Date not null; active otherwise
siteRetireDate	N/A	Civic Address.Retired Date
fullSiteDescriptor	N/A	N/A
accessPointRetireDate	N/A	Site.Retired Date
N/A	N/A	N/A

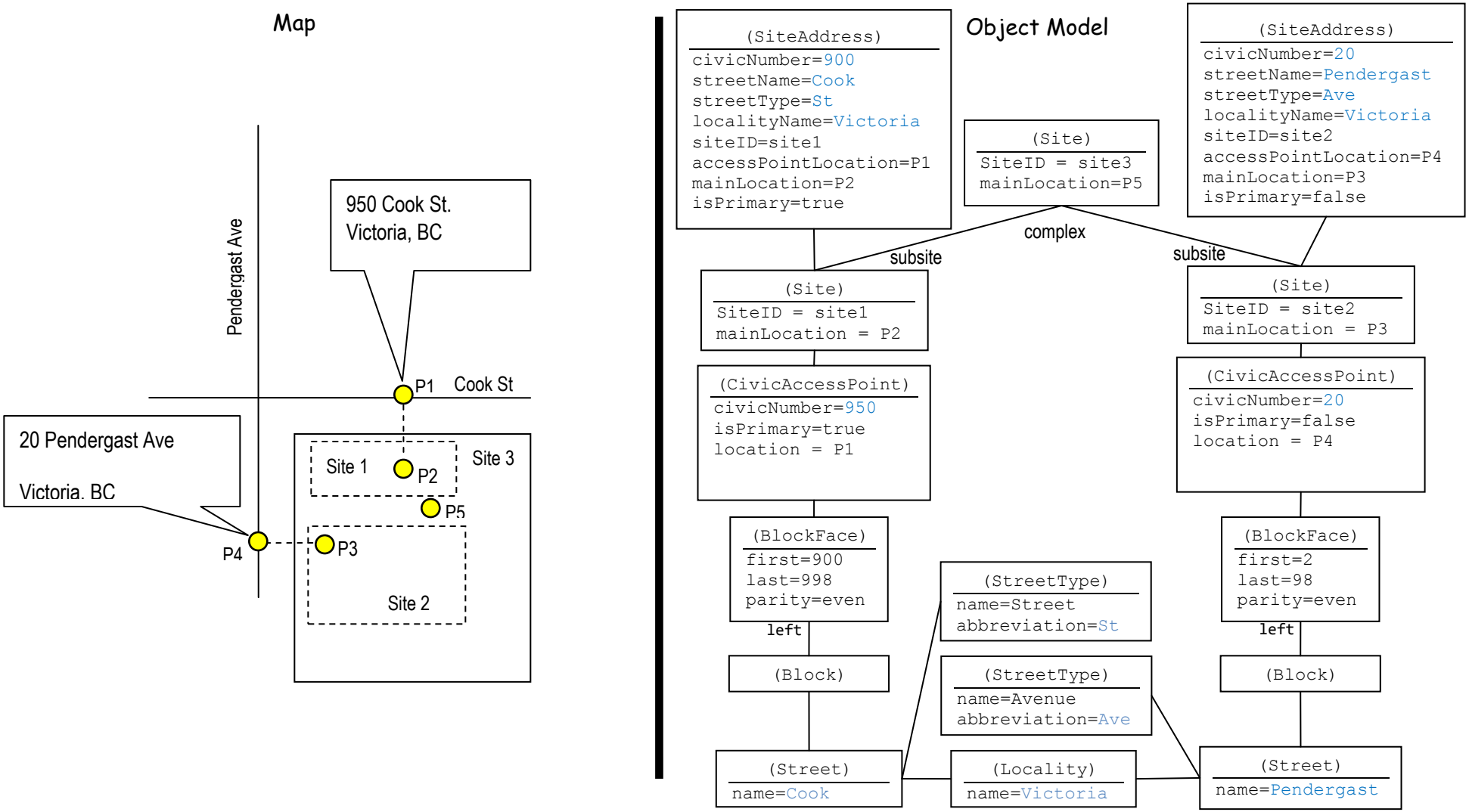
2.7 Address Examples

These object diagrams illustrate how normal and unusual addressing situations are supported. Each diagram shows an object model and its depiction on a map. Objects and attributes not relevant to the example are not shown.

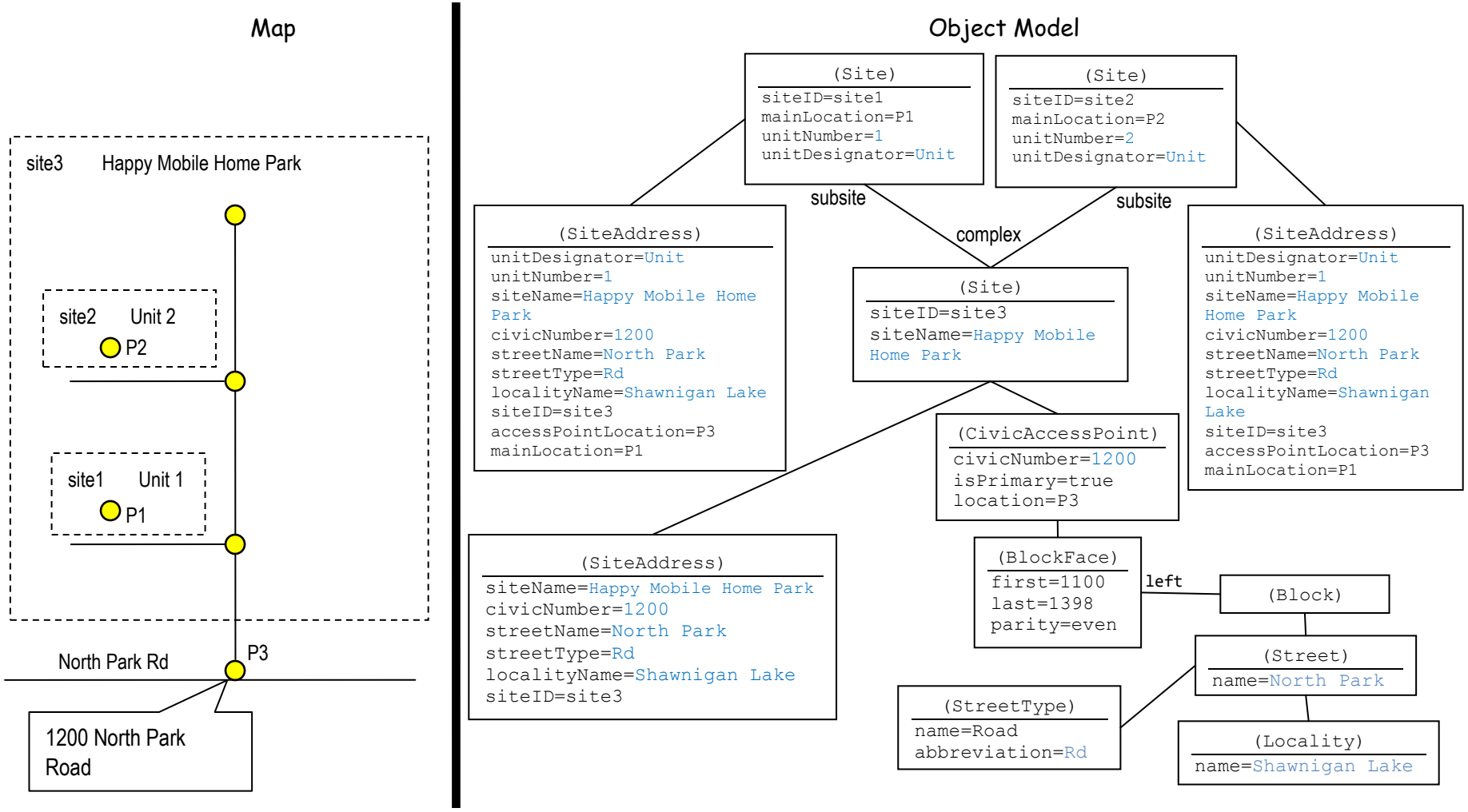
2.7.1 A Single Site with A Single Address



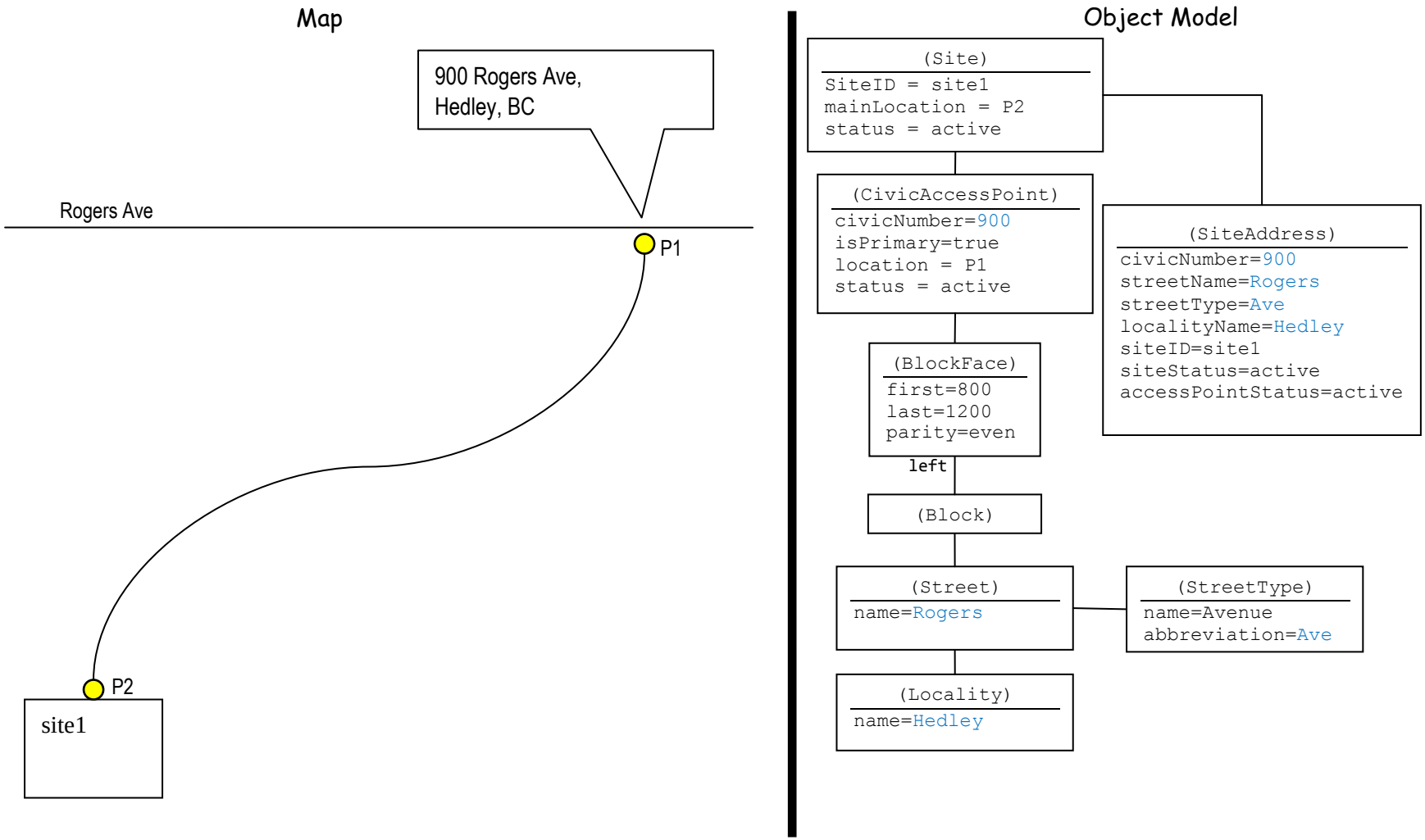
2.7.2 A Single Building with Addresses on Two Streets



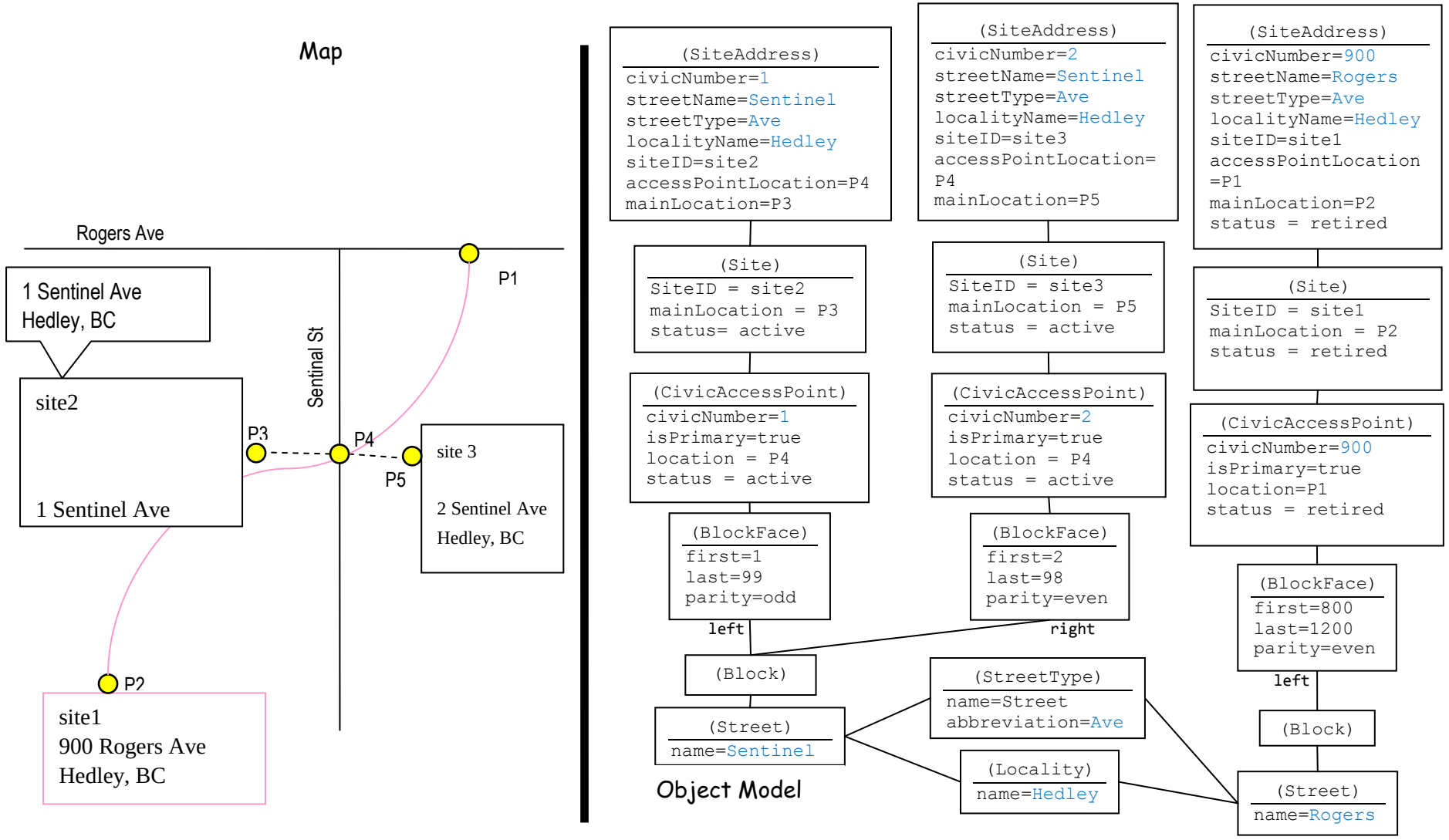
2.7.3 A Mobile Home Park



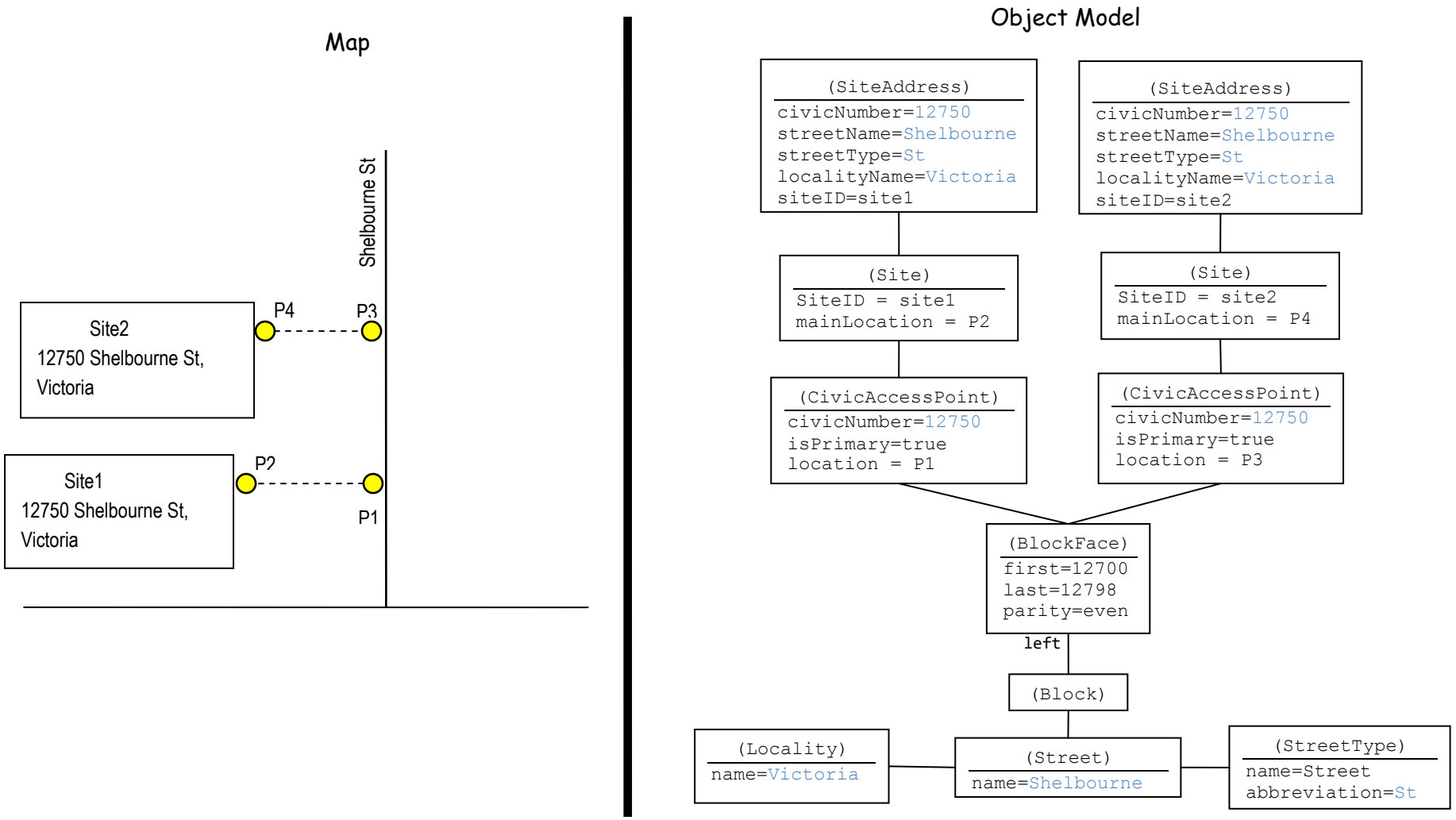
2.7.4 A Five Acre Lot with a House Set Back From The Road



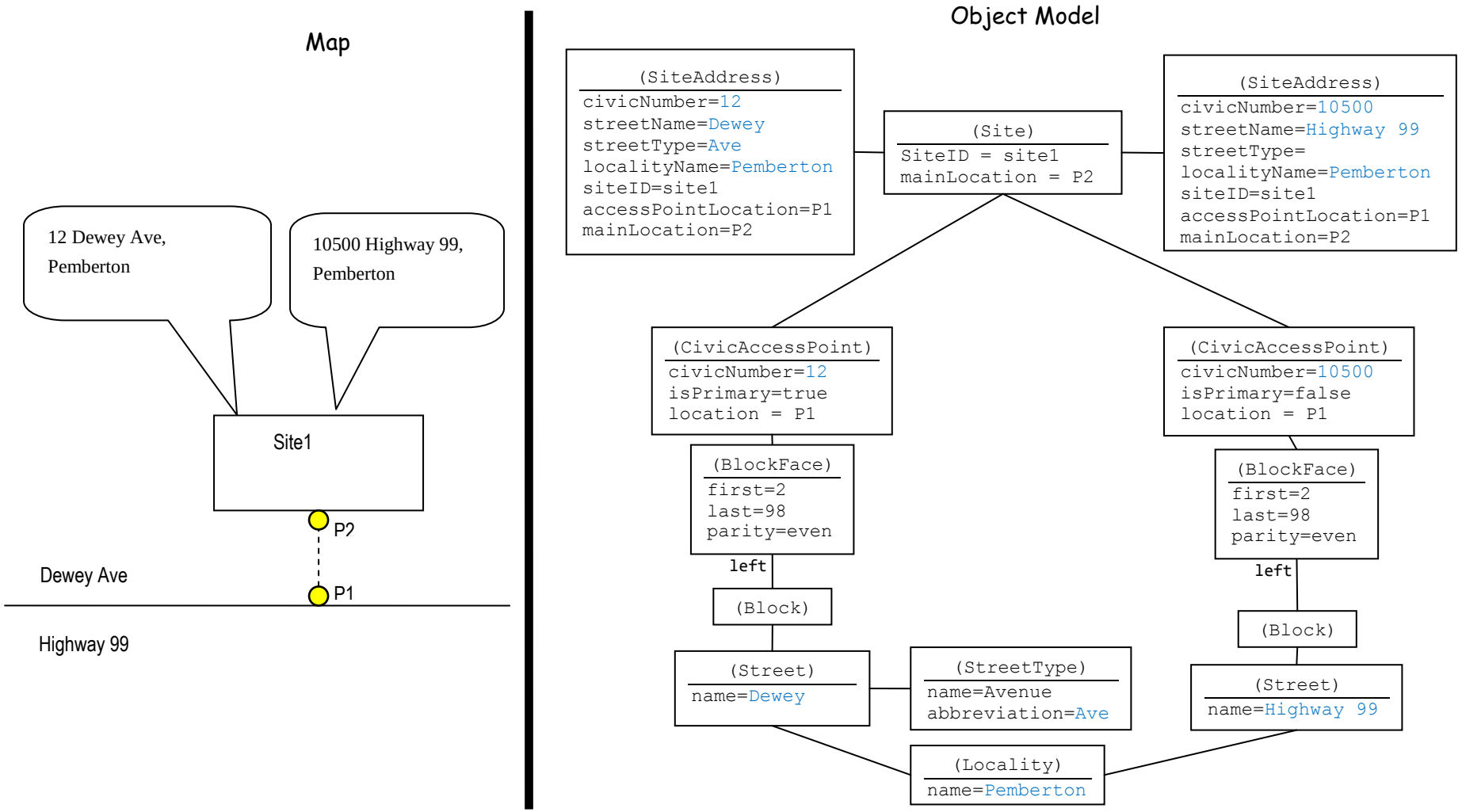
2.7.5 A House On a Five Acre Lot is Demolished and Replaced By New Houses



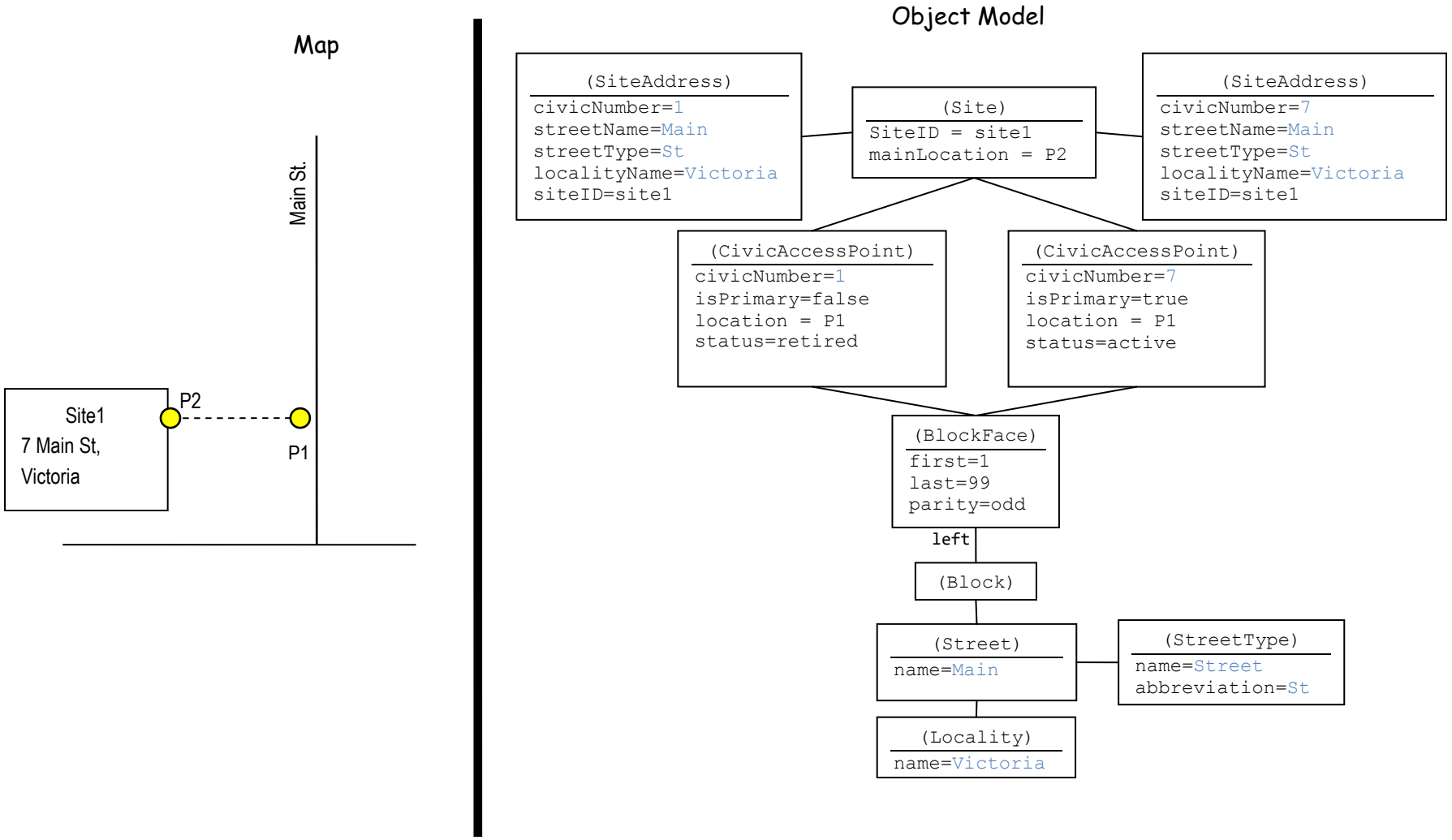
2.7.6 Two Buildings Assigned The Same Civic Number



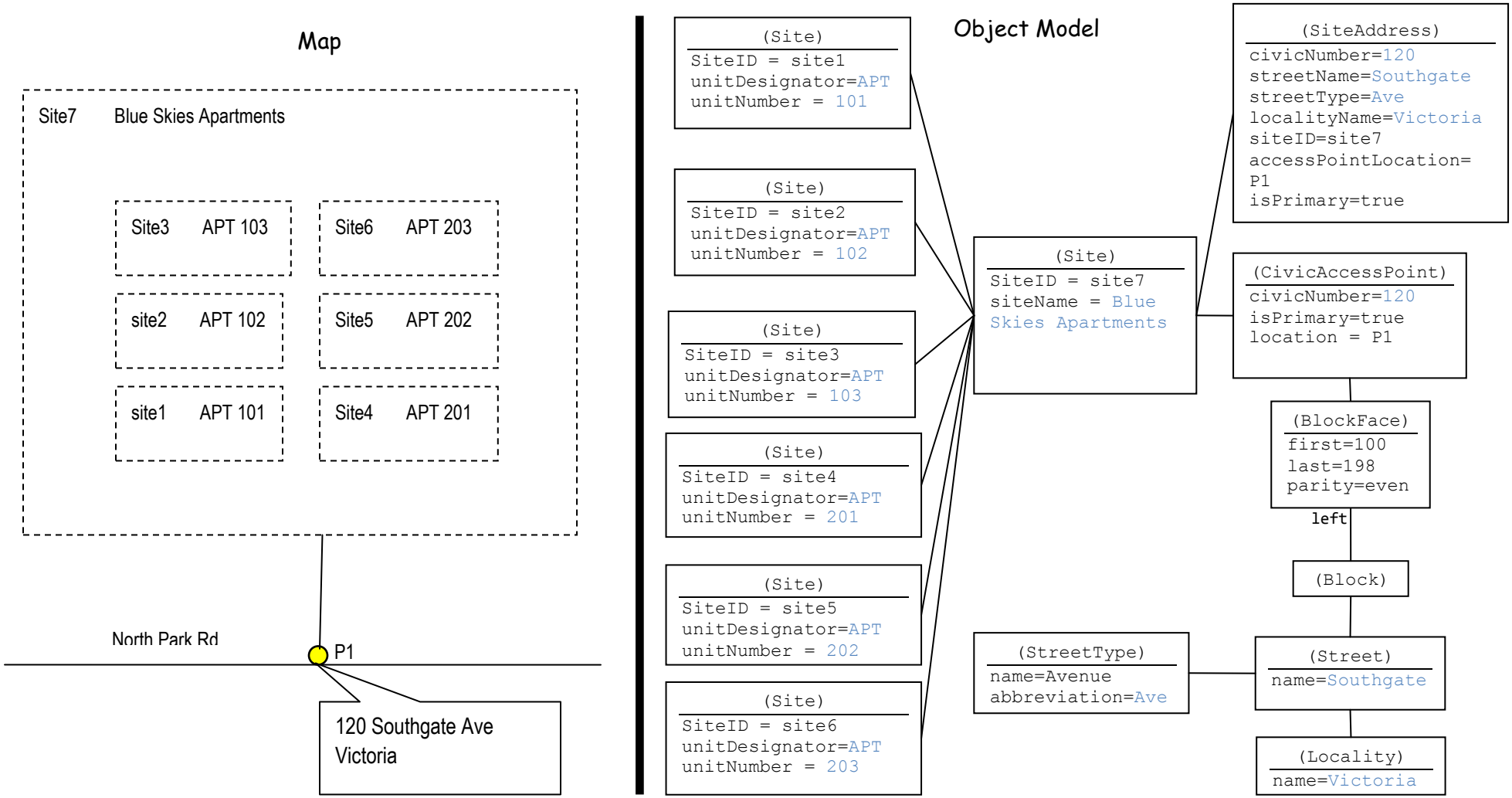
2.7.7 One House with Two Addresses



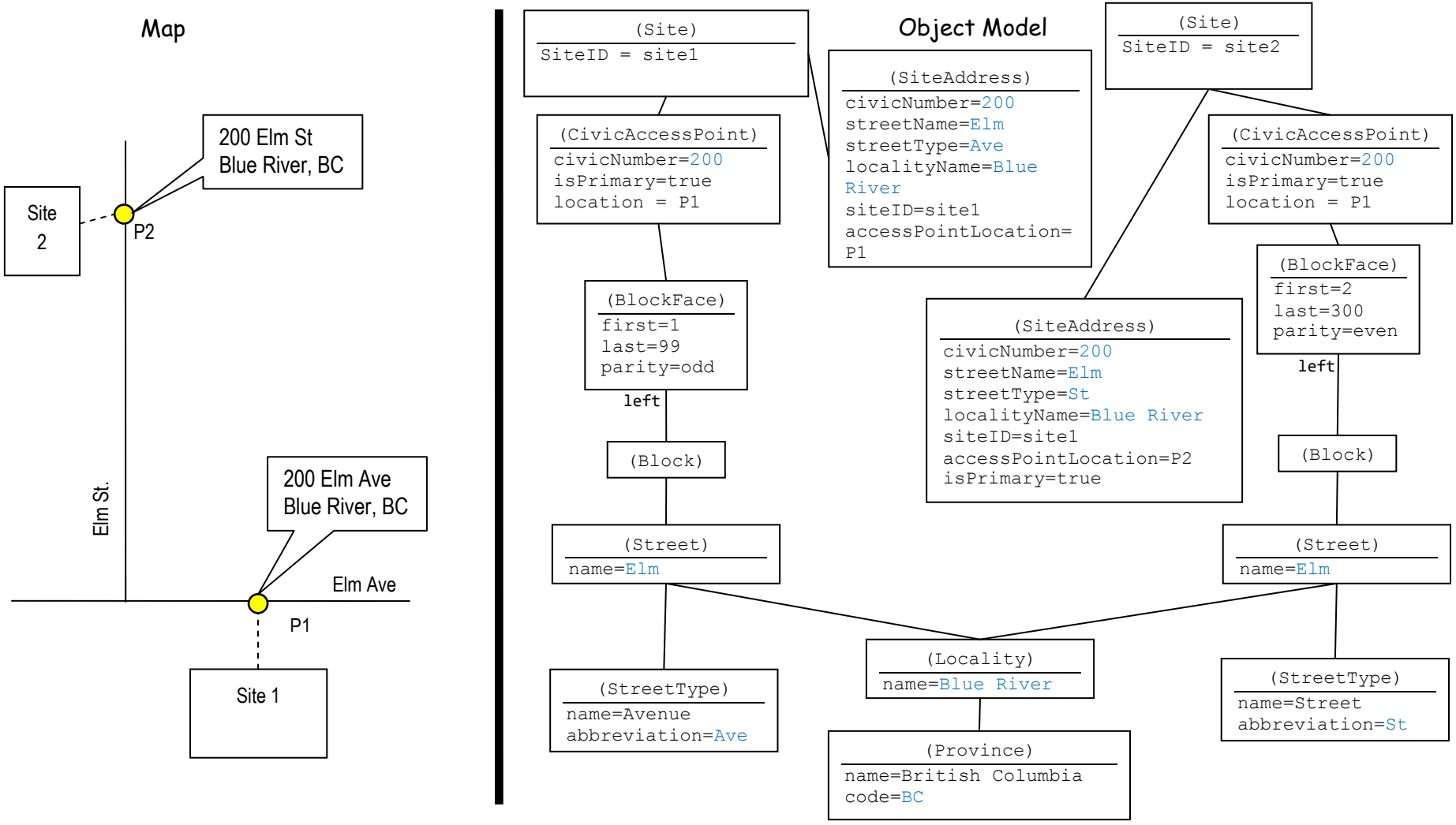
2.7.8 A House Gets a new Civic Number



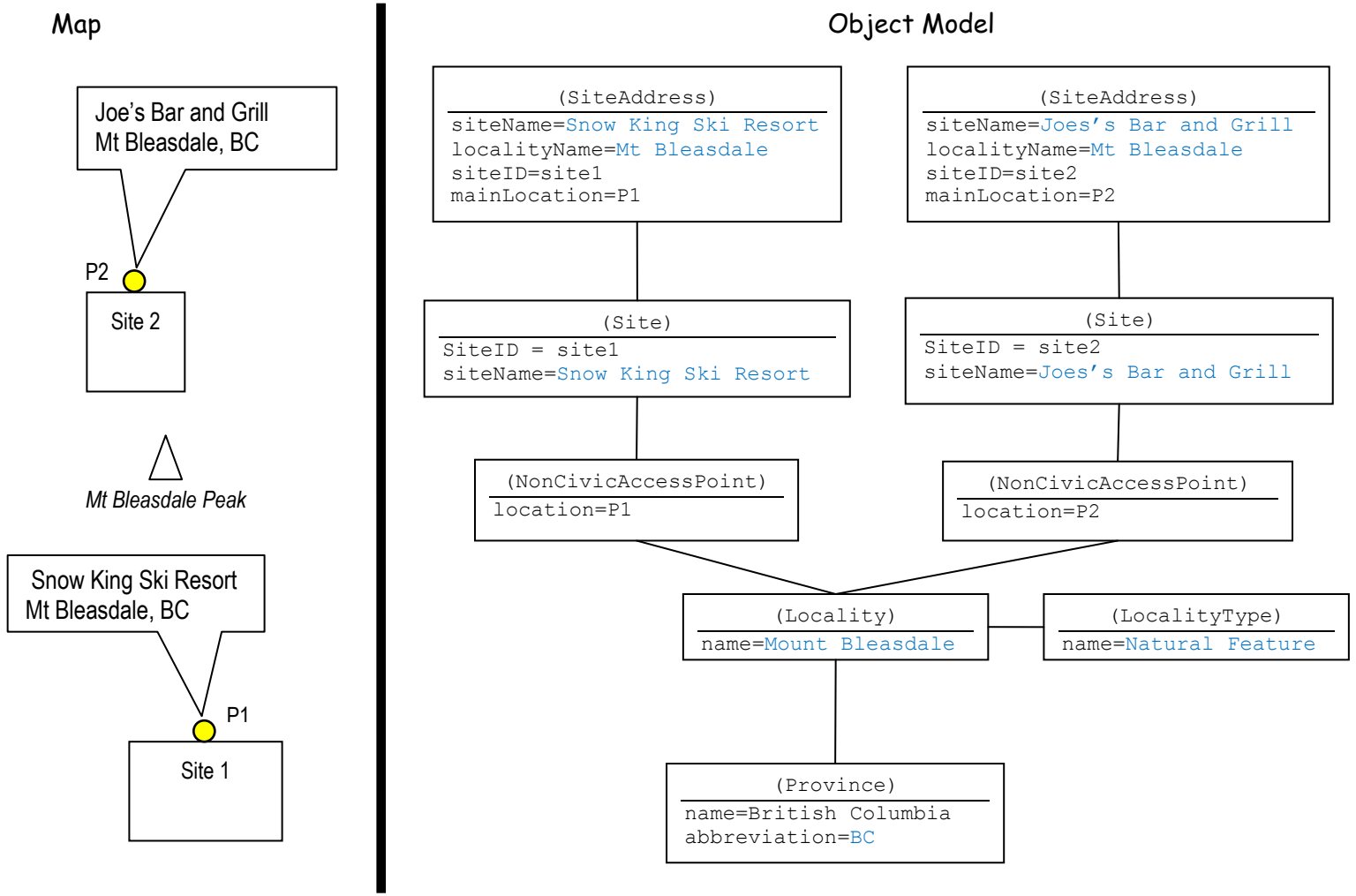
2.7.9 A Multiple Unit Dwelling



2.7.10 Two Addresses with The Same Street Name But Different Street Types

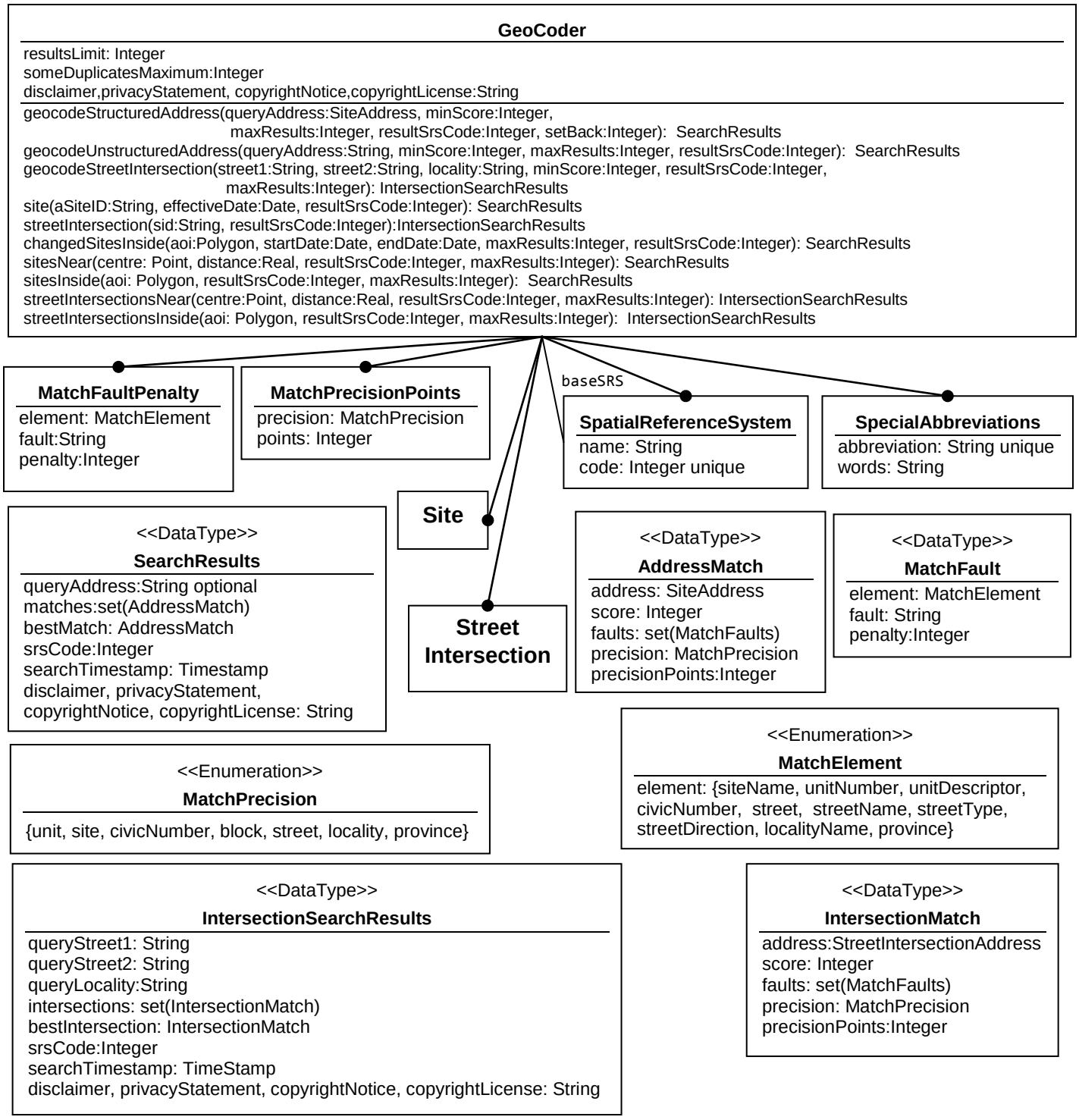


2.7.11 Two Buildings On The Same Natural Feature (Non-civic Addresses)



3. SERVICE MODEL (GEOCODER)

3.1 Class Diagram



```
Context: Geocoder::streetIntersection(sid:String, resultSrsCode:Integer):  SearchResults
```

3.2 Class Definition

3.2.1 Overview

The Geocoder class performs geocoding and reverse geocoding. Geocoding returns the geographic coordinates of a civic address, street intersection address, site identifier, street intersection identifier, site name within a natural feature, or site name within a locality. Reverse geocoding returns all addresses within a geographic area or near a geographic location.

The Geocoder is forgiving. It tries to correct spelling errors in input, overlook missing address elements, replace street and locality aliases, and return all matches regardless of match quality.

The SpatialReferenceSystem class defines the spatial reference systems that the Geocoder supports. A geocoder must support SpatialReferenceSystems with the following codes: lat/lon(4326), bc albers(3005), and UTM NAD83 Zones 7 - 11 (26907-26911). Additional spatial referencing systems may be defined. baseSRS is the SpatialReferenceSystem that all geometry is kept in.

SearchResults and AddressMatch are data types used to hold the results of a site address geocoding or reverse geocoding request. IntersectionSearchResults and IntersectionMatch are used to hold the results of a street intersection address geocoding request.

An AddressMatch conveys site address match quality through *score*, *precision*, and *faults* properties. MatchFault is a data type that defines the nature of the fault, the address property affected, and the fault penalty. MatchPrecisionPoints defines the points assigned to each match precision level. MatchPrecision is an enumeration of all precision levels. MatchFaultPenalty defines the penalty value of each possible fault. Penalties are subtracted from the appropriate precision level points to arrive at a match score.

An IntersectionMatch conveys street intersection address match quality in the same way AddressMatch does.

3.2.2 Address Standardization

Geocoding operations standardize their input addresses as follows:

- Abbreviated site names, street names, and locality names are expanded. For example, "W Broadway" is expanded to "West Broadway" and "N Vancouver" is expanded to "North Vancouver"
- Misspelled street names are corrected.

- Unabbreviated street types are abbreviated.
- Unabbreviated street directions are abbreviated.
- Missing street types and directions are added.
- Site aliases are replaced by their official sitenames (e.g., "Malaspina College" is replaced by "Vancouver Island University")
- Street aliases are replaced by their official street names
- Locality aliases are replaced by their prime localities (e.g., "Fairfield" is replaced by "Victoria").
- Unabbreviated province names are abbreviated.
- An alternate or retired address is augmented by its associated primary address. For example, if 5 Elm St. was retired and replaced by 7 Elm St, both 5 and 7 Elm St are added to the set of match results. For another example, if an input address is 5 Elm St, and the primary address for that site is 41 Maple Ave, both 5 Elm St and 41 Maple Ave are added to the set of match results.

Address as a single string is standardized and returned as the addressString attribute of a SiteAddress. See section 2.1.3 for further details.

3.2.3 Match Quality

An AddressMatch conveys match quality through precision level, faults, and overall score. MatchPrecision is an enumeration of all precision levels. MatchFault is a data type that defines the nature of a fault and the address element affected.

MatchPrecisionPoints defines the number of points for each match precision level. The maximum number of points allowed is 100. Recommended initial values are as follows:

Precision Level	Points
site	100
unit	100
civicNumber	100
block	95
street	40
locality	20
province	0

Here is the definition of each precision level:

- **province** – no match was found; lowest precision
- **locality** – the locality was matched but not the street, civicNumber, or siteName
- **street** - the street and locality matched but the civicNumber didn't match or fall within a block range
- **block** – the street and locality matched and the civicNumber falls within a block range
- **civicNumber** – the civicNumber, street, and locality matched; highest precision
- **unit** - the unitNumber, unitDesignator(if input), civicNumber, street, and locality matched; highest precision
- **site** – the site name and locality matched; if civicNumber and street were input, they matched too; highest precision

MatchFaultPenalty defines the penalty value of each possible fault. Determining penalty values is best done through trial and error.

MatchFault is a data type with the following attributes:

element is the name of the address element involved in the fault. Element values are defined in the MatchElement enumeration in section 3.1. Examples include **civicNumber, streetName,** and **locality**.

fault represents the nature of the fault. Examples include **isSiteAlias, isStreetAlias, isLocalityAlias, someDuplicates, manyDuplicates, spellCorrected, notMatched, missing, notinBlock** (only applies to **civicNumber** element), **misplaced**, and **notinLocality**(only applies to **street** element). So, for a given address match, the **civicNumber** element may be **notInBlock** (not found on any block of a given street), the **streetDirection** element may be **missing** and the locality element may be **spellCorrected** because it was spelled **Ross** instead of **Woss**.

penalty is the amount (between 1 and 100) that was subtracted from the appropriate MatchPrecisionPoints.

Match score is determined by taking the appropriate precision level points and subtracting any fault penalties. Match score must be between 0 and 100.

3.2.4 Positional Accuracy

Positional accuracy is determined differently for civic and non-civic addresses.

3.2.4.1 Civic Addresses

To achieve the highest positional accuracy for civic addresses, the geocoder uses adaptive address interpolation which has four cases as follows:

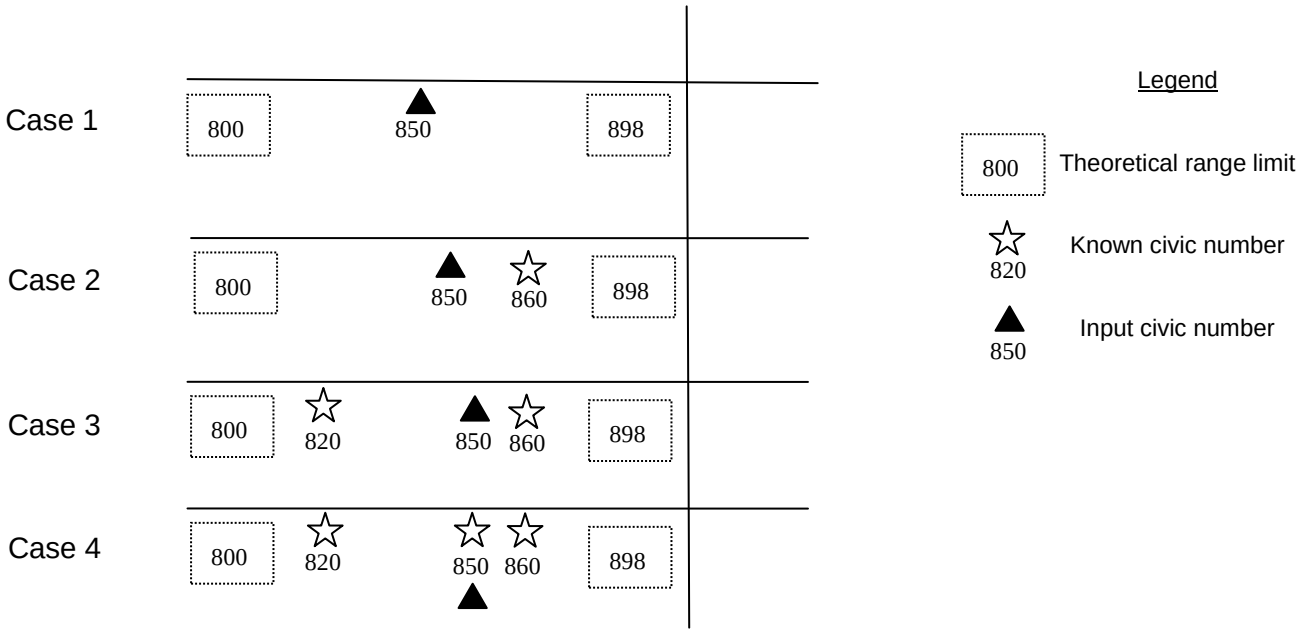
1. The input address matches to a block range defined by two theoretical civic number limits. A linear interpolation is performed based on the block range and the input civic number.

2. The input address matches to a block sub-range defined by one known site access point and a theoretical civic number limit. A linear interpolation is performed based on the sub-range and the input civic number.

3. The input address matches to a block sub-range defined by two known site access points. A linear interpolation is performed based on the sub-range and the input civic number.

4. The input address matches to the civic number of an associated site.

   Interpolation accuracy of all cases is illustrated in the following diagram:



Since there is no interpolation involved in case 4, the geocoder would return the value of CivicAccessPoint.positionalAccuracy. For cases 1-3, the geocoder will return an accessPointPositionalAccuracy of **medium.**

3.2.4.2 Non-Civic Addresses

To achieve the highest positional accuracy for non-civic addresses, the geocoder will utilize the site name, main location, and locality. Here are three cases:

1. The input address matches to a site name within a locality and the site's *mainLocation* is defined. The returned *accessPointLocation* is set to the value of *mainLocation* and *accessPointPositionalAccuracy* is set to *mainLocationPositionalAccuracy.*

2. The input address matches to a *siteName* within a *locality* but the site's *mainLocation* is not defined. The returned *accessPointLocation* is set to the value of *locality.asPoint* and *accessPointPositionalAccuracy* is set to **low.**

3. The input address doesn't match a *siteName* but does match a *locality*. The returned *accessPointLocation* is set to the value of *locality.asPoint* and *accessPointPositionalAccuracy* is set to **low.**

SearchResult and AddressMatch are used to hold the results of an address geocoding or reverse geocoding request.

IntersectionSearchResults and IntersectionMatch are used to hold the results of a street intersection geocoding request.

3.2.5 Using the Geocoder as an Address Hub in Client Applications

The geocoder class was designed to allow applications to update their own addresses and on their own schedules. Here's an update scenario:

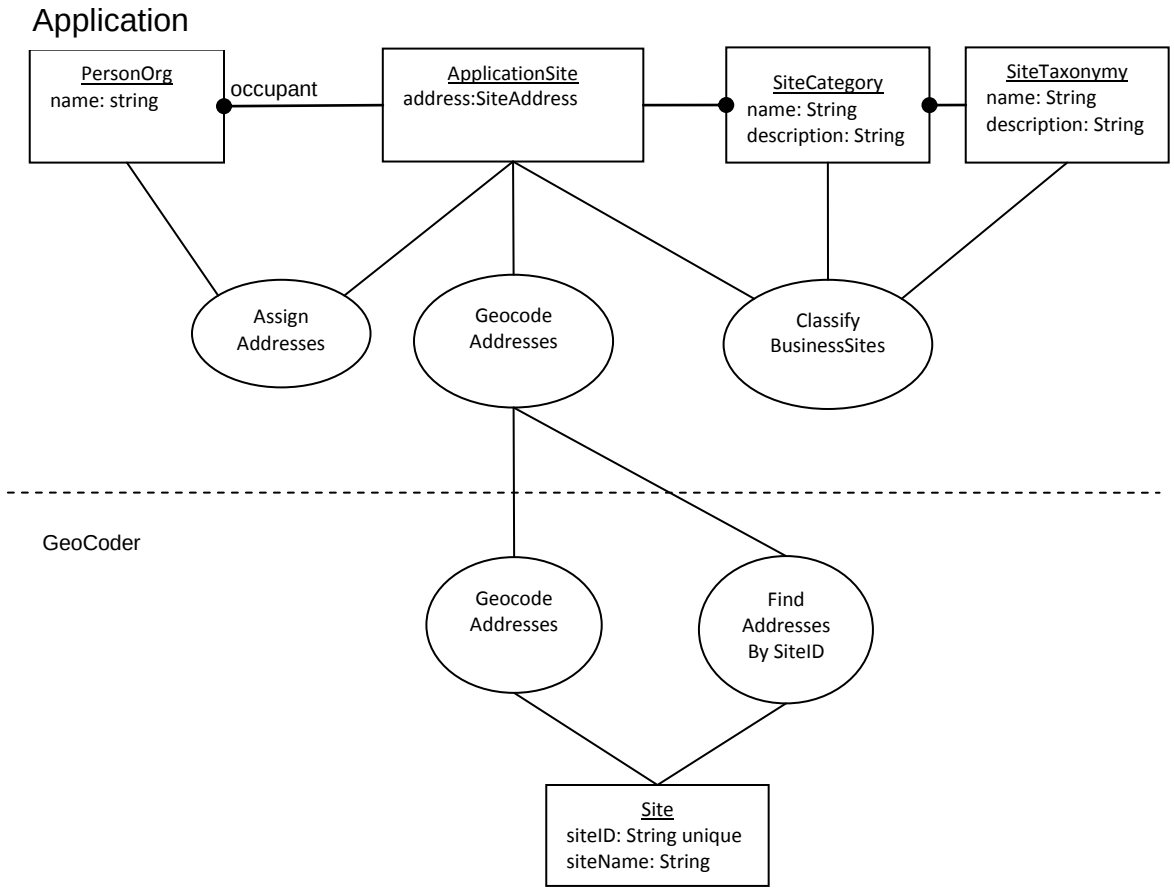
1. A client application uses the geocoder to geocode a set of addresses and stores the results. Addresses that matched down to the site level will include a unique, immutable siteID.

2. According to a business area's own update schedule,

    a. A client application makes a request to the geocoder for all sites that have changed since a given date and applies the changes as appropriate.

    b. Addresses in the client application that weren't initially matched down to the matchPrecision level of site are batch geocoded and changes are stored as appropriate. See section 3.2.3 for a discussion of match precision levels.

The following diagram illustrates the scenario described above:



The application avoids having to manage Site, AccessPoint, BlockFace, Block, Street, Locality, and Province objects by using a single ApplicationSite.address property of type SiteAddress instead. Use of such a denormalized value doesn't risk update anomaly because ApplicationSite.address is never updated; it is only replaced in its entirety by a new value returned by the geocoder. As is common with applications that manage addresses, the application in the diagram also tracks site occupants and classifies sites according to multiple, application-specific taxonomies.

An application can manage street intersection addresses like civic and non-civic addresses with the intersectionID playing the same role as siteID.

The need for information sharing agreements should be considered by each project implementing a multi-agency, geocoding service.

3.3 Attributes

A geocoder has the following attributes:

resultLimits defines the absolute maximum number of search results that can be returned in a single request. A given request can specify a lower limit.

someDuplicatesMaximum defines the largest number of duplicate matches that can be considered a matchFault of **someDuplicates** instead of **manyDuplicates**.

privacyStatement defines the privacy policy governing the information accessible to the geocoder. For an example, see http://www.gov.bc.ca/com/privacy.html

disclaimer defines the warranty disclaimer. For an example, see http://www.gov.bc.ca/com/disclaimer.html

copyrightNotice is the text of the notice. For example:

Copyright (c) 2010, Province of British Columbia

copyrightLicense defines the license under which the geocoder service is copyrighted. For an example, see http://www.gov.bc.ca/com/copyright.html

privacyStatement, *disclaimer*, *copyrightNotice*, and *copyrightLicense* are included with all geocoder search results.

A SearchResults is a data type with the following attributes:

queryAddress is the SiteAddress as provided to the geocoder

matches is the set of all AddressMatches

bestMatch is the best-matching address in *matches*

srsCode is the code of the spatial reference system of the points in *matches*

searchTimeStamp is the completion time of the operation that created the search results.

privacyStatement (see above)

disclaimer (see above)

copyrightNotice (see above)

copyrightLicense (see above)

An AddressMatch is a data type with the following attributes:

address is a matching civic or non-civic SiteAddress. If the match precision level is **site**, **civicNumber**, or **unit**, siteID is set appropriately; otherwise it is set to the empty string. In the case of a score of zero (no match), *address.accessPointLocation* and *address.mainLocation* are set to *province.asPoint*.

score represents the quality of the match; 0 means no match, 100 means a perfect match (see section 3.2.3 for details.

faults represents the problems that were encountered in matching an input address to a known address. See section 3.2.3 for some examples.

precision is the level of precision of the match (see section 3.2.3 for details)

precisionPoints is the number of points assigned to the *precision*.

An IntersectionSearchResults is a data type with the following attributes:

queryStreet1 and *queryStreet2* are the two streets input to the geocodeStreetIntersection operation.

queryLocality is the locality name input to the geocodeStreetIntersection operation.

intersections is the set of IntersectionMatches found.

bestIntersection is the IntersectionMatch with the highest score.

srsCode is the code of the Spatial Referencing System that the results are returned in.

searchTimeStamp is the completion time of the operation that created the search results.

privacyStatement (see above)

disclaimer (see above)

copyrightNotice (see above)

copyrightLicense (see above)

An IntersectionMatch is a data type with the following attributes:

address is the StreetIntersectionAddress of the matching StreetIntersection.

score represents the quality of the match; 0 means no match, 100 means a perfect match.

faults represents the problems that were encountered in matching an input address to a known address. See section 3.2.3 for some examples.

precision is the level of precision of the match.

precisionPoints is the number of points assigned to the *precision*.

3.4 Operations

3.4.1 Geocode Structured Address

```
context:Geocoder::geocodeStructuredAddress(queryAddress:SiteAddress, minScore:Integer,
                maxResults:Integer, resultSrsCode:Integer, setBack:Real): SearchResults
```

Returns all sites whose site addresses match a queryAddress.

setback (in metres) must be zero or greater

```
pre:   setBack >=0
```

minScore defines the minimum score that a match must achieve for inclusion in search results.

```
pre:   0 <= minScore and minScore <= 100
```

maxResults defines the maximum number of search results returned. It cannot be larger than resultsLimit.

```
pre:   0 < maxResults and maxResults <= resultsLimit
```

resultSrsCode defines the code of the spatial reference system that accessPoints and mainLocations are to be returned in. resultSrsCode must be a supported spatial reference system

```
pre:   spatialReferenceSystem->exists(s| s.code=resultSrsCode)
```

For best matching, any of the following sets of SiteAddress elements should be provided:

unitNumber, unitNumberSuffix, civicNumber, civicNumberSuffix, streetName, streetType, streetDirection, locality

civicNumber, civicNumberSuffix, streetName, streetType, streetDirection, locality

siteName, locality (if non-civic address)

Matching is possible with the following element combinations but the match quality is lower:

street, locality
street
locality

unitDesignator and province are optional

streetType, streetDirection, and locality are optional but if missing, may result in a lower match score.

post:

Given a civic address (e.g., 1204 Esquimalt Rd, Esquimalt, BC) and a matched address, the following table shows how access point and related properties are determined at each match precision level:

match Precision	siteID	accessPointLocation	accessPoint positional Accuracy	mainLocation	mainLocation positional accuracy
province	None	Province.asPoint	Low	Province.asPoint	low
locality	None	Locality.asPoint	Low	Locality.asPoint	low
street	None	street.asPoint	Low	Street.asPoint	low
block	None	matchedBlock.interpolateAlongCentre Line(civicNumber)	Medium	matchedBlock.interpolateMain Location(civicNumber, aSetBack)	medium
civicNumber	matchedSite .siteID	matchedAccessPoint.location	High	matchedSite.mainLocation()	matchedSite.main LocationPositional Accuracy()
unitNumber	matchedSite .unitNumber	matchedSite.unitNumber	High	matchedSite.mainLocation()	matchedSite.main LocationPositional Accuracy()

Given a non-civic address (e.g., Remote Logging Camp, Faraway Lake, BC) and a matched address, the following table shows how access point and related properties are determined at each match precision level:

match Precision	siteID	accessPointLocation	accessPoint Positional Accuracy	mainLocation	mainLocation positional accuracy
province	none	matchedProvince.asPoint	Low	Province.asPoint	low
locality	none	matchedLocality.asPoint	Low	Locality.asPoint	low
site	matchedSite.siteID	matchedAccessPoint.location	matchedAccessPoint. positionalAccuracy	matchedSite.mainLocation()	matchedSite.mainLocation PositionalAccuracy()
unitNumber	matchedSite.siteID	matchedAccessPoint.location	matchedAccessPoint. positionalAccuracy()	matchedSite.mainLocation()	matchedSite.mainLocation PositionalAccuracy()

If no match is found in a given locality, alias sites, streets, and localities are searched.

All string comparisons ignore case and are made using the smartMatch operation; smarter alternatives to smartMatch are allowed.

All expanded forms of queryAddress.siteName, queryAddress.streetName, and queryAddress.locality are smartMatched. Expansion is performed by the expandName operation (see section 3.4.11)

Score values are set as per section 3.2.3.

Valid result

```
post:  not result->isEmpty()

post:  result.matches->forAll(m | m.address.siteID.notEmpty() implies Site->exists( s |
m.address.siteID= s.siteID and (m.address= s.civicAddress(resultSrsCode) or
m.address=nonCivicAddress(resultSrsCode) ) ) )

post:  result.matches->forAll(m | m.score <= bestMatch.score)

post:  result.matches->forAll(m | o<=m.score and m.score <= 100)

post:  result.matches->exists(m| m = bestMatch)
```

3.4.2 Geocode Unstructured Address

```
Context: Geocoder::geocodeUnstructuredAddress(queryAddress:String, minScore:Integer,
             maxResults:Integer, resultSrsCode:Integer, setback: Real):  SearchResults
```

Returns all addresses that match or partially match an unstructured (free form) address string.

minScore defines the minimum score that a match must achieve for inclusion in search results.

```
pre:   0 <= minScore and minScore <= 100
```

maxResults defines the maximum number of search results returned. It cannot be larger than resultsLimit.

```
pre:   0 < maxResults and maxResults <= resultsLimit
```

resultSrsCode defines the code of the spatial reference system that accessPoints and mainLocations are to be returned in. resultSrsCode must be a supported spatial reference system

```
pre:   spatialReferenceSystem->exists(s| s.code=resultSrsCode)
```

setback (in metres) must be zero or greater

```
pre:   setBack >=0
```

pre: queryAddress is an address as a single string and can take one the following formats (square brackets means zero or one occurrences; an asterisk following square brackets means zero or more):

```
Format 1. [[unitDesignator unitNumber[unitNumberSuffix]] [siteName],]*
            civicNumber[civicNumberSuffix] streetName streetType [streetDirection]
             [,localityName] [,provinceCode]

Format 2. unitNumber[unitNumberSuffix]-
            civicNumber[civicNumberSuffix] streetName streetType [streetDirection]
             [,localityName] [,provinceCode]

Format 3. civicNumber[civicNumberSuffix] streetName streetType [streetDirection]
            [[unitDesignator unitNumber[unitNumberSuffix]]
             [,localityName] [,provinceCode]

Format 4. [[unitDesignator unitNumber[unitNumberSuffix]] [siteName],]*
            [,localityName] [,provinceCode]

Format 5. streetName streetType [streetDirection], [,localityName] [,provinceCode]

Format 6. localityName [,provinceCode]
```

Here are some examples:

1. A civic address with no unit number:

   ```
   1025 HAPPY VALLEY RD, METCHOSIN, BC
   420 GORGE RD E, VICTORIA, BC
   ```

2. A civic address with a unit number:

   ```
   PAD 2, 1200 NORTH PARK RD, SHAWNIGAN LAKE, BC
   1200 NORTH PARK RD PAD 2, SHAWNIGAN LAKE, BC
   2-1200 NORTH PARK RD, SHAWNIGAN LAKE, BC
   ```

3. A civic address with a simple site name:

   ```
   PORT ALICE HEALTH CENTRE, 1090 MARINE DRIVE, PORT ALICE, BC
   ROYAL ATHLETIC PARK, 1014 CALEDONIA AVE, VICTORIA, BC
   ```

4. A civic address with a unit within a named complex:

   ```
   PAD 2, HAPPY MOBILE HOME PARK, 1200 NORTH PARK RD, SHAWNIGAN LAKE, BC
   ROOM 103A, CLEARIHUE BUILDING, UNIVERSITY OF VICTORIA, 3800 FINNERTY RD,
   VICTORIA, BC
   ROOM 230, WEST BLOCK, ROYAL JUBILEE HOSPITAL, 1952 BAY ST, VICTORIA, BC
   ```

5. A civic address with a unit within a named complex and the unit has both a number and a name:

   ```
   CABIN C HERON, HAPPY FISHING RESORT, WHOPPER, BC
   ```

6. A civic address with a unit within a unit of a named complex:

```
PAD 11, TERMINAL 3, BC SPACEPORT, 1 MILKY WAY, STAR CITY, BC
```

7. A street within a locality:

```
WILLOW DRIVE, 70 MILE HOUSE, BC
HORSE LAKE ROAD, 100 MILE HOUSE, BC
```

8. A locality within the province:

```
PEACE RIVER REGIONAL DISTRICT, BC
100 MILE HOUSE, BC
V8T, BC
```

```
post: see section 3.4.1 Geocode Structured Address
```

3.4.3 Geocode Street Intersection

```
context Geocoder::geocodeStreetIntersection(street1:String, street2:String,
                              locality:String, minScore:Integer resultSrsCode:Integer,
                              maxResults:Integer):  IntersectionSearchResults
```

Returns the geographic location of all intersections of the two given streets within the given locality

street1 and street2 may take one of the following forms:

```
streetName
streetName streetType [streetDirection]
```

minScore defines the minimum score that a match must achieve for inclusion in search results.

```
pre:   0 <= minScore and minScore <= 100
```

maxResults defines the maximum number of search results returned. It cannot be larger than resultsLimit.

```
pre:   0 < maxResults and maxResults <= resultsLimit
```

resultSrsCode defines the code of the spatial reference system that intersection locations are to be returned in. resultSrsCode must be the code of a supported spatial reference system

```
pre:   spatialReferenceSystem->exists(s| s.code=resultSrsCode)
```

```
post:
```

An IntersectionMatch is created for each StreetIntersection that has associated streets named *street1* and *street2* within the locality named *locality*.

Multiple IntersectionSearchResults may be returned in the following cases:

- Two streets named street1 and street2 are found and their road centrelines cross at several intersections.
- For a given locality, no streetType was specified for a streetName that has more than one (e.g., Happy St, Happy Rd),
- No streetDirection was specified for a streetName/Type that has more than one (e.g. Gorge Rd E, Gorge Rd W),
- The locality was unspecified.

If streetType, streetDirection, or locality are not specified in street1 or street2, matches may still be possible but will have a lower score.

If no match is found in a given locality, street and locality aliases are searched.

All expanded forms of the name part of street1 and street2 are smartMatched. Expansion is performed by the expandName operation (see section 3.4.12).

All string comparisons are made using the smartMatch operation.

Score values are set as per section 3.2.3

Valid result

```
post:  not result->isEmpty()
post:  result.intersections->forAll(m | m.score <= bestIntersection.score)
post:  result.intersections->forAll(m | o<=m.score and m.score <= 100)
post:  result.intersections->exists(m| m = bestIntersection)
post:  result.matches->forAll(m | m.address.srsCode = resultSrsCode)
```

3.4.4 Reverse Geocode Sites Inside An Area Of Interest

```
context Geocoder::sitesInside(aoi: Polygon, resultSrsCode:Integer,
                              maxResults:Integer):  SearchResult
```

Returns all sites and their addresses within a given area of interest. Only the primary address of each site is returned.

```
---    A valid area of interest can't be null and must be a valid polygon
pre: not aoi.isNull()
pre: not aoi.isNull() implies aoi.isValid
```

maxResults defines the maximum number of search results returned. It cannot be larger than resultsLimit.

```
pre:   0 < maxResults and maxResults <= resultsLimit
```

resultSrsCode defines the code of the spatial reference system that accessPoints and mainLocations are to be returned in. resultSrsCode must be a supported spatial reference system

```
pre:   spatialReferenceSystem->exists(s| s.code=resultSrsCode)


---    aoi is assumed to be defined in the spatial reference system
---    defined by resultSrsCode
```

Valid result

```
post:  not result->isEmpty()implies
       result.matches->forAll(m | m.address.siteID.notEmpty())


post:  result.matches->forAll(m | Site->exists( s | m.address.siteID = s.siteID and
                        (m.address= s.civicAddress(resultSrsCode) or
                         m.address=nonCivicAddress(resultSrsCode))))

post: result.matches->forAll(m |
                     m.address.accessPointLocation.inside(aoi)

post:  not result->isEmpty()implies
       result.matches->forAll(m | m.score=100)

post:  not result->isEmpty()implies result.matches->exists(m| m = bestMatch)
```

3.4.5 Reverse Geocode Street Intersections Inside An Area Of Interest

```
context Geocoder::streetIntersectionsInside(aoi: Polygon, resultSrsCode:Integer,
                                        maxResults:Integer):  IntersectionSearchResults
```

Returns all street intersections that intersect a given area of interest.

maxResults defines the maximum number of search results returned. It cannot be larger than resultsLimit.

```
pre:   0 < maxResults and maxResults <= resultsLimit
```

resultSrsCode defines the code of the spatial reference system that intersection locations are to be returned in. resultSrsCode must be the code of a supported spatial reference system

```
pre:   spatialReferenceSystem->exists(s| s.code=resultSrsCode)


---    A valid area of interest can't be null and must be a valid polygon
pre: not aoi.isNull()
pre: not aoi.isNull() implies aoi.isValid

---    aoi is assumed to be defined in the spatial reference system defined by
resultSrsCode
```

Valid result

```
post:  not result->isEmpty() implies
             result.intersections->forAll(m | m.score = 100 and m.precisionLevel=block)

post:  not result->isEmpty() implies result.intersections->exists(m| m = bestIntersection)

post:  result.matches->forAll(m | m.address.srs = resultSrsCode)
```

3.4.6 Reverse Geocode Street Intersections Near a Given Point

```
context Geocoder::streetIntersectionsNear(centre:Point, distance:Real,
                    resultSrsCode:Integer, maxResults:Integer): IntersectionSearchResults
--- returns all street intersections within a given distance of a given point
```

maxResults defines the maximum number of search results returned. It cannot be larger than resultsLimit.

```
pre:   0 < maxResults and maxResults <= resultsLimit
```

resultSrsCode defines the code of the spatial reference system that intersection locations are to be returned in. resultSrsCode must be the code of a supported spatial reference system

```
pre:   spatialReferenceSystem->exists(s| s.code=resultSrsCode)
```

A valid centre must not be null and must be a valid point.

```
pre: not centre.isNull()
pre: not centre.isNull() implies centre.isValid()
```

Valid distance

```
pre: distance > 0

---    centre is assumed to be defined in the spatial reference system
---    defined by resultSrsCode
```

Valid result

```
post: result = streetIntersectionsInside(circleInSquare(centre, distance))

post:  result.matches->forAll(m | m.address.srs = resultSrsCode)
```

3.4.7 Reverse Geocode Sites Near a Given Point

```
context Geocoder::sitesNear(centre: Point, distance:Real, resultSrsCode:Integer,
```

```
                    maxResults:Integer):  SearchResults
```

Returns all sites and their addresses within a given distance of a given point. Only one address per site is returned (e.g., primary civic address or non-civic address).

maxResults defines the maximum number of search results returned. It cannot be larger than resultsLimit.

```
pre:   0 < maxResults and maxResults <= resultsLimit
```

resultSrsCode defines the code of the spatial reference system that accessPoints and mainLocations are to be returned in. resultSrsCode must be the code of a supported spatial reference system

```
pre:   spatialReferenceSystem->includes(resultSrsCode)
```

A valid centre must not be null and must be a valid point.

```
pre: not centre.isNull()
pre: not centre.isNull() implies centre.isValid()

---    centre is assumed to be defined in the spatial reference system
---    defined by resultSrsCode
```

Valid distance

```
pre: distance > 0
```

Valid result

```
post:  not result->isEmpty()implies
       result.matches->forAll(m | m.address.siteID.notEmpty())

post:  result.matches->forAll(m | Site->exists( s | m.address.siteID = s.siteID and
                      (m.address= s.civicAddress(resultSrsCode) or
                       m.address=nonCivicAddress(resultSrsCode))))

post: result.matches->forAll(m |
                    m.address.accessPointLocation.inside(circleInSquare(centre, distance))
```

3.4.8 Reverse Geocode Site Identifier

```
Context: Geocoder::site(aSiteID:String, effectiveDate:Date,resultSrsCode:Integer):
SearchResults
```

Returns all addresses associated with a given site identifier as of a given date.

The address as of the date, effectiveDate is returned

resultSrsCode defines the code of the spatial reference system that accessPoints and mainLocations are to be returned in.

```
---    resultSrsCode must be the code of a supported spatial reference system
pre:   spatialReferenceSystem->exists(s | s.code = resultSrsCode)

---    Effective date must be the today or in the past
pre:   effectiveDate <= today()

---    A site with id, siteID must exist
pre:   Site->exists(s | s.siteID = aSiteID)

post: see section 3.4.1 Geocode Structured Address
```

3.4.9 Reverse Geocode Street Intersection Identifier

```
Context: Geocoder::streetIntersection(sid:String, resultSrsCode:Integer):
IntersectionSearchResults
```

Returns address associated with a given street intersection identifier, sid

resultSrsCode defines the code of the spatial reference system that the intersection location is to be returned in.

```
---    resultSrsCode must be the code of a supported spatial reference system
pre:   spatialReferenceSystem->exists(s | s.code = resultSrsCode)


---    A street intersection with id, sid must exist
pre:   StreetIntersection->exists(i | i.intersectionID = sid)

post: see section 3.4.3 Geocode Street Intersection
```

3.4.10 Reverse Geocode Sites Within A Given Time Period and Area

```
Context: Geocoder::changedSitesInside(aoi:Polygon, startDate:Date, endDate:Date,
maxResults:Integer, resultsSrsCode:Integer): SearchResults
```

Returns all sites and their addresses inside a given area of interest that have changed within a given time period. Only one address per site is returned (e.g., primary civic address or non-civic address).

```
---    A valid area of interest (aoi) can't be null and must be a valid polygon
```

```
pre:   not aoi.isNull()
pre:   not aoi.isNull() implies aoi.isValid
---    aoi must be defined in the spatial reference system defined by resultSrsCode

---    Valid time period
pre:   startDate <= today()
pre:   endDate <= today()
pre:   startDate <= endDate

---    maxResults defines the maximum number of search results returned.
---    It cannot be larger than resultsLimit.
pre:   0 < maxResults and maxResults <= resultsLimit

---    resultSrsCode defines the code of the spatial reference system that
---     accessPoints and mainLocations are to be returned in.
---     resultSrsCode must be a supported spatial reference system
pre:   spatialReferenceSystem->includes(resultSrsCode)

---    Valid result
post:  not result->isEmpty()implies
       result.matches->forAll(m | m.address.siteID.notEmpty())


post:  result.matches->forAll(m | Site->exists( s | m.address.siteID = s.siteID and
                      (m.address= s.civicAddress(resultSrsCode) or
                       m.address=nonCivicAddress(resultSrsCode))))


post: result.matches->forAll(m | m.accessPointLocation.inside(aoi))

post:  not result->isEmpty()implies
       result.matches->forAll(m | m.score=100)

post:  not result->isEmpty()implies result.matches->exists(m| m = bestMatch)

post:  result.matches->forAll(m | m.address.srsCode = resultSrsCode)


post:  result.notEmpty() implies
             result.forAll(x | startDate <= x.changeDate and x.changeDate <= endDate)
```

3.4.11 Smart String Matcher

```
context Geocoder::smartMatch(string1:String, string2:String):  Boolean
```

smartMatch returns true if string1 approximately matches string2. This definition represents the minimum a smart matcher must do to be compliant. Two strings are separated by an edit distance of 1 if they can be made identical by the insertion, deletion, or modification of a single character. Two strings are separated by a swap distance of 1 if they can be made identical by a single swap of two adjacent characters.

smartMatch ignores case

```
post:  result = if editDistance(string1, string2) <= 1
                    and swapDistance(string1, string2) <= 1 then
                 true
               else
                 false
               endif
```

3.4.12 Site, Street, and Locality Name Expander

```
context Geocoder::expandName(aName:String):  set(String)
```

expandName returns all possible expansions of the name of a given site, street, or locality name. Expansion involves replacing abbreviated words with full words.

expandName uses specialAbbreviations which contains abbreviations commonly found in site, street, and locality names. Examples include N (North), Mt (Mount), Mtn (Mountain), CFB (Canadian Forces Base), St. (Saint), and Psg (Passage).

Case is ignored in all string comparisons

Results contain no special abbreviations

```
post:  result->forAll(x | SpecialAbbreviations->forAll(y | not x.contains(y.abbreviation) )
)
```

3.4.13 Circle in Square Generator

```
context Geocoder::circleInSquare(centre:Point, radius:real):Polygon
---    returns a square polygon that encloses a circle with a given radius(in metres)
---    and centre point (in lat/lon projection)
---    and that conforms to the Open GeoSpatial Consortium Simple Feature Specification

pre:   not centre.isNull()
pre:   not centre.isNull() implies centre.isValid
pre:   radius>0

post: result.isValid()
```

3.4.14 Geometry Reprojection

```
context Geocoder::reproject(shape: Geometry, sourceSRS:Integer,
                            resultSrsCode:Integer):Geometry
```

```
---    returns the shape reprojected into the requested spatial referencing system

---    inputSRS and outputSRS is supported
pre:   spatialReferenceSystem.code ->exists(s | s.code=sourceSRS)
       spatialReferenceSystem.code ->exists(s | s.code=resultSrsCode)

pre:   shape.notEmpty()
pre:   shape.isValid()

post:  result.isValid()
```

3.5 Constraints

1. Precision level points must be between 0 and 100

```
context PrecisionLevelPoints inv withinLegalRange
0<=points and points<=100
```

2. Match fault penalties must be between 0 and 100

```
context MatchFault inv withinLegalRange
0<=penalty and penalty<=100
```

3. resultsLimit must be positive

```
context Geocoder inv positiveResultsLimit
resultsLimit>0
```

4. someDuplicatesMaximum must be positive

```
context Geocoder inv multipleSomeDuplicatesMaximum
someDuplicatesMaximum>1
```

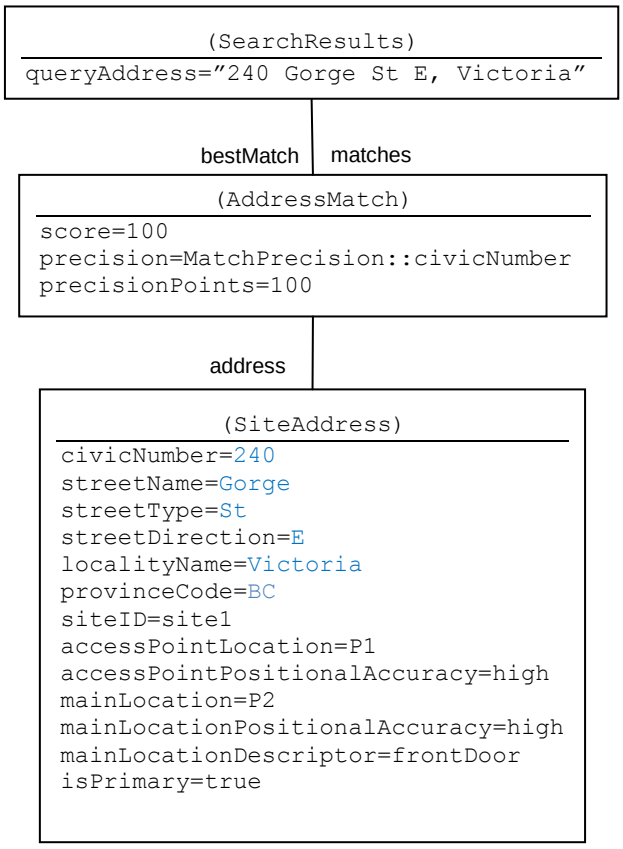
3.6 Geocoding Examples

These examples illustrate requests to geocode a site with a single address as defined in section 2.6.1 with the following additions:

- P1W is the precise access point position of 240 Gorge St W in Saanich
- P3 is the representative point (Street.asPoint) for Gorge St E in Victoria
- P4 is the interpolated position of 212 Gorge St E, Victoria
- Saanich is a LocalityAlias for Victoria (and vice-versa)

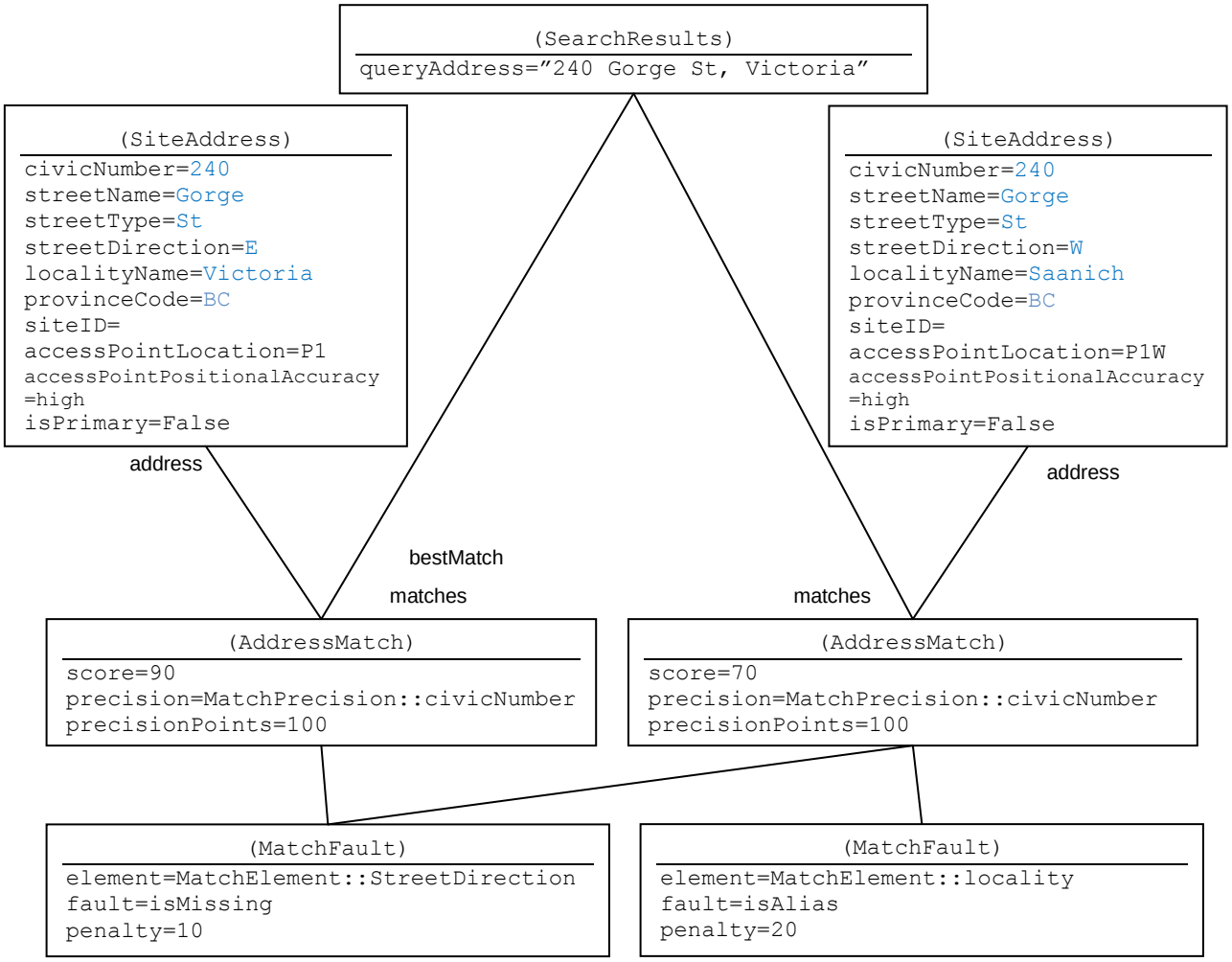
3.6.1 A Perfect Match

geocoder.geocodeUnstructureAddress(queryAddress="240 Gorge St E, Victoria", minScore=0,maxResults=100,resultSrsCode=4326)



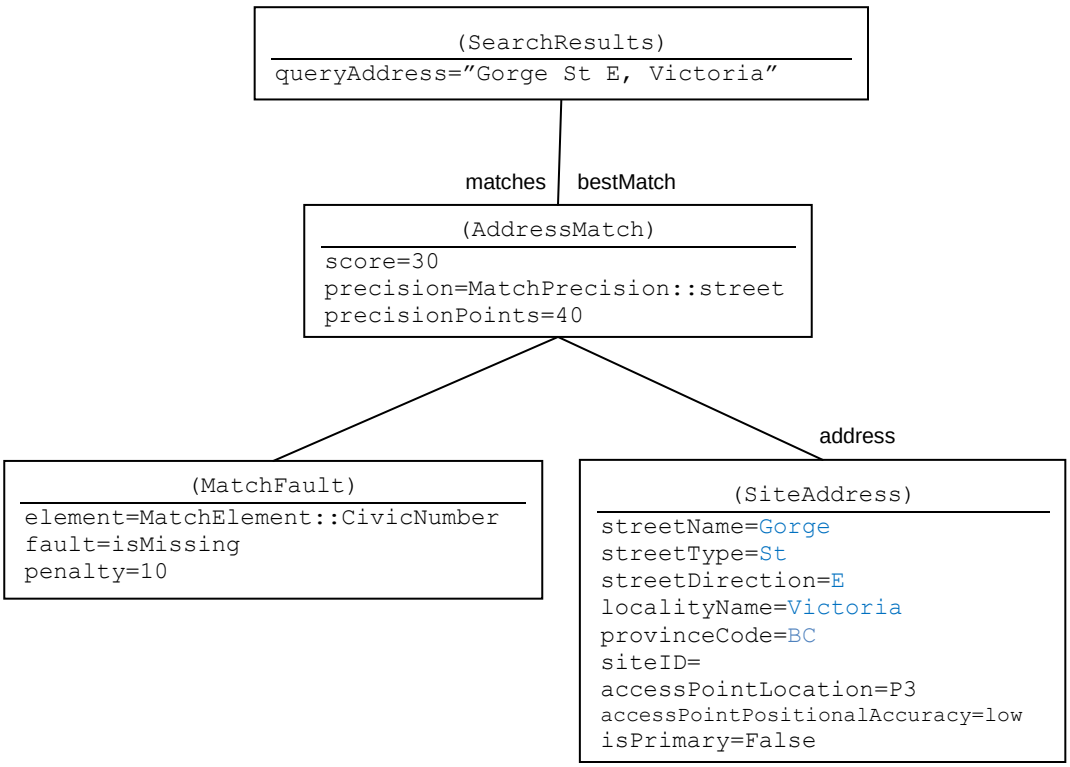
3.6.2 A Missing Street Direction

geocoder.geocodeUnstructureAddress(queryAddress="240 Gorge St, Victoria", minScore=0,maxResults=100,resultSrsCode=4326)



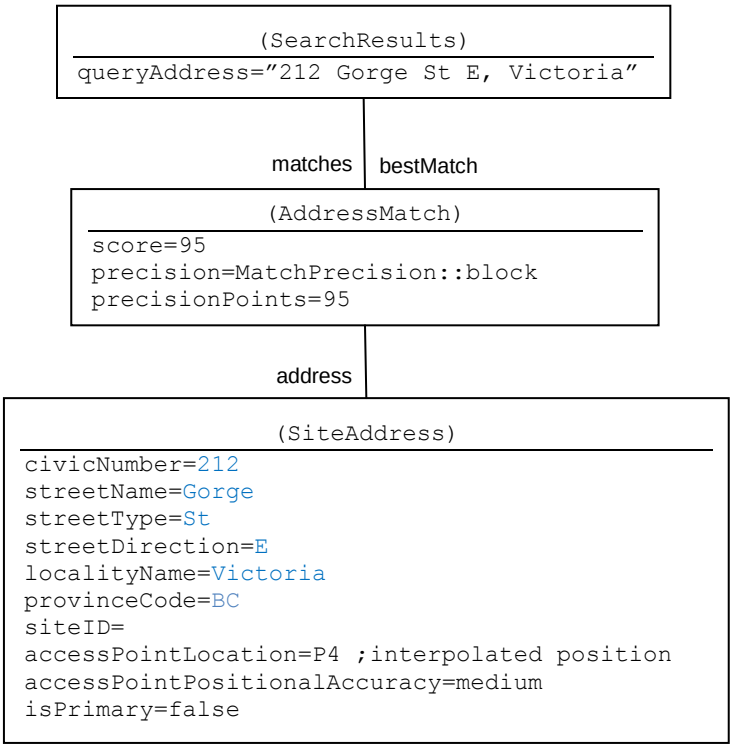
3.6.3 A Missing Civic Number

geocoder.geocodeUnstructureAddress(queryAddress=”Gorge St E, Victoria”, minScore=0,maxResults=100,resultSrsCode=4326)



3.6.4 Interpolation Required

geocoder.geocodeUnstructureAddress(queryAddress=”212 Gorge St E, Victoria”, minScore=0,maxResults=100,resultSrsCode=4326)



4. REFERENCES

R1 Canada Post Addressing Guidelines
http://www.canadapost.ca/tools/pg/manual/PGaddress-e.asp

R2 BC Geographical Names Information Service
http://geobc.gov.bc.ca/bcnames/

R3 Nova Scotia Civic Address Users Guide
http://www3.nsgc.gov.ns.ca/civic_help/V5/pdf/CivicAddressUsersGuidev.4.1.pdf

R4 Open Geospatial Consortium Simple Feature Access
http://www.opengeospatial.org/standards/sfa

R5 Unified Modelling Language 2.0
http://en.wikipedia.org/wiki/Unified_Modeling_Language

R6 Object Constraint Language 2.0
http://en.wikipedia.org/wiki/Object_Constraint_Language

R7: BC Mailing and Delivery Address Standards
http://www.cio.gov.bc.ca/other/DAF/docs/MailingDeliveryAddress_Standards.doc

R8: Towards An International Address Standard
http://www.isotc211.org/address/Copenhagen_Address_Workshop/papers/CoetzeeEtAl_TowardsAnInternationalAddressStandard_GSDI-10_2008.pdf

R9 United States Postal Service Postal Addressing Standards
http://pe.usps.gov/cpim/ftp/pubs/Pub28/Pub28.pdf

R10:Spatial Referencing for Geographical Datasets (BS7666-2006)
http://www.agi.org.uk/SITE/UPLOAD/DOCUMENT/Standards/BS7666_1.pdf

R11: ***Trouble with twins*** , James Rumbaugh, pp16-21, Journal of Object Oriented Programming; July/August 1994

R12: ESRI Address Data Model
http://support.esri.com/index.cfm?fa=downloads.dataModels.filteredGateway&dmid=32

R13: URISA Street Address Data Standard
http://www.urisa.org/about/initiatives/addressstandard

R14: AddressBC Technical Specifications v2.0 (June 12, 2008)

R15: **OpenGIS Location Service (OpenLS) Implementation Standards**

http://www.opengeospatial.org/standards/ols

R16: BC Civic Address Geocoder User Guide
http://www.data.gov.bc.ca/local/dbc/docs/geo/geocode/geocoder_user_guide_1.1.pdf

R17: Spatial Referencing System Codes
http://spatialreference.org/

R18: BC Date and Time Standard

R19: ISO 3166-2 Sub-Country Codes
http://en.wikipedia.org/wiki/ISO_3166-2:CA

R20: ISO 19112 Geographic information – Spatial referencing by geographic identifiers
http://people.ischool.berkeley.edu/~ryanshaw/pdf/ISO_19112.pdf