An Esri® Geocoding Technical Paper •

Customizing Locators in ArcGIS® 10

Esri, 380 New York St., Redlands, CA 92373-8100 USA
TEL 909-793-2853 • FAX 909-793-5953 • E-MAIL info@esri.com • WEB esri.com

Copyright © 2010 Esri
All rights reserved.
Printed in the United States of America.

The information contained in this document is the exclusive property of Esri. This work is protected under United States copyright law and other international copyright treaties and conventions. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage or retrieval system, except as expressly permitted in writing by Esri. All requests should be sent to Attention: Contracts and Legal Services Manager, Esri, 380 New York Street, Redlands, CA 92373-8100 USA.

The information contained in this document is subject to change without notice.

Esri, the Esri globe logo, ArcGIS, ArcMap, ArcCatalog, esri.com, and @esri.com are trademarks, registered trademarks, or service marks of Esri in the United States, the European Community, or certain other jurisdictions. Other companies and products mentioned herein may be trademarks or registered trademarks of their respective trademark owners.

Customizing Locators in ArcGIS 10

An Esri Geocoding Technical Paper

Contents **Page**

Introduction 1

The Geocoding Process 2

Scoring 3

The Locator Style File 5

(Locator) Grammar 8
- Aliases 9
- US States 11
- Top level elements 11
  - Location 11
  - Postal 11
  - FullAddress 11
  - FullNormalAddress 11
  - FullIntersection 11
  - NormalAddress 11
  - MultiLineAddress 12
  - OptionalUnit 12
  - MultiLineOptional Unit 12
  - MultiLineOptional UnitPrefix 12
  - FullStreetName 12
  - FullStreetName ForStd 12
  - prefix 12
  - pretype 12
  - StName 12
  - suftype 13
  - suffix 13
  - intConnector 13
  - name 13
  - NumSeparator 13
  - OptNumSeparator 13

Contents **Page**

unitAndNumber 13
MultiLineUnitAnd Number 13
MultiLineUnitAnd NumberPrefix 14
Zones 14
ZonesNoSearch 15
Basic elements 16
Coordinates 17
Spatial Operators 17
Linear Units 18
House numbers 19
Street directions 21
Prefix types 22
Suffix types 25
Unit names 33
Multiline input 35
Spelling 36

(Locator) Mapping Schemas 39

(Locator) Reference Data Styles 40

Output Formats 43

(Locator) Plugins 46

Appendixes

Appendix A: Example of Editing Locator Properties 47

Appendix B: Example of a Runtime Property 52

Appendix C: Examples of Adding Aliases 54

Appendix D: Examples of Adding Alternate Values 55

Appendix E: Example of Defining a New House Number Format 56

Appendix F: Example of Defining Custom Zone Elements and a Supporting Schema 58

Appendix G: Example of Adjusting a Mapping Schema 65

Appendix H: Example of Adjusting the Scoring Weights 71

Contents **Page**

Appendix I: Example of Adding a Top-Level Element 72

Appendix J: Example of Customizing Inputs 79

Appendix K: Example of a New Intersection Type 81

Appendix L: Adjusting Spatial Operators 84

Customizing Locators in ArcGIS 10

Introduction

Geocoding in ArcGIS® has always been customizable; this document continues support for users' needs for custom geocoding using Esri's new geocoding engine delivered in ArcGIS 10. It will be helpful to learn some basics of the new engine, after which this document will go into detail on customization options.

Perhaps the most noteworthy quality of geocoding at ArcGIS 10 compared to its predecessors is that its international applicability (any addressing standard, language, or writing system) is in the scope of a common geographic information system (GIS) geocoding platform.

ArcGIS 10 continues to use the accepted terms and workflows for geocoding that users are familiar with: Locator styles encapsulate the rules for locator creation, and locators enable geocoding by storing rules and reference data, may be stored in all ArcGIS workspace types, and may be used interactively or in batch mode either from a workspace or via a service after publication to ArcGIS Server.

Locators may be deployed in any workspace.

The concept of an address style is both retained and enhanced in ArcGIS 10. In previous versions, an address style was narrowly defined by a set of rule-base files; one style handled only one address definition with limited matching criteria that could be tuned by comparatively few parameters, necessitating redesign and proliferation of styles. ArcGIS 9.3.1, for example, shipped with 30 styles for geocoding in only the United States. In each of these 30 legacy styles, a set of rule-base files needed to be managed across all desktops where locators were to be created or rebuilt. ArcGIS 10 ships with a single U.S. style definition file encoding six address formats for the same number of use cases, and only the one file is needed for locator definition, making the new technology easier to implement and support. The last differentiator, which will not be covered by this document, is that the new geocoding engine in ArcGIS 10 is extensible through the creation of plug-ins. Locator plug-ins are a development opportunity to provide custom behavior within the locator framework.

This document will explain the structure and principles behind geocoding and locator definition, then work through a range of customization scenarios.

The Geocoding Process

To understand why and where you need to make customizations, it will help to understand the geocoding engine's matching strategy. By *matching,* we mean correspondence of input address data with reference data such as street centerlines or rooftop points having a schema supporting the desired style of address.

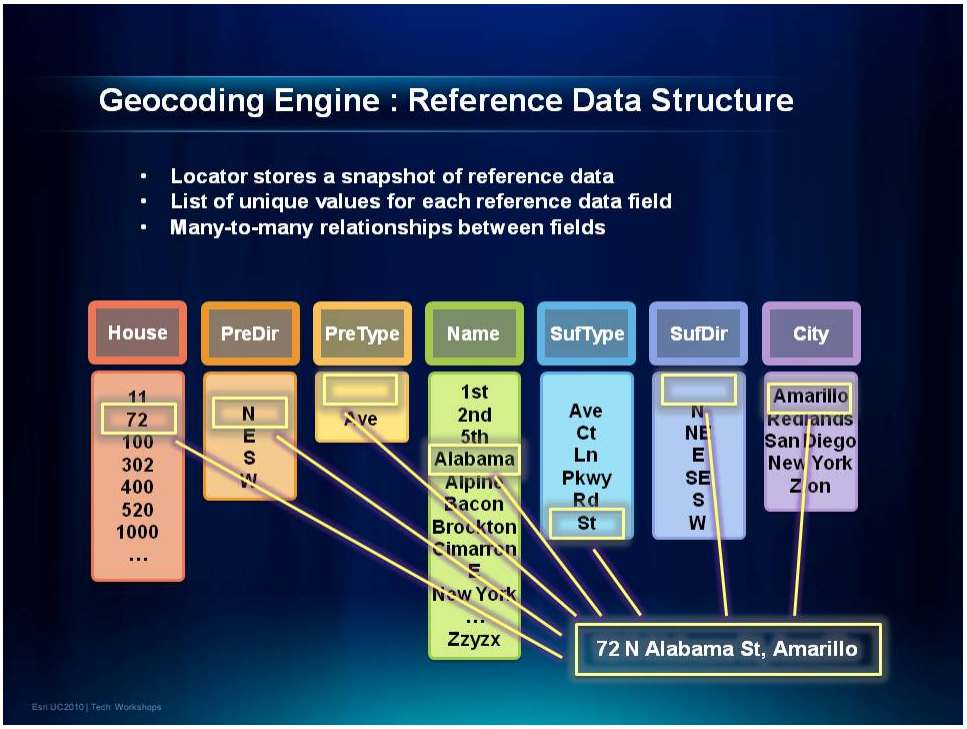
The ArcGIS 10 geocoding engine is not a search engine of the classic Web search pattern. Greatly simplified, a Web search engine takes unstructured data and looks for words in the data in its index store. Context to the search may be applied when certain word patterns are detected, but in any event, what is returned is usually a set of result candidates ranked by index match and previous search popularity. This is good for dependably returning a sufficient count of results, but not ideal for discriminating within a search context according to any kind of scoring methodology the user might have in mind. That is why search engines rely on the user to do the final selection.

Geocoding has a search context defined by the reference data used and by an understanding of the ways in which address information is commonly supplied to the engine. It is possible to apply a Web-style search to a reverse hash index built from address reference data words, but this does not handle abbreviation and aliasing well, nor is it easily adapted across addressing "cultures." For this reason, the ArcGIS 10 geocoding engine uses a constrained search filtered by the importance the locator designer puts on address elements and their variability. This lets the engine supply a single best result to support automation of the whole process.

The geocoding engine search strategy consists of the following:

- The Locator index stores a snapshot of standardized reference data, which has all address components in separate fields.
- The locator cross-references geometry against all unique values in the reference data.
- Address grammar defines the address components to be recognized.
- Inputs are searched for grammar elements invariantly expected to be present, such as house, street name, and city for U.S. styles.
- Input elements may have multiple contexts; all will be considered.
- Invariant elements are used to filter an index search.
- The index is searched starting with records matching the invariant components (matching uses computational linguistics, not simple character comparison or a soundex).
- The search is refined by matches to optionally present components.
- Candidates are scored (described below) according to weights defined by the locator.
- Candidates are returned in score rank in the form found in the reference data.

Where the grammar defines an element composed of a set of other elements, like FullStreetName, you will notice that the child elements may be defined with values including an "empty" option; this has the effect of allowing the element to be "missing" from the input yet still match the pattern. For example, if you open the USAddress.lot.xml file in your install Locators directory (e.g., C:\Program Files (x86)\ArcGIS\Desktop10.0\Locators) in a browser, you will see the element "prefix" is defined for both forms of FullStreetName but is defined as *dir* or *empty* (look in the Grammar/Top level elements section):



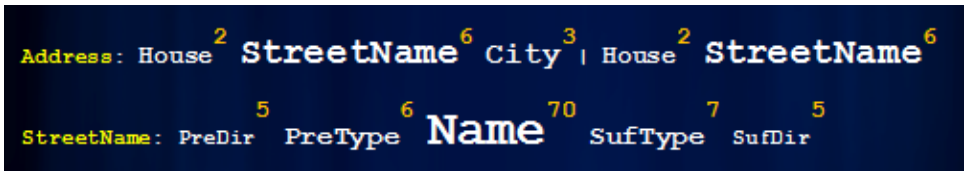
Conceptual View of Reference Data in a Locator

All the behavior described above is accessible via the locator definition file, which will be the focus of this document. Esri uses the workflow we outline below, namely to begin with an existing, functioning definition file closest to the address style you want to support and edit a copy. Do not attempt to create a locator definition file from scratch. Esri plans to support locator definition from a stub file of one example of each grammar element at a future release.

Scoring

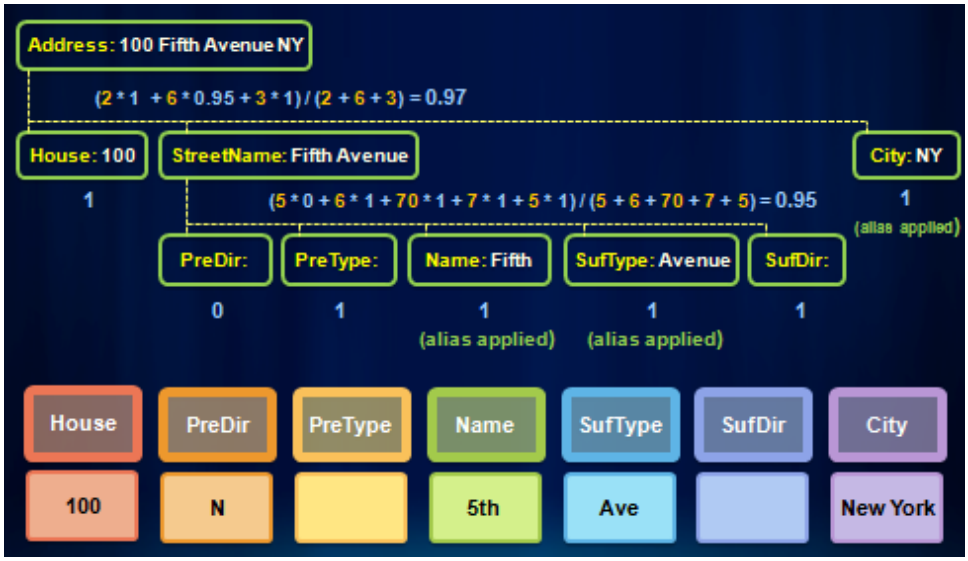
Runtime parameters that may be adjusted by the user are the **minimum match score** and the **minimum candidate score.** Successful geocodes meet at least the minimum match score, and only reference values supporting the minimum candidate score are considered. Scores are decimal numbers calculated in the range 0.0 to 1.0 according to weights defined in the locator definition but are reported in the normalized range of 1 to 100. Scores are only considered a tie if their geometry differs.

Let's illustrate score calculation with a worked example. When the engine is given an address, it parses it into recognized components, and there may be more than one successful parse.



Score Weights for a Simple Address

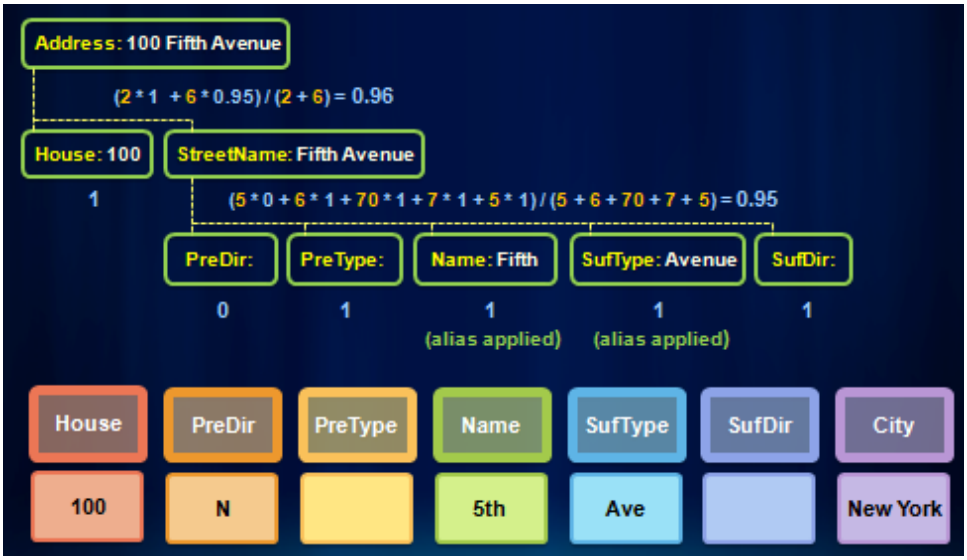
This example means that an address may be recognized as having a house number, street name, and city name *or* a house number and a street name but no city, and that a street name is composed of prefix direction, prefix type, base name, suffix type, and suffix direction. The superscripted numbers are the score weights for each element, and the font size is scaled according to the score weight. Score weights are relative values within the element and do not have to add up to any constant. Now, examine the case of an address given as "100 Fifth Avenue NY":



Score Calculation Example

The boxed values along the bottom of the graphic represent the reference data values to which we are matching. With inspection, we can see the reference values "5th" for street name, "Ave" for suffix type, and "New York" for city. These differ from the given address values but are known aliases, so the locator makes these substitutions without penalty. The final score, 0.97, is calculated by adding 1 or 0 times the weight for each basic grammar element, dividing by the weight total, then passing this value up to the next highest element, and so on. You can see the correspondence between found elements in the given address and 1 or 0 score component multipliers—0 when the address and reference data disagree and 1 when they agree.

Now, examine the situation where the given address is "100 Fifth Avenue," meaning the city name is not given, an allowed case according to the definition of the address.



You can see that the street name element is scored the same, but the score of 0.96 for the whole address is lower than the more complete situation given earlier.

This simple scoring explanation is not the full story; instead of 0 or 1 being used as the weight multiplier, a decimal number in the range 0–1 may be used in the situation where a spelling variation is allowed or a character substitution is made. Spelling correction uses a number of computational linguistics approaches such as Levenshtein distance, while character substitution may be defined in a locator to suit the language in use, for example, where a set of characters sound similar and should be considered equivalent. Character substitution supports not only common misspellings (0 and O, 1 and l), but also the situation where a telephone caller dictates an address to an operator who may not record the correct spelling for the address but nevertheless needs to reliably geocode it.

Score weights may be 0 and may add up to 0 (all weights for an element are 0), in which case they will play no part in candidate ranking but will continue to participate in address parsing; this will have its own effect on geocoding quality.

Note that the scoring approach outlined does not penalize incorrect data; it is only additive.

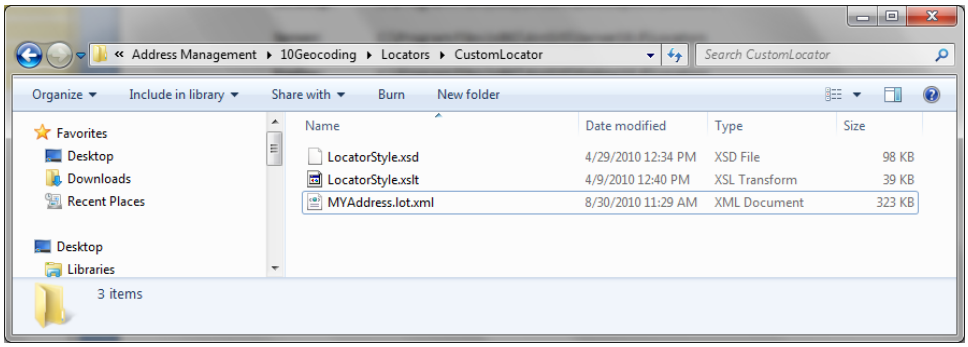
The Locator Style File

Locator styles are defined by XML files deployed in your ArcGIS 10 installation directory:

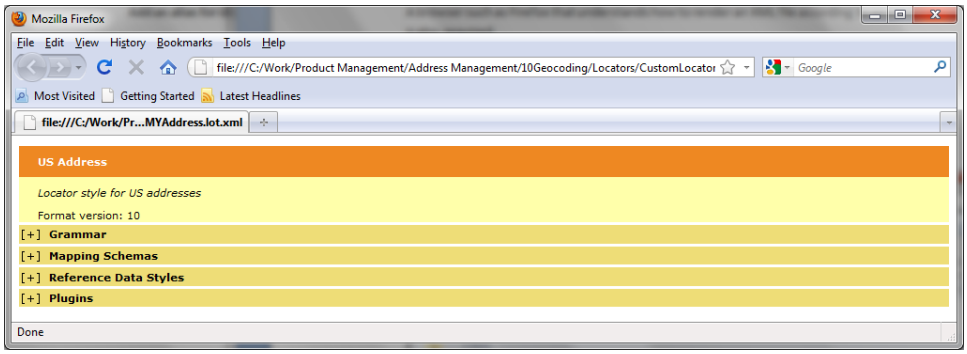
Desktop: C:\Program Files (x86)\ArcGIS\Desktop10.0\Locators
Server: C:\Program Files (x86)\ArcGIS\Server10.0\Locators
Engine: C:\Program Files (x86)\ArcGIS\Engine10.0\Locators

The U.S. style file we will be working with in these locations is named **USAddress.lot.xml**. This is a system style and will always be present. Also in the installation are XSD and XSLT files used to validate and display the XML file. These are **LocatorStyle.xsd** and **LocatorStyle.xslt**. Developer skills with XML, XSD, and XSLT files are not required to customize locator definitions; all that is required is a basic understanding of how these files interoperate and how to edit an XML file in an XML-aware editor such as NotePad++. A browser, such as Firefox, that understands how to render an XML file according to an XSLT file is also required.

Begin by copying **USAddress.lot.xml**, **LocatorStyle.xsd**, and **LocatorStyle.xslt** to a working directory. Rename **USAddress.lot.xml** to a meaningful new name (here, **MYAddress.lot.xml**) and open it in your browser.



Working Project Directory



Locator Definition File Opened with Firefox

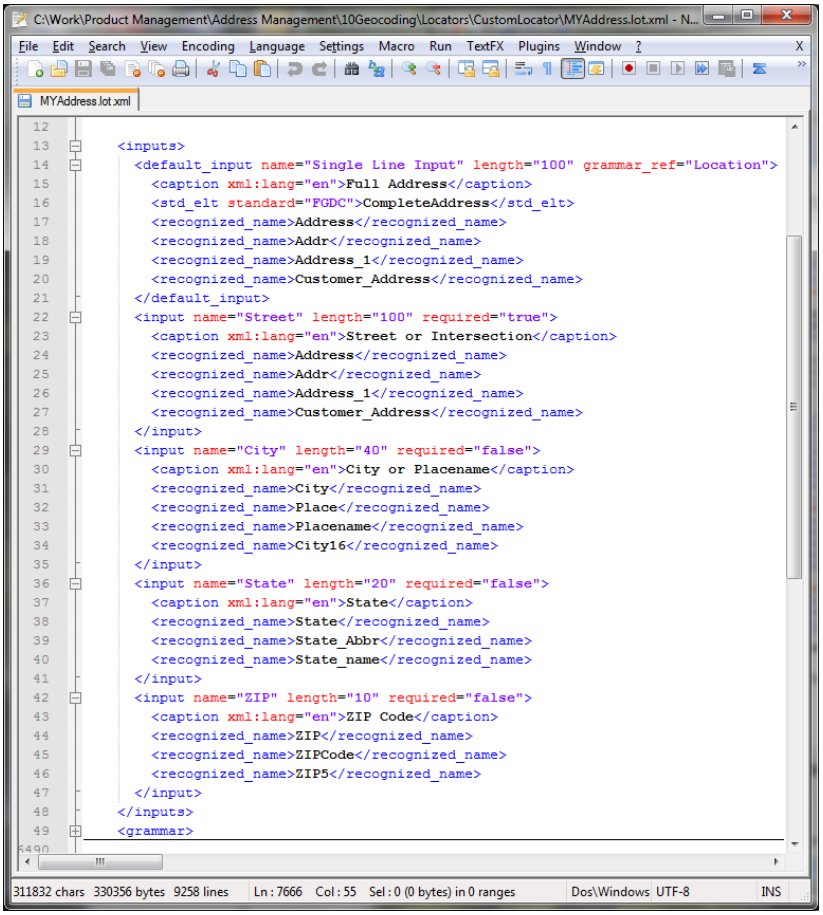
Before any edits are made, the browser still picks up the internal display string "US Address" from the XML file.

A good practice for locator customization is to create a document that describes all the customizations you want to make, such as adding, changing, or removing elements, aliases, and mapping schemas. This might be in a form you circulate in your organization as part of an approval process or use simply for your own records. The document may begin as a wish list, then develop into a checklist of steps and completion verifications.

In the browser view, you can see four expandable root elements in the XML: **Grammar**, **Mapping Schemas**, **Reference Data Styles**, and **Plugins**. The way in which the XML file is rendered in the browser is determined by the XSLT file and may vary between service packs and releases of ArcGIS, and in any event, is independent of the element order and details of the source XML, so do not be alarmed when, while editing, you see that the XML file has far more granularity than the browser view.

Open the XML file in your editor and rename the descriptive strings to agree with your chosen naming convention—here, "MY Address" and "Locator style for MY Addresses".

We will navigate the locator style file and describe its components in the order visible through the browser view—Grammar, Mapping Schemas, Reference Data Styles, and Plugins.



MYAddress.lot.xml Being Edited with Notepad++

In the image above, we can see a section named "inputs." This section is not exposed in the browser view of the style file; it controls how the Geocode Addresses geoprocessing tool appears and functions for the style. There is a default input for this style—Single Line Input—and other possible inputs that may be required or optional.

(Locator) Grammar

The Grammar section defines address elements known to the locator and their possible usage in an address. The order of grammar element topics in this document agrees with how they are displayed in a browser, but understanding of the element hierarchy begins with the top-level elements, so you may want to skip a couple of topics and begin reading "Top level elements," then return to "Aliases" and "US States."

The browser view of the locator style file has an expandable tree of elements on the left and, for each branch, a delimited set of optional component elements on the right; a colon begins the set of options, pipe characters delimit each option, and a semicolon ends the option set. For example, the Location element from the top-level elements displays like this:

```
Location        :  FullAddress
                |  Coordinates
                |  SpatialOperator
                ;
```

Interpret this as meaning a Location element may be a FullAddress element, a Coordinates element, or a SpatialOperator element. It may seem unusual that a Location may be a SpatialOperator until you follow the tag link for that element and see it includes Location in its definition (via DirectedOffset):

```
SpatialOperator :  DirectedOffset
                ;
DirectedOffset  :  positiveRealNumber^0 LinearUnits^0 Bearing^0 From^0 Location
                   { @directed_offset($1, $2, $3, $5) } format: { ['Distance'] ['Units'] " bearing " ['Bearing'] " from " ['Match_addr'] }
```

So, you have seen how to follow tag links and decompose the element hierarchy. For now, also note that the object in braces exposes how the engine uses a function @directed_offset and that the following text is commentary. All superscripted numbers are score weights; notice that a SpatialOperator has 0 score weight sum.

The browser view of the style file also shows some built-in properties of the locator, although many more optional properties are able to be defined with embedded switches; these will be described later. The behaviors visible in the browser view are only relevant in a fallback situation. Below is an example showing that a FullIntersection will only be searched for if no reasonable FullNormalAddress candidate has been found:

```
FullAddress     :  FullNormalAddress
                |  FullIntersection      | used only if no candidates with score above 10 has been previously found
                                         | used only if supportsIntersections is true
                ;
```

Another hint visible in the browser view is whether a preseparator or postseparator is required around an element. Below, we see how separators are specified for FullStreetName. If no separator is specified, then the element may be concatenated with a neighboring element.

```
FullStreetName  :  ≡prefix^5 ≡pre_type_no_sthwy^6 ≡StName^7 ^0 ≡suftype^7 ≡suffix^5
                |  ≡prefix^5 ≡pre_type_sthwy^6 OptHyphen^0 ↔StName^7 ^0 ≡suftype^7 ≡suffix^5
```

Interpret the above graphic as meaning that a FullStreetName may be made up as

- prefix + pre_type_no_sthwy + StName + suftype + suffix entirely separated, or
- Prefix + pre_type_sthwy + OptHyphen + StName + suftype + suffix, where StName may be optionally concatenated with a preceding hyphen after pre_type_sthwy

The first form might be like "North Avenue Walnut Road East," and the second like "North Road Number 6 West" or "I-10."

The full set of separator hints is as follows:

←	`pre_separator = 'none'`
↔	`pre_separator  = 'optional'` `post_separator = 'optional'`
→	`post_separator = 'none'`
≡	`pre_separator  = 'required'` `post_separator = 'required'`

Separators are a white space or one of a set of characters specified in the XML.

Aliases

Aliases in this style are defined for street names, cities, and states.

Aliases are commonly recognized values for elements and may be sets of alternate literal values on a line or tag references for a value set defined (and probably also used) elsewhere. They are used to support word substitution (equivalence) between input addresses and reference data.

```
[-] Grammar

  [-] Aliases

  StreetNameAliases
     rl          "rl" "real"
     lincoln     "lincoln" "abraham lincoln" "abe lincoln" "a lincoln"
     alt         "alt" "alternate"
     armor       "armor" "armrd" "armd" "armored"
     _ave        _ave
```

The graphic above shows a few street name aliases. It does not matter whether you define aliases with their common abbreviation as the root name or a fully spelled version. Note the alias named "_ave". A convention used in the locator style file is to precede tag reference names with an underscore.

For the _ave tag, we can see the set of values recognized for the suffix type for Avenue is referred to in the street name aliases.

```
_ave    :  "Ave"
        |  "Avd"
        |  "Av"
        |  "Aveflr"
        |  "Aven"
        |  "Avenu"
        |  "Avenue"
        |  "Avn"
        |  "Avnue"
        |  "Avenue of the"
        |  "Ave of the"
        |  "Av of the"
        ;
```

Because street names can include pretty much anything, there are other cases where separately defined elements are referred to—notably, U.S. states. You may notice that the aliases defined for states as an element in their own right are different from those defined in street name word aliases (see "calfornia"):

```
StateAliases
    ak      "ak" "alas" "alaska"
    al      "al" "ala" "alabama"
    ar      "ar" "ark" "arkansas"
    az      "az" "ariz" "arizona"
    ca      "ca" "cal" "calif" "california" "calfornia"
```

State Aliases in the Aliases Section

```
st_ak   :  "ak"
        |  "alas"
        |  "alaska"
        ;
st_al   :  "al"
        |  "ala"
        |  "alabama"
        ;
st_ar   :  "ar"
        |  "ark"
        |  "arkansas"
        ;
st_az   :  "az"
        |  "ariz"
        |  "arizona"
        ;
st_ca   :  "ca"
        |  "cal"
        |  "calif"
        |  "california"
```

State Values in the US States Section

This is to allow more flexibility for finding states used in street names than when finding states as state elements, which are handled more strictly. It is expected that more spelling leeway be required in street names.

US States

US States are defined as the set of their common abbreviations and spellings, with some including compass quadrant words that have their own set of abbreviations.

Top level elements

There are 25 top-level elements for this locator. These are the building blocks of all address formats the locator can understand.

Location

Location is what an address defines; everything begins here. If you navigate from FullAddress, you can reach every other grammar element.

```
Location        :  FullAddress
                |  Coordinates
                |  SpatialOperator
                ;
```

Postal

This is the authoritative postal zone and has more than one form in the United States, so it is linked to its own section where these forms are defined. The content in braces is a hint that a particular search context applies for the element. The engine manages sets of tests for elements within search contexts; these are discussed later in this document.

```
Postal          :  GenZIP
                   { ['#ZIPSearch'] }
                ;
```

FullAddress

The locator understands street addresses and centerline intersections.

```
FullAddress     :  FullNormalAddress
                |  FullIntersection
                ;
```

FullNormalAddress

This is from FullAddress. The content in braces is a hint that a search context applies for the element.

```
FullNormalAddress  :  NormalAddress[70] OptSeparator[0] Zone[20]
                      { ['#addressSearch'] }
                   ;
```

FullIntersection

This is from FullAddress. The content in braces is a hint that a function is used for the element—in this case, the intersection function.

```
FullIntersection   :  FullStreetName[40] intConnector FullStreetName[40] OptSeparator Zone[20]
                      { @intersection( ['#leftStreet'] , ['#rightStreet'] ) }
                   ;
```

NormalAddress

This is from FullNormalAddress. A valid customization for international jurisdictions might be to allow a form with OptionalUnit preappended to the address. Note that the House element supports some complex forms but is still intended to identify a unique delivery address; use OptionalUnit to model multitenanted structures. Note also that in this style, FullStreetName requires pre- and postseparators and that unit information is expected to follow the base address information.

```
NormalAddress      :  House[15] ≡FullStreetName[40] OptionalUnit[5]
                   ;
```

MultiLineAddress

MultiLineAddress and its subsidiary elements, MultiLineOptionalUnitPrefix and MultiLineOptionalUnit, support batch geocoding fallback situations where unit information may be confounded with street address details.

```
MultiLineAddress        :  MultiLineOptionalUnitPrefix House FullStreetName MultiLineOptionalUnit
                        ;
```

OptionalUnit

```
OptionalUnit            :
                        |  unitAndNumber
                        ;
```

MultiLineOptional Unit

```
MultiLineOptionalUnit   :
                        |  MultiLineUnitAndNumber
                        ;
```

MultiLineOptional UnitPrefix

```
MultiLineOptionalUnitPrefix  :
                             |  MultiLineUnitAndNumberPrefix
                             ;
```

FullStreetName

There are two forms here, special cases for highways being the second. In the United States, there are a number of forms of street naming that use street types appended to the street name, for example, "Highway of the Americas."

```
FullStreetName          :  ≡prefix≡pre_type_no_sthwy ≡StName≡suftype ≡suffix
                        |  ≡prefix≡pre_type_sthwy OptHyphen ↔StName≡suftype ≡suffix
                        ;
```

FullStreetName ForStd

This element enables casting prefix and suffix elements to StName values, as in "Park Avenue." A valid customization for a new case like "The Drive" being an intended StName value would be to add "The" to prefix types.

```
FullStreetNameForStd    :  ≡prefix≡pre_type_no_sthwy ≡StName≡suftype ≡suffix
                        |  ≡prefix≡pre_type_sthwy OptHyphen ↔NumericAlpha≡suftype ≡suffix
                        |  ≡prefix≡≡pre_type≡suftype ≡suffix
                        |  ≡dir≡suftype ≡suffix
                        ;
```

prefix

```
prefix                  :  dir
                        |
                        ;
```

Note the OR condition with an empty value.

pretype

```
pretype                 :  pre_type
                        |
                        ;
```

Note the OR condition with an empty value.

StName

```
StName                  :  name
                        ;
```

Name elements are sets of words not otherwise recognized. Words may be hyphenated.

suftype

```
suftype          :  suf_type
                 |
                 ;
```

Note the OR condition with an empty value.

suffix

```
suffix           :  dir
                 |
                 ;
```

Note the OR condition with an empty value.

intConnector

```
intConnector     :  andConnector
                 |  "at"
                 |  "@"
                 |  "/"
                 |  "|"
                 ;
```

These are street intersection connectors, as in "New York St & Redlands Blvd."

name

```
name             :  word
                 |  word name
                 |  word "-" name
                 ;
```

A name is no different from a word list, except that it may contain hyphens.

NumSeparator

```
NumSeparator     :  "-"
                 |  "#"
                 |  "/"
                 ;
```

This is used in unitAndNumber, like "Apt #9."

OptNumSeparator

```
OptNumSeparator  :  ↔NumSeparator↔
                 |
                 ;
```

Note the OR condition with an empty value and that pre- and postseparators are optional.

unitAndNumber

```
unitAndNumber    :  HousingUnitType OptNumSeparator AlphaNumericUnit
                 |  ↔NumSeparator↔AlphaNumericUnit
                 |  ↔NumSeparator↔latinAlphaWord
                 ;
```

These support finding unit forms such as "Suite 2A" or "#3"; in the default style, unit information is expected to follow the base address information.

MultiLineUnitAndNumber

```
MultiLineUnitAndNumber  :  HousingUnitType OptNumSeparator UnitInfo
                        |  ↔NumSeparator↔UnitInfo
                        |  UnitInfo
                        ;
```

MultiLineUnitAndNumberPrefix

```
MultiLineUnitAndNumberPrefix : HousingUnitType OptNumSeparator AlphaNumericUnit≡
                             | AlphaNumericUnit HousingUnitType≡
                             | ↔NumSeparator↔AlphaNumericUnit≡
                             | wordlist
                             ;
```

This completes the Top level elements definition section.

Zones

Zones for this locator include City, State, and ZIP. Note that for ZIP information, the 5-digit and ZIP+4, 9-digit forms are supported.

```
[−] Zones
Zone       : OptCity20 OptState20 OptGenZIP50
           | OptCityNoSearch10 OptStateNoSearch20 GenZIP50
           | City50 OptState20 OptZipNoSearch10
           ;
OptState   : State
           |
           ;
OptCity    : City
           |
           ;
OptGenZIP  : GenZIP
           |
           ;
City       : wordlist
           ;
State      :
           ;
GenZIP     : ZIP5 OptPlus4
           ;
ZIP5       : "`[0-9][0-9][0-9][0-9][0-9]`"
           ;
ZIP4       : "`[0-9][0-9][0-9][0-9]`"
           ;
ZIPPlus4   : ZIP5100 OptHyphen0 ↔ZIP450
           ;
OptPlus4   :
           | OptHyphen0 ↔ZIP450
           ;
```

Note the regular expression syntax for ZIP5 and ZIP4 elements. The expressions mean any combination of exactly 5- and 4-digit numbers, respectively, including with a leading 0.

The Zones elements named "Opt*" are defined as per their non-Opt counterparts but include empty alternate values, meaning that they may be missing in an address where they are used.

ZonesNoSearch

A NoSearch zone element in a definition means that the engine will not use the zone value in its search dictionary to restrict the search of nonzone fields but will still score the zone field. This approach is indicated when you expect zone values to be erratically supplied (or guessed) in input addresses, but you want plausible candidates evaluated.

```
[-] ZonesNoSearch
OptCityNoSearch   :  CityNoSearch
                  |
                  ;
CityNoSearch      :  wordlist
                  ;
OptStateNoSearch  :  StateNoSearch
                  |
                  ;
StateNoSearch     :  wordlist
                  ;
OptZipNoSearch    :  GenZIPNoSearch
                  |
                  ;
GenZIPNoSearch    :  ZIP5NoSearch OptPlus4
                  ;
ZIP5NoSearch      :  "`[0-9][0-9][0-9][0-9][0-9]`"
                  ;
```

Basic elements

These define character sequences to be recognized.

```
[−] Basic elements
realNumber          :  OptHyphen positiveRealNumber
                    ;
positiveRealNumber  :  number OptDecimal
                    ;
OptDecimal          :  ←"." ←number
                    |
                    ;
number              :  "`[0-9]+`"
                    ;
wordlist            :  word
                    |  word wordlist
                    ;
latinAlphaWord      :  "`[a-zA-Z]+`"
                    ;
alphaNumericWord    :  "`[a-zA-Z0-9]+`"
                    ;
NumericAlpha        :  number ←OptHyphen→latinAlphaWord
                    |  number
                    ;
word                :  "`[^ \t,;#$%^&*@!~|":]+`"
                    ;
UnitInfo            :  "`[^#&]+`"
                    |
                    ;
Hyphen              :  "`[−−−]`"
                    |  "-"
                    ;
OptHyphen           :  ↔Hyphen↔
                    |
                    ;
Separator           :  "."
                    |  ","
                    |  "|"
                    ;
OptSeparator        :  ↔Separator↔
                    |
                    ;
```

Again, note the use of regular expression syntax:

- Number—One or more occurrences of integers in the range 0–9
- latinAlphaWord—One or more latin alphabet characters in any case
- alphaNumericWord—As above but also allowing integers
- word—Anything not white space or the given punctuation characters
- UnitInfo—Anything not # or &, intended to match noise details for units
- Hyphen—Any of "–", "—" or "―" or a literal hyphen: "-"

Regular expressions give you a way to define elements that are better defined as patterns than sets of literals. User-defined identifiers for features are often promulgated as a standard arrangement of letters, numbers, and punctuation that you can represent as regular expressions to match an unlimited number of values.

Coordinates

Locators understand World Geodetic System (WGS) coordinates of the form W 117.3, N 39.7 and -117.3, 39.7. You might customize this section to recognize another datum or a prefix character taken from another language.

```
[-] Coordinates
OptCoordsysID   :  "WGS_1984"
                |
                ;
CoordSeparator  :  " "
                |  ","
                |  ";"
                |  "|"
                |
                ;
Coordinates     :  longitudePrefix ↔positiveRealNumber CoordSeparator ↔latitudePrefix ↔positiveRealNumber OptCoordsysID
                   { @create_point($1, $2, $4, $5, $6) }
                ;
longitudePrefix :  "w"
                |  "e"
                |  "-"
                |  "+"
                |
                ;
latitudePrefix  :  "n"
                |  "s"
                |  "-"
                |  "+"
                |
                ;
```

Spatial Operators

You may apply an offset to an address, as in "150 meters north from 380 New York Street Redlands CA."

A valid customization here would be to add "of" or "heading" to the From values.

```
[-] Spatial Operators
SpatialOperator :  DirectedOffset
                ;
DirectedOffset  :  positiveRealNumber LinearUnits Bearing From Location
                   { @directed_offset($1, $2, $3, $5) } format: { ['Distance'] ['Units'] " bearing " ['Bearing'] " from " ['Match_addr'] }
                ;
From            :  "from"
                ;
Bearing         :  north
                   {0.}
                |  "nne"
                   {22.5}
                |  northeast
                   {45.}
                |  "ene"
                   {67.5}
                |  east
                   {90.}
                |  "ese"
                   {112.5}
                |  southeast
                   {135.}
                |  "sse"
                   {157.5}
                |  south
                   {180.}
                |  "ssw"
                   {202.5}
                |  southwest
                   {225.}
                |  "wsw"
                   {247.5}
                |  west
                   {270.}
                |  "wnw"
                   {292.5}
                |  northwest
                   {315.}
                |  "nnw"
                   {337.5}
                |  "bearing" realNumber
                   { $2 }
                ;
```

Linear Units

```
[-] Linear Units
LinearUnits   :  Inches
                 {1}
              |  Points
                 {2}
              |  Feet
                 {3}
              |  Yards
                 {4}
              |  Miles
                 {5}
              |  NauticalMiles
                 {6}
              |  Millimeters
                 {7}
              |  Centimeters
                 {8}
              |  Decimeters
                 {12}
              |  Meters
                 {9}
              |  Kilometers
                 {10}
              ;
Inches        :  ↔""≡
              |  "In"
              |  "Inch"
              |  "Inches"
              ;
Points        :  "Pt"
              |  "Point"
              |  "Points"
              ;
Feet          :  ↔''≡
              |  "Ft"
              |  "Foot"
              |  "Feet"
              ;
```

```
Yards         :  "Yd"
              |  "Yrd"
              |  "Yrds"
              |  "Yard"
              |  "Yards"
              ;
Miles         :  "mi"
              |  "Mile"
              |  "Miles"
              ;
NauticalMiles :  "Nautical Mile"
              |  "Nautical Miles"
              ;
Millimeters   :  "mm"
              |  "Millimeter"
              |  "Millimeters"
              ;
Centimeters   :  "cm"
              |  "Centimeter"
              |  "Centimeters"
              ;
Decimeters    :  "dm"
              |  "Decimeter"
              |  "Decimeters"
              ;
Meters        :  "m"
              |  "Meter"
              |  "Meters"
              ;
Kilometers    :  "km"
              |  "Kilometer"
              |  "Kilometers"
              |  "Klick"
              |  "Klicks"
              ;
```

These enumerations agree with Esri standard values; you might add *Metre* and *Metres* for international usage.

House numbers

```
[-] House numbers
_HouseNum              :  AlphaNumericHouse
                       |  NyNumber
                       |  WisconsinIllinoisNumber
                       ;
House                  :  _HouseNum
                       |

                       ;
NyNumber               :  crossstreet "-" ↔number
                       ;
WisconsinIllinoisNumber :  wisc_grid ←number
                       |  wisc_dir ←number
                       |  number ←illinois_dir→number
                       ;
wisc_grid              :  wisc_dir ←number→wisc_dir
                       ;
illinois_dir           :  "n"
                       |  "s"
                       |  "w"
                       ;
wisc_dir               :  illinois_dir
                       |  "e"
                       |  "r"
                       ;
crossstreet            :  number↔
                       ;
AlphaNumericHouse      :  number OptFraction
                       |  alpha
                       |  alpha ↔OptHyphen↔number
                       |  number ↔Hyphen↔alpha
                       |  number→alpha
                       |  number "-" number alpha
                       |  Fraction
                       ;
```

```
OptHyphenAlphaNum   :  ←Hyphen↔alphaNumericWord
                    |
                    ;
AlphaNumericUnit    :  number OptFraction
                    |  alphaNumericWord ←OptHyphenAlphaNum
                    ;
OptFraction         :
                    |  Fraction
                    ;
Fraction            :  number→"/" ←number
                    |  "¼"
                    |  "½"
                    |  "¾"
                    |  "⅓"
                    |  "⅔"
                    |  "⅕"
                    |  "⅖"
                    |  "⅗"
                    |  "⅘"
                    |  "⅙"
                    |  "⅚"
                    |  "⅛"
                    |  "⅜"
                    |  "⅝"
                    |  "⅞"
                    ;
alpha               :  "`[a-zA-Z]`"
                    ;
```

Let's look at a few cases of House numbers supported by the above definitions, as local variation in delivery addresses will be a common customization requirement. AlphaNumericHouse and AlphaNumericUnit are the principal elements; examining the subordinate elements, we see that the following forms are supported:

Number OptFraction	"380", "380 ½"
Alpha	"B"
Alpha OptHyphen number	"C380", "C-380"
Number Hyphen alpha	"380-C"
Number alpha	"380C", "380 C"
Number "-" number alpha	"380-12B"
Fraction	"1/2"
Number OptFraction	"380 ½"
alphaNumericWord OptHyphenAlphaNum	"ROOM6", "ROOM6—TOWER2"

Street directions

```
[-] Street directions
dir          :  strictdir
             |  halfdir
             |  bounddir
             ;
strictdir    :  west
             |  east
             |  south
             |  north
             ;
halfdir      :  southeast
             |  southwest
             |  northeast
             |  northwest
             ;
bounddir     :  northbound
             |  southbound
             |  eastbound
             |  westbound
             ;
west         :  "W"
             |  "West"
             |  "West Side"
             ;
east         :  "E"
             |  "East"
             |  "East Side"
             ;
south        :  "S"
             |  "So"
             |  "South"
             |  "South Side"
             ;
north        :  "N"
             |  "No"
             |  "North"
             |  "North Side"
             ;
```

```
southeast    :  "SE"
             |  "Southeast"
             ;
southwest    :  "SW"
             |  "Southwest"
             ;
northeast    :  "NE"
             |  "Northeast"
             ;
northwest    :  "NW"
             |  "Northwest"
             ;
northbound   :  "NB"
             |  "Northbound"
             ;
southbound   :  "SB"
             |  "Southbound"
             ;
eastbound    :  "EB"
             |  "Eastbound"
             ;
westbound    :  "WB"
             |  "Westbound"
             ;
```

A valid customization for Street directions would be to add values for another language to be recognized.

Prefix types

```
[-] Prefix types
_cty          :  "County"
              |  "Cnty"
              |  "Coun"
              |  "Cty"
              |  "Co"
              ;
_ctyln        :  _cty _ln
              ;
_coast        :  "Coast"
              |  "Cst"
              |  "Co"
              ;
_csthwy       :  _coast _hwy
              ;
_cord         :  _cty _rd
              |  "Cord"
              |  "Co rd"
              |  "Coord"
              |  "Cr"
              |  "Cth"
              |  "HC"
              |  "HCR"
              |  "HCO"
              |  "HWC"
              ;
_cordso       :  _cord south
              ;
_cordse       :  _cord southeast
              ;
_cordsw       :  _cord southwest
              ;
_cohwy        :  _cty _hwy
              |  _cty "trunk" _hwy
              ;
_cohwyne      :  _cty _hwy northeast
              ;
_cohwynw      :  _cty _hwy northwest
              ;
_cohwyse      :  _cty _hwy southeast
              ;
_cohwysw      :  _cty _hwy southwest
              ;
_corte        :  _cty _rte
              |  "Cort"
              |  "Corte"
              ;
```

```
_coc          :  "Coc"
              |  "Co c"
              ;
_cntry        :  "Country"
              |  "Cntry"
              |  "Cnty"
              |  "Cty"
              |  "Coun"
              ;
_cntryrd      :  _cntry _rd
              |  _cntry _rte
              |  _cntry _hwy
              ;
_dev          :  "Development"
              |  "Developement"
              |  "Develop"
              |  "devlpmnt"
              ;
_fm           :  "Fm"
              ;
_h            :  "H"
              ;
_hist         :  "Historic"
              |  "Hist"
              ;
_histrte      :  _hist _rte
              ;
_i            :  "I"
              |  "Interstate"
              ;
_indian       :  "Indian"
              |  "Ind"
              |  "In"
              ;
_indrte       :  _indian _rte
              ;
_natl         :  "National"
              |  "Natl"
              |  "Nat"
              |  "Ntl"
              ;
_no           :  "Number"
              |  "Num"
              |  "No"
              ;
_old          :  "Old"
              |  "Older"
              ;
```

```
_oldhwy        :  _old _us _hwy
               |  _old _hwy
               |  _old _us
               ;
_oldcty        :  _old _cty
               ;
_oldctyhwy     :  _old _cty _hwy
               |  _old _cty _rd
               |  _old _cty _rte
               ;
_oldcntryrd    :  _old _cntry _rd
               |  _old _cntry _hwy
               |  _old _cntry
               ;
_oldsthwy      :  _old _sthwy
               ;
_parkrd        :  _park _rd
               ;
_rdno          :  _rd _no
               ;
_rec           :  "Recreational"
               |  "Recreation"
               |  "Recl"
               |  "Rec"
               ;
_recrd         :  _rec _rd
               ;
_rm            :  "Rm"
               ;
_rr            :  "RR"
               |  "Railroad"
               |  "Rail" _rd
               ;
_rrte          :  "RR"
               |  _rte
               |  "Rural" _rte
               ;
_stt           :  "State"
               |  "St"
               ;
_sr            :  "State Rd"
               |  "SR"
               |  _state _rd
               |  _stt _rd
               ;
```

```
_twp           :  "Township"
               |  "Town Ship"
               |  "Twp"
               |  "Tw"
               |  "Ts"
               ;
_twhy          :  _twp _hwy
               ;
_us            :  "US"
               |  "Unites States"
               |  "US" _rte
               ;
_usfs          :  "USFS"
               |  "Us fs"
               |  _us "Forest Service"
               |  "Forest Service"
               ;
_usfsrd        :  _usfs _rd
               |  _usfs _rte
               |  _usfs _hwy
               |  _usfs _hwy _rte
               |  _natl _frst _dev
               |  _natl _frst _dev _rd
               |  _frst _rd
               |  _frst _rte
               |  _frst _hwy _rte
               ;
pre_type       :  pre_type_no_sthwy
               |  pre_type_sthwy
               ;
```

```
pre_type_no_sthwy :  _ave
                  |  _avenida
                  |  _blvd
                  |  _byp
                  |  _cll
                  |  _ctyln
                  |  _csthwy
                  |  _cord
                  |  _cordso
                  |  _cordse
                  |  _cordsw
                  |  _cohwy
                  |  _cohwyne
                  |  _cohwynw
                  |  _cohwyse
                  |  _cohwysw
                  |  _corte
                  |  _coc
                  |  _cntryrd
                  |  _dr
                  |  _expy
                  |  _fwy
                  |  _fm
                  |  _h
                  |  _hwy
                  |  _histrte
                  |  _i
                  |  _indrte
                  |  _no
                  |  _old
                  |  _oldhwy
                  |  _oldcty
                  |  _oldctyhwy
                  |  _oldcntryrd
                  |  _oldsthwy
                  |  _parkrd
                  |  _ramp
                  |  _recrd
                  |  _rte
                  |  _rdno
                  |  _rm
                  |  _rr
                  |  _rrte
                  |  _rue
                  |  _twhy
                  |  _usfsrd
                  |  _ushwy
                  |  _via
                  |
                  ;
```

```
pre_type_sthwy  :  _cord
                |  _cordso
                |  _cordse
                |  _cordsw
                |  _cohwy
                |  _cohwyne
                |  _cohwynw
                |  _cohwyse
                |  _cohwysw
                |  _corte
                |  _cntryrd
                |  _fwy
                |  _hwy
                |  _indrte
                |  _i
                |  _no
                |  _oldhwy
                |  _oldcty
                |  _oldctyhwy
                |  _oldcntryrd
                |  _oldsthwy
                |  _rdno
                |  _sthwy
                |  _strte
                |  _sr
                |  _twhy
                |  _us
                |  _ushwy
                ;
```

Suffix types

```
[-] Suffix types
_acrd    :  "Acrd"
         ;
_aly     :  "Aly"
         |  "Allee"
         |  "Alley"
         |  "Ally"
         |  "Al"
         ;
_anx     :  "Anx"
         |  "Annex"
         |  "Annx"
         |  "Anex"
         ;
_arc     :  "Arc"
         |  "Arcade"
         ;
_avct    :  "Avct"
         |  "Avect"
         |  "Avenue" _ct
         |  "Ave" _ct
         |  "Av" _ct
         ;
_avdr    :  "Avdr"
         |  "Avedr"
         ;
_ave     :  "Ave"
         |  "Avd"
         |  "Av"
         |  "Aveflr"
         |  "Aven"
         |  "Avenu"
         |  "Avenue"
         |  "Avn"
         |  "Avnue"
         |  "Avenue of the"
         |  "Ave of the"
         |  "Av of the"
         ;
_avenida :  "Avenida"
         |  "Ave"
         ;
_avln    :  "AvLn"
         |  "AveLn"
         ;
_avpl    :  "AvPl"
         |  "AvePl"
         ;
_bch     :  "Bch"
         |  "Beach"
         ;
```

```
_bg      :  "Bg"
         |  "Burg"
         ;
_bgs     :  "Bgs"
         |  "Burgs"
         ;
_blf     :  "Blf"
         |  "Bluf"
         |  "Bluff"
         ;
_blfs    :  "Blfs"
         |  "Bluffs"
         ;
_blvd    :  "Blvd"
         |  "Bv"
         |  "Bl"
         |  "Blv"
         |  "Bd"
         |  "Boul"
         |  "Boulevard"
         |  "Boulv"
         |  "Bvd"
         |  "Bvld"
         ;
_bnd     :  "Bnd"
         |  "Bend"
         ;
_br      :  "Br"
         |  "Brnch"
         |  "Branch"
         ;
_brg     :  "Brg"
         |  "Bridge"
         |  "Brdge"
         ;
_brk     :  "Brk"
         |  "Brook"
         ;
_brks    :  "Brks"
         |  "Brooks"
         ;
_btm     :  "Btm"
         |  "Bot"
         |  "Bottm"
         |  "Bottom"
         ;
_bus     :  "Bus"
         |  "Business"
         ;
```

```
_byp     :  "Byp"
         |  "Bypa"
         |  "Bypas"
         |  "Bypass"
         |  "Byps"
         ;
_byu     :  "Byu"
         |  "Bayoo"
         |  "Bayou"
         ;
_cam     :  "Cam"
         |  "Cami"
         |  "Camino"
         ;
_cer     :  "Cerrada"
         |  "Cer"
         ;
_cir     :  "Cir"
         |  "Ci"
         |  "Circulo"
         |  "Circ"
         |  "Circl"
         |  "Circle"
         |  "Corcle"
         |  "Cr"
         |  "Crcl"
         |  "Crcle"
         |  "Crl"
         ;
_cirs    :  "Cirs"
         |  "Circles"
         ;
_clb     :  "Clb"
         |  "Club"
         ;
_clf     :  "Clf"
         |  "Cliff"
         ;
_clfs    :  "Clfs"
         |  "Cliffs"
         ;
_cll     :  "Cll"
         |  "Calle"
         ;
_cmn     :  "Cmn"
         |  "Common"
         ;
_cmns    :  "Cmns"
         |  "Commons"
         ;
```

```
_cmt      :  "Cmt"
          |  "Caminito"
          ;
_cor      :  "Cor"
          |  "Corner"
          ;
_cors     :  "Cors"
          |  "Corners"
          ;
_cp       :  "Cp"
          |  "Camp"
          |  "Cmp"
          ;
_cpe      :  "Cpe"
          |  "Cape"
          ;
_cres     :  "Cres"
          |  "Crecent"
          |  "Crescent"
          |  "Crescnt"
          |  "Cresent"
          |  "Crscnt"
          |  "Crsent"
          |  "Crsnt"
          ;
_crk      :  "Crk"
          |  "Ck"
          |  "Cr"
          |  "Creek"
          ;
_crse     :  "Crse"
          |  "Course"
          ;
_crst     :  "Crst"
          |  "Crest"
          ;
_cswy     :  "Cswy"
          |  "Csway"
          |  "Causeway"
          |  "Causway"
          |  "Causwa"
          ;
_ct       :  "Ct"
          |  "Court"
          |  "Crt"
          ;
_ctav     :  "Ctav"
          |  "Ctave"
          ;
```

```
_ctr      :  "Ctr"
          |  "Cen"
          |  "Cent"
          |  "Center"
          |  "Centr"
          |  "Centre"
          |  "Cnter"
          |  "Cntr"
          ;
_ctrs     :  "Ctrs"
          |  "Centers"
          ;
_cts      :  "Cts"
          |  "Courts"
          |  "Crts"
          ;
_curv     :  "Curv"
          |  "Curve"
          ;
_cut      :  "Cutoff"
          |  "Cut Off"
          |  "Cut"
          ;
_cv       :  "Cv"
          |  "Cove"
          ;
_cvs      :  "Cvs"
          |  "Coves"
          ;
_cyn      :  "Cyn"
          |  "Canyn"
          |  "Canyon"
          |  "Cn"
          |  "Cnyn"
          ;
_div      :  "Div"
          |  "Divide"
          |  "Dv"
          |  "Dvd"
          ;
_dl       :  "Dl"
          |  "Dale"
          ;
_dm       :  "Dm"
          |  "Dam"
          ;
```

```
_dr       :  "Dr"
          |  "Driv"
          |  "Drive"
          |  "Drives"
          |  "Drunit"
          |  "Drv"
          |  "Drve"
          ;
_drct     :  "Drct"
          |  "Drive" _ct
          |  "Dr" _ct
          ;
_drs      :  "Drs"
          |  "Drives"
          ;
_ent      :  "Ent"
          |  "Entrada"
          ;
_est      :  "Est"
          |  "Estate"
          ;
_ests     :  "Ests"
          |  "Estates"
          ;
_expy     :  "Expy"
          |  "Epwy"
          |  "Ex"
          |  "Exp"
          |  "Expr"
          |  "Express"
          |  "Expressway"
          |  "Expswy"
          |  "Expw"
          |  "Expwy"
          |  "Exwy"
          |  "Exy"
          |  "Xpwy"
          |  "Xpy"
          ;
_ext      :  "Ext"
          |  "Extended"
          |  "Extension"
          |  "Extn"
          |  "Extnsn"
          ;
_exts     :  "Exts"
          |  "Extensions"
          ;
```

```
_fall   :  "Fall"
        |  "Fll"
        ;
_fld    :  "Fld"
        |  "Field"
        ;
_flds   :  "Flds"
        |  "Fields"
        ;
_fls    :  "Fls"
        |  "Falls"
        ;
_flt    :  "Flt"
        |  "Flat"
        ;
_flts   :  "Flts"
        |  "Flats"
        ;
_frd    :  "Frd"
        |  "Ford"
        ;
_frds   :  "Frds"
        |  "Fords"
        ;
_frg    :  "Frg"
        |  "Forg"
        |  "Forge"
        ;
_frgs   :  "Frgs"
        |  "Forges"
        ;
_frk    :  "Frk"
        |  "Fork"
        ;
_frks   :  "Frks"
        |  "Forks"
        ;
_frst   :  "Forest"
        |  "Frst"
        |  "For"
        |  "Forrest"
        |  "Forests"
        ;
_fry    :  "Fry"
        |  "Ferry"
        |  "Frry"
        ;
```

```
_front  :  "Frontage"
        |  "Front"
        |  "Frnt"
        |  "Frntge"
        |  "Fr"
        ;
_frntrd :  _front _rd
        ;
_ft     :  "Ft"
        |  "Fort"
        |  "Frt"
        ;
_fwy    :  "Fwy"
        |  "Freeway"
        |  "Freewy"
        |  "Frway"
        |  "Frw"
        |  "Frwy"
        |  "Fw"
        |  "Fh"
        ;
_gdn    :  "Gdn"
        |  "Garden"
        |  "Gardn"
        |  "Grden"
        |  "Grdn"
        ;
_gdns   :  "Gdns"
        |  "Gardens"
        |  "Grdns"
        ;
_gln    :  "Gln"
        |  "Glen"
        ;
_glns   :  "Glns"
        |  "Glens"
        ;
_grn    :  "Grn"
        |  "Green"
        ;
_grns   :  "Grns"
        |  "Greens"
        ;
_grv    :  "Grv"
        |  "Grove"
        |  "Grov"
        ;
```

```
_grvs   :  "Grvs"
        |  "Groves"
        ;
_gtwy   :  "Gtwy"
        |  "Gateway"
        |  "Gatewy"
        |  "Gatway"
        ;
_hbr    :  "Hbr"
        |  "Harb"
        |  "Harbor"
        |  "Harbr"
        |  "Hrbor"
        ;
_hbrs   :  "Hbrs"
        |  "Harbors"
        ;
_hl     :  "Hl"
        |  "Hill"
        ;
_hls    :  "Hls"
        |  "Hills"
        ;
_holw   :  "Hollow"
        |  "Holw"
        |  "Hllw"
        |  "Hollows"
        |  "Holws"
        |  "Hlws"
        |  "Hlw"
        ;
_holws  :  "Holws"
        |  "Hollows"
        |  "Hlws"
        ;
_hts    :  "Hts"
        |  "Height"
        |  "Heights"
        |  "Hgts"
        |  "Hghts"
        |  "Hgt"
        |  "Ht"
        ;
_hvn    :  "Hvn"
        |  "Haven"
        |  "Havn"
        ;
```

```
_ushwy   :  "US Hwy"
         |  "US" _hwy
         |  "UShwy"
         |  "Ushy"
         ;
_sthwy   :  "Sthy"
         |  "Sthwy"
         |  _rte
         |  _state _hwy
         |  "State" _hwy
         |  "St" _hwy
         |  _state
         ;
_strte   :  "State Rte"
         |  _state _rte
         |  "State" _rte
         |  "St" _rte
         ;
_hwy     :  "Hwy"
         |  "Hghwy"
         |  "Hghway"
         |  "Hgwy"
         |  "Highw"
         |  "Highway"
         |  "Highways"
         |  "Highwy"
         |  "Hiway"
         |  "Hwys"
         |  "Hw"
         |  "Hway"
         |  "Hy"
         |  "Interstate"
         ;
_inlt    :  "Inlt"
         |  "Inlet"
         ;
_is      :  "Is"
         |  "Island"
         |  "Islnd"
         ;
_isle    :  "Isle"
         |  "Isles"
         ;
_iss     :  "Iss"
         |  "Islands"
         |  "Islnds"
         ;
```

```
_jct     :  "Jct"
         |  "Jction"
         |  "Jcton"
         |  "Jctn"
         |  "Jctns"
         |  "Jcts"
         |  "Junctn"
         |  "Juncton"
         |  "Junction"
         |  "Junctions"
         ;
_knl     :  "Knl"
         |  "Knol"
         |  "Knoll"
         ;
_knls    :  "Knls"
         |  "Knolls"
         ;
_ky      :  "Ky"
         |  "Key"
         ;
_kys     :  "Kys"
         |  "Keys"
         ;
_land    :  "Land"
         |  "Lnd"
         ;
_lck     :  "Lck"
         |  "Lock"
         ;
_lcks    :  "Lcks"
         |  "Locks"
         ;
_ldg     :  "Ldg"
         |  "Ldge"
         |  "Lodg"
         |  "Lodge"
         ;
_lf      :  "Lf"
         |  "Loaf"
         ;
_lgt     :  "Lgt"
         |  "Light"
         ;
_lgts    :  "Lgts"
         |  "Lights"
         ;
_lk      :  "Lk"
         |  "Lake"
         ;
```

```
_lks     :  "Lks"
         |  "Lakes"
         ;
_ln      :  "Ln"
         |  "Lane"
         |  "Lanes"
         |  "La"
         |  "Lne"
         ;
_lndg    :  "Lndg"
         |  "Landing"
         |  "Lndng"
         ;
_locks   :  "Locks"
         |  "Lcks"
         ;
_loop    :  "Loop"
         |  "Loops"
         |  "Lp"
         ;
_mall    :  "Mall"
         |  "Mll"
         ;
_mdw     :  "Mdw"
         |  "Meadow"
         ;
_mdws    :  "Mdws"
         |  "Meadows"
         |  "Medows"
         ;
_mews    :  "Mews"
         ;
_ml      :  "Ml"
         |  "Mill"
         ;
_mls     :  "Mls"
         |  "Mills"
         ;
_mnr     :  "Mnr"
         |  "Manor"
         ;
_mnrs    :  "Mnrs"
         |  "Manors"
         ;
_msn     :  "Msn"
         |  "Mission"
         |  "Missn"
         |  "Mssn"
         ;
```

```
_mt      :  "Mt"
         |  "Mnt"
         |  "Mount"
         ;
_mtn     :  "Mtn"
         |  "Mntain"
         |  "Mntn"
         |  "Mountain"
         |  "Mountin"
         |  "Mtin"
         ;
_mtns    :  "Mntns"
         |  "Mountains"
         ;
_mtwy    :  "Mtwy"
         |  "Motorway"
         ;
_nck     :  "Nck"
         |  "Neck"
         ;
_opas    :  "Opas"
         |  "Overpass"
         |  "Overps"
         |  "Ovps"
         |  "Ovrps"
         ;
_orch    :  "Orch"
         |  "Orchard"
         |  "Orchrd"
         ;
_oval    :  "Oval"
         |  "Ovl"
         ;
_ovlk    :  "Ovlk"
         |  "Overlook"
         ;
_park    :  "Park"
         |  "Parks"
         |  "Prk"
         |  "Pk"
         ;
_pass    :  "Pass"
         ;
_path    :  "Path"
         |  "Paths"
         |  "Pth"
         ;
_pike    :  "Pike"
         |  "Pikes"
         |  "Pke"
         ;
```

```
_pine    :  "Pine"
         |  "Pines"
         |  "Pne"
         ;
_pkwy    :  "Pkwy"
         |  "Parkway"
         |  "Parkwy"
         |  "Pkway"
         |  "Pkw"
         |  "Pky"
         |  "Prkway"
         |  "Prkwy"
         |  "Pw"
         |  "Pwy"
         |  "Py"
         |  "Parkways"
         |  "Pkwys"
         ;
_pl      :  "Pl"
         |  "Plac"
         |  "Place"
         |  "Plc"
         |  "Plce"
         ;
_pla     :  "Pla"
         |  "Placita"
         ;
_pln     :  "Pln"
         |  "Plain"
         ;
_plns    :  "Plns"
         |  "Plaines"
         |  "Plains"
         ;
_plz     :  "Plz"
         |  "Plaz"
         |  "Plaza"
         |  "Plza"
         |  "Pz"
         ;
_pnes    :  "Pnes"
         ;
_prr     :  "Prr"
         |  "Pr"
         |  "Prairie"
         |  "Prarie"
         ;
```

```
_prd     :  "Prd"
         |  "Rpd"
         ;
_prds    :  "Prds"
         |  "Rapids"
         |  "Rpds"
         ;
_prom    :  "Prom"
         |  "Promenade
         ;
_prt     :  "Prt"
         |  "Port"
         ;
_prts    :  "Prts"
         |  "Ports"
         ;
_psge    :  "Psge"
         |  "Passage"
         ;
_pso     :  "Pso"
         |  "Paseo"
         ;
_pt      :  "Pt"
         |  "Pnt"
         |  "Point"
         ;
_pts     :  "Pts"
         |  "Points"
         |  "Pt"
         ;
_radl    :  "Radl"
         |  "Radial"
         |  "Radiel"
         |  "Rad"
         ;
_ramp    :  "Ramp"
         |  "Rmp"
         ;
_rch     :  "Rch"
         |  "Rancho"
         |  "Ranch"
         |  "Ranches"
         |  "Ranchos"
         |  "Rnch"
         ;
_rd      :  "Rd"
         |  "Rdbox"
         |  "Rdrr"
         |  "Road"
         |  "R"
         ;
```

```
_rdgs   :  "Rdgs"
        |  "Ridges"
        ;
_rds    :  "Rds"
        |  "Roads"
        ;
_riv    :  "Riv"
        |  "River"
        |  "Rvr"
        |  "Rivr"
        ;
_rnch   :  "Rnch"
        |  "Ranch"
        |  "Ranches"
        |  "Rnchs"
        ;
_row    :  "Row"
        ;
_rpd    :  "Rpd"
        |  "Rapid"
        ;
_rst    :  "Rst"
        |  "Rest"
        ;
_rte    :  "Rte"
        |  "Rt"
        |  "Route"
        ;
_rue    :  "Rue"
        ;
_run    :  "Run"
        ;
_shl    :  "Shl"
        |  "Shoal"
        ;
_shls   :  "Shls"
        |  "Shoals"
        ;
_shr    :  "Shr"
        |  "Shoar"
        |  "Shore"
        ;
_shrs   :  "Shrs"
        |  "Shoars"
        |  "Shores"
        ;
_skwy   :  "Skwy"
        |  "Skway"
        |  "Skyway"
        ;
```

```
_smt    :  "Smt"
        |  "Sumit"
        |  "Sumitt"
        |  "Summit"
        ;
_spg    :  "Spg"
        |  "Spring"
        |  "Spng"
        |  "Sprng"
        ;
_spgs   :  "Spgs"
        |  "Springs"
        |  "Spngs"
        |  "Sprngs"
        ;
_spur   :  "Spur"
        |  "Spurs"
        ;
_sq     :  "Sq"
        |  "Sqr"
        |  "Sqre"
        |  "Squ"
        |  "Square"
        ;
_sqs    :  "Sqs"
        |  "Squares"
        |  "Suqares"
        |  "Sqrs"
        ;
_st     :  "St"
        |  "Stapt"
        |  "Stbox"
        |  "Stbx"
        |  "Steet"
        |  "Stlot"
        |  "Stpobox"
        |  "Str"
        |  "Strd"
        |  "Street"
        |  "Streetbox"
        |  "Strfd"
        |  "Strr"
        |  "Strt"
        ;
_stext  :  _st _ext
        ;
```

```
_sta    :  "Sta"
        |  "Station"
        |  "Statn"
        |  "Stn"
        ;
_stct   :  "Stct"
        |  "Strct"
        |  "Street" _ct
        |  "St" _ct
        |  "Str" _ct
        ;
_stdr   :  "Stdr"
        |  "Strdr"
        ;
_stln   :  "Stln"
        |  "Strln"
        ;
_stpl   :  "Stpl"
        |  "Strpl"
        ;
_stra   :  "Stra"
        |  "Strav"
        |  "Strave"
        |  "Straven"
        |  "Stravenue"
        |  "Stravn"
        |  "Strvn"
        |  "Strvnue"
        ;
_strm   :  "Strm"
        |  "Stream"
        |  "Streme"
        ;
_sts    :  "Sts"
        |  "Streets"
        ;
_ter    :  "Ter"
        |  "Te"
        |  "Terr"
        |  "Terrace"
        |  "Terrce"
        ;
_tpke   :  "Tpke"
        |  "Tnpk"
        |  "Tpk"
        |  "Trnpk"
        |  "Trpk"
        |  "Turn"
        |  "Turnpike"
        |  "Turnpk"
        ;
```

```
_trak     :  "Trak"
          |  "Track"
          |  "Tracks"
          |  "Trk"
          |  "Trks"
          ;
_trce     :  "Trce"
          |  "Trace"
          |  "Traces"
          ;
_trfy     :  "Trfy"
          |  "Trafficway"
          ;
_trl      :  "Trl"
          |  "Tl"
          |  "Tr"
          |  "Trail"
          |  "Trails"
          |  "Trls"
          ;
_trlr     :  "Trlr"
          |  "Trailer"
          |  "Trlrs"
          ;
_trvce    :  "Transverse"
          |  "trans"
          |  "trnsvrce"
          ;
_trwy     :  "Trwy"
          |  "Throughway"
          |  "Thruway"
          |  "Thruwy"
          ;
_tunl     :  "Tunl"
          |  "Tunel"
          |  "Tunls"
          |  "Tunnel"
          |  "Tunnels"
          |  "Tunnl"
          ;
_un       :  "Un"
          |  "Union"
          ;
_uns      :  "Uns"
          |  "Unions"
          ;
_upas     :  "Upas"
          |  "Underpass"
          ;
```

```
_ver      :  "Ver"
          |  "Vereda"
          ;
_via      :  "Via"
          |  "Vdct"
          |  "Viadct"
          |  "Viaduct"
          ;
_vis      :  "Vis"
          |  "Vist"
          |  "Vista"
          |  "Vst"
          |  "Vsta"
          ;
_vl       :  "Vl"
          |  "Ville"
          ;
_vlg      :  "Vlg"
          |  "Vill"
          |  "Villag"
          |  "Village"
          |  "Villg"
          |  "Villiage"
          ;
_vlgs     :  "Vlgs"
          |  "Villages"
          ;
_vly      :  "Vly"
          |  "Valley"
          |  "Vally"
          |  "Vlly"
          ;
_vlys     :  "Vlys"
          |  "Valleys"
          ;
_vw       :  "Vw"
          |  "View"
          ;
_vws      :  "Vws"
          |  "Views"
          ;
_walk     :  "Walk"
          |  "Walks"
          |  "Wk"
          ;
_wall     :  "Wall"
          |  "Wll"
          ;
```

```
_way      :  "Way"
          |  "Wy"
          ;
_ways     :  "Ways"
          |  "Wys"
          ;
_wl       :  "Wl"
          |  "Well"
          ;
_wls      :  "Wls"
          |  "Wells"
          ;
_xing     :  "Xing"
          |  "Crossing"
          |  "Crssing"
          |  "Crsg"
          |  "Crssng"
          ;
_xrd      :  "Xrd"
          |  "Crossroad"
          |  "Crossrd"
          ;
_xrds     :  "Xrds"
          |  "Crossroads"
          |  "Crossrds"
          ;
suf_type  :  _acrd
          |  _aly
          |  _anx
          |  _arc
          |  _avct
          |  _avdr
          |  _ave
          |  _avenida
          |  _avln
          |  _avpl
          |  _bch
          |  _bg
          |  _bgs
          |  _blf
          |  _blfs
          |  _blvd
          |  _bnd
          |  _br
          |  _brg
          |  _brk
          |  _brks
          |  _btm
          |  _bus
          |  _byp
```

```
| _byu
| _cam
| _cer
| _cir
| _cirs
| _clb
| _clf
| _clfs
| _cll
| _cmn
| _cmns
| _cmt
| _cor
| _cord
| _cohwy
| _cohwyne
| _cohwynw
| _cohwyse
| _cohwysw
| _corte
| _cors
| _cp
| _cpe
| _cres
| _crk
| _crse
| _crst
| _cswy
| _ct
| _ctav
| _ctr
| _ctrs
| _cts
| _curv
| _cut
| _cv
| _cvs
| _cyn
| _div
| _dl
| _dm
| _dr
| _drct
| _drs
| _ent
| _est
| _ests
| _expy
| _ext
```

```
| _exts
| _fall
| _fld
| _flds
| _fls
| _flt
| _flts
| _frd
| _frds
| _frg
| _frgs
| _frk
| _frks
| _frst
| _fry
| _front
| _frntrd
| _ft
| _fwy
| _gdn
| _gdns
| _gln
| _glns
| _grn
| _grns
| _grv
| _grvs
| _gtwy
| _hbr
| _hbrs
| _hl
| _hls
| _holw
| _holws
| _hts
| _hvn
| _hwy
| _inlt
| _is
| _isle
| _iss
| _jct
| _knl
| _knls
| _ky
| _kys
| _land
| _lck
| _lcks
| _ldg
```

```
| _lf
| _lgt
| _lgts
| _lk
| _lks
| _ln
| _lndg
| _locks
| _loop
| _mall
| _mdw
| _mdws
| _mews
| _ml
| _mls
| _mnr
| _mnrs
| _msn
| _mt
| _mtn
| _mtns
| _mtwy
| _nck
| _opas
| _orch
| _oval
| _ovlk
| _park
| _pass
| _path
| _pike
| _pine
| _pkwy
| _pl
| _pla
| _pln
| _plns
| _plz
| _pnes
| _prr
| _prd
| _prds
| _prom
| _prt
| _prts
| _psge
| _pso
| _pt
| _pts
| _radl
```

```
| _ramp
| _rch
| _rd
| _rdg
| _rdgs
| _rds
| _riv
| _rnch
| _row
| _rpd
| _rst
| _rte
| _rue
| _run
| _shl
| _shls
| _shr
| _shrs
| _skwy
| _smt
| _spg
| _spgs
| _spur
| _sq
| _sqs
| _st
| _stext
| _sta
| _stct
| _stdr
| _stln
| _stpl
| _stra
| _strm
| _sts
| _ter
| _tpke
| _trak
| _trce
| _trfy
| _trl
| _trlr
| _trvce
| _trwy
| _tunl
| _un
| _uns
| _upas
| _ushwy
```

```
| _ver
| _via
| _vis
| _vl
| _vlg
| _vlgs
| _vly
| _vlys
| _vw
| _vws
| _walk
| _wall
| _way
| _ways
| _wl
| _wls
| _xing
| _xrd
| _xrds
;
```

Note that for prefix and suffix types, the Standardize Addresses geoprocessing tool will output the first value among the alternates for the output standardized value. For example, the address "601 CALLE PALO COLORADO" will be standardized with "CLL" as the prefix type.

Unit names Alias lists are defined for variations of unit types to be recognized.

```
[-] Unit names
_acres      :  "acres"
            |  "acr"
            |  "acre"
            |  "acrs"
            ;
_apt        :  "apt"
            |  "apartments"
            |  "apts"
            |  "apartment"
            |  "aprtmnt"
            ;
_box        :  "box"
            |  "bx"
            ;
_bldg       :  "bldg"
            |  "bld"
            |  "bldg"
            |  "building"
            |  "blg"
            |  "room"
            ;
_church     :  "church"
            |  "chr"
            |  "ch"
            ;
_coll       :  "coll"
            |  "college"
            ;
_condo      :  "condo"
            |  "condominiums"
            |  "condos"
            ;
_coop       :  "coop"
            ;
_dorm       :  "dorm"
            |  "dormitory"
            ;
_dept       :  "dept"
            |  "department"
            ;
_hngr       :  "hngr"
            |  "hangar"
            ;
_home       :  "home"
            |  "homes"
            |  "house"
            |  "hse"
            ;
_hotel      :  "hotel"
            |  "htl"
            ;
_house      :  "house"
            |  "hse"
            ;
_inn        :  "inn"
            |  "motel"
            ;
_lbby       :  "lbby"
            |  "lobby"
            ;
_lowr       :  "lowr"
            |  "lower"
            ;
_ofc        :  "ofc"
            |  "office"
            ;
_motel      :  "motel"
            |  "mtl"
            ;
_pier       :  "pier"
            |  "pier"
            ;
_plnt       :  "plnt"
            |  "plantation"
            ;
_school     :  "school"
            |  "sch"
            ;
_sem        :  "sem"
            |  "seminary"
            ;
_shctr      :  "shctr"
            |  "shopping"
            |  "shoppinngcenter"
            |  "shoppinng center"
            |  "shoppingctr"
            |  "shopping ctr"
            ;
_side       :  "side"
            ;
_slip       :  "slip"
            ;
_stop       :  "stop"
            ;
```

```
_tower      :  "tower"
            |  "twr"
            |  "twrs"
            |  "towers"
            ;
_univ       :  "univ"
            |  "university"
            |  "unv"
            |  "unvrsty"
            ;
_whs        :  "whs"
            |  "ware house"
            |  "warehse"
            ;
_bsmt       :  "bsmt"
            |  "bsmnt"
            |  "basement"
            ;
_unit       :  "unit"
            |  "units"
            ;
_uppr       :  "uppr"
            |  "upper"
            ;
_floor      :  "fl"
            |  "flr"
            |  "floor"
            ;
_ste        :  "ste"
            |  "suite"
            ;
_entry      :  "entry"
            ;
_lot        :  "lot"
            ;
_ms         :  "ms"
            ;
_pnth       :  "ph"
            |  "pnth"
            |  "penthouse"
            |  "penthse"
            |  "pent house"
            ;
_frnt       :  "frnt"
            |  "front"
            ;
_rear       :  "rear"
            ;
```

```
_room       :  "room"
            |  "rm"
            ;
_sp         :  "spc"
            |  "sp"
            |  "space"
            ;
_stall      :  "stall"
            |  "stll"
            ;
```

Now, the alias lists are used to define the HousingUnitType element.

```
HousingUnitType :  _acres
                 |  _apt
                 |  _box
                 |  _bldg
                 |  _church
                 |  _coll
                 |  _condo
                 |  _coop
                 |  _dept
                 |  _dorm
                 |  _hngr
                 |  _home
                 |  _hotel
                 |  _house
                 |  _inn
                 |  _lbby
                 |  _lowr
                 |  _ofc
                 |  _motel
                 |  _pier
                 |  _plnt
                 |  _school
                 |  _sem
                 |  _shctr
                 |  _side
                 |  _slip
                 |  _stop
                 |  _tower
                 |  _trlr
                 |  _univ
                 |  _whs
                 |  _bsmt
                 |  _unit
                 |  _uppr
                 |  _floor
                 |  _ste
                 |  _entry
                 |  _lot
                 |  _ms
                 |  _pnth
                 |  _frnt
                 |  _rear
                 |  _room
                 |  _sp
                 |  _stall
                 |  "pmb"
                 ;
```

Multiline input

```
[-] Multiline input
$Multiline_input :  [Input:ZIP][Input:City][Input:State][Input:Street]FullNormalAddress
                    { ['#addressSearch'] }
                 |  [Input:ZIP][Input:City][Input:State][Input:Street]FullIntersection
                    { @intersection( ['#leftStreet'] , ['#rightStreet'] ) }
                 |  [Input:Street]MultiLineAddress70 multilineZone30
                    { ['#addressSearch'] }
                 |  [Input:Street]FullStreetName40 intConnector FullStreetName40 multilineZone30
                    { @intersection( ['#leftStreet'] , ['#rightStreet'] ) }
                 ;
multilineZone    :  [Input:ZIP]GenZIP100 [Input:City]OptCityNoSearch20 [Input:State]OptStateNoSearch20
                 |  [Input:City]City40 [Input:State]OptState20 [Input:ZIP]OptZipNoSearch20
                 ;
```

These definitions are used in batch geocoding. Multiline input supports using addresses concatenated from what would usually be multiple fields into a single input string. The first two cases for Multiline_input above indicate that where the input zone fields (ZIP, City, and State) are not supplied in the tool dialog, the input field for the Street component is parsed according to the FullNormalAddress grammar definition. To interpret the syntax shown (where <empty> means "is empty"), read it as

```
"[Input ZIP] <empty> [Input City] <empty> [Input State]
<empty> [input Street] is FullNormalAddress"
```

If no plausible candidate is found and the style supports intersections, read it as

```
"[Input ZIP] <empty> [Input City] <empty> [Input State]
<empty> [input Street] is FullIntersection"
```

If no plausible candidates are found yet, it is

```
"[Input Street] is MultiLineAddress + multilineZone"
```

Finally, if no plausible candidate is found and the style supports intersections, then

```
"[input Street] is FullStreetName + intConnector +
FullStreetName + multilineZone"
```

This cascading search is termed *fallback* searching.

Spelling

This section defines two types of character substitution for word elements not distinguishable in the browser view but exposed in the XML. Some substitutions are by character group, and some are by individual character. Noise characters are not currently implemented. An example of a group substitution would be Fischer → Fisher, where "sch" is substituted with "sh" or vice versa. Each entry in the Spelling section may be unidirectional or bidirectional, as indicated by the → or ↔ connectors.

The default section is where you can specify a set of known or expected substitutions for typographic or phonetically based errors useful for dictionary search in the locator index.

[−] **Spelling**

[−] **default**

From		To	SpellSens level	Cost
sch	→	sh	90	0.1
sh	→	sch	90	0.1
mi	→	my	90	0.1
my	→	mi	90	0.1
yn	→	in	90	0.1
in	→	yn	90	0.1
ph	→	f	90	0.1
BFPVbfpv	↔	BFPVbfpv	90	0.1
Cc	↔	GKSgksÇçĆćČčĞğĢĝĶķŚśŞşŠš	90	0.1
Gg	↔	CJcjÇçĆćČčĞğĢĝ	90	0.1
Hh	↔	ħ	90	0.1
Jj	↔	GgğĞĢĝ	90	0.1
Kk	↔	CcÇçĆćČčĶķ	90	0.1
Qq	↔	Kk	90	0.1
Ss	↔	CcÇçĆćČčŚśŞşŠš	90	0.1
Xx	↔	HhZz	90	0.1
Zz	↔	ŹźŻżŽž	90	0.1
DTdt	↔	ĎďĐđŢţŤť	90	0.1
Ll	↔	1ĻļĽłŁł	90	0.1
MNmn	↔	ÑńńŅņŇň	90	0.1
Rr	↔	ŕŘř	90	0.1
Ww	↔	Vv	90	0.1

Noise chars:

[−] **scoring**

From		To	SpellSens level	Cost
sch	→	sh	90	0.1
sh	→	sch	90	0.1
mi	→	my	90	0.1
my	→	mi	90	0.1
yn	→	in	90	0.1
in	→	yn	90	0.1
ph	→	f	90	0.1
AOao	↔	ÀÁÄÅÓÔÕÖØàáâãäåðòóôõöøĀāăĄąŐő	90	0.1
EIei	↔	eiÈÉÉèéêëìíîïĒēĖėęěĨĩĮıĖë	90	0.1
Uu	↔	AIaiÀÁÄÅÍÚÜàáâãäåìíîïùúûüĀāăĄąĨĩĮıŪūŮűŲ	90	0.1
Yy	↔	IıÍìíîïýÿĨĩĮıı	90	0.1
BFPVbfpv	↔	BFPVbfpv	90	0.1
Cc	↔	GKSgksÇçĆćČčĞğĢģĶķŚśŞşŠš	90	0.1
Gg	↔	CJcjÇçĆćČčĞğĢģ	90	0.1
Hh	↔	ħ	90	0.1
Jj	↔	GgĞğĢģ	90	0.1
Kk	↔	CcÇçĆćČčĶķ	90	0.1
Qq	↔	Kk	90	0.1
Ss	↔	CScsÇçĆćČčŚśŞşŠš	90	0.1
Xx	↔	HhZz	90	0.1
Zz	↔	ŹźŻżŽž	90	0.1
DTdt	↔	ĎďĐđŢţŤť	90	0.1
Ll	↔	1ĻļĽľŁł	90	0.1
MNmn	↔	ÑñńŅņŇň	90	0.1
Rr	↔	ŕŘř	90	0.1
Ww	↔	Vv	90	0.1

Noise chars:

The scoring section repeats the default definitions and adds vowels and vowels with diacritics; vowels are not used in searching but are considered for scoring. In both sections, the substitutions only apply if the runtime parameter for spelling sensitivity has been set to less than 90 (or another value). If it is set to 90 (or another value) or above, then substitutions are not made when searching for candidates.

The score cost penalty is subtracted from the weight of the element. For the example given, supposing the weight for the street name element is 6 in the grammar and a match required a substitution of SCH for SH to geocode an input of "123 Fischer Parade" to a reference value combination of "123 Fisher Parade," then the street name element score contribution would be 6 - 0.1 = 5.9.

You can use the SpellSens level and Cost parameters to tune the behavior of a locator when handling common transliterations and typographic mistakes, like keying the number 0 instead of the letter O (this particular case is not in the example).

(Locator) Mapping Schemas

This section defines how reference data logically relates to grammar elements—the "logical model." It also specifies the dictionary for indexing and relationships of the address components, although this cannot be seen in the browser view. If we were to diagram the attribute and grammar relationships, we would see a simple 1:1 correspondence between reference data fields and grammar element definitions. For the Fields elements, the bolded names are used as variables by the engine. Comments are in gray.

We will inspect **SingleAddress**:

```
SingleAddress
Description:                 Single house addresses (points)
Geometry type:               point
Fields:                      [−]
                             Shape      Shape field
                             No preferred names defined
                             ID         Unique ID field
                             No preferred names defined
                             House      House number
                             Maps to grammar element House
                             Names:     [+]
                             PreDir     Prefix Direction
                             Maps to grammar element prefix
                             Names:     [+]
                             PreType    Prefix Type
                             Maps to grammar element pretype
                             Names:     [+]
                             StreetNameStreet Name
                             Maps to grammar element StName
                             Names:     [+]
                             SufType    Suffix Type
                             Maps to grammar element suftype
                             Names:     [+]
                             SufDir     Suffix Direction
                             Maps to grammar element suffix
                             Names:     [+]
                             City       City or Place
                             Maps to grammar element City
                             Names:     [+]
                             ZIP        Suffix Direction
                             Maps to grammar element ZIP5
                             Names:     [+]
                             State      State
                             Maps to grammar element State
                             Names:     [+]
                             Rank       Specifies rank (primary or secondary) for alternate records of the same geometry
                             Names:     [+]
                             User_fld   Additional field
                             No preferred names defined
Filter criteria (default):   $StreetName <> '' AND $StreetName <> ' ' AND UPPER($StreetName) <> 'UNNAMED' AND
                             UPPER($StreetName) <> 'UNNAMED STREET'
Filter criteria (MSAccess):  $StreetName <> '' AND $StreetName <> ' ' AND $StreetName <> 'UNNAMED' AND $StreetName
                             <> 'UNNAMED STREET'
Filter criteria (Oracle):    $StreetName IS NOT NULL AND $StreetName <> ' ' AND UPPER($StreetName) <> 'UNNAMED'
                             AND UPPER($StreetName) <> 'UNNAMED STREET'
```

We see that SingleAddress requires point geometry and can accept ID, House, PreDir, PreType, StreetName, SufType, SufDir, City, ZIP, State, Rank, and User_fld attributes in the reference data. Information about whether a field is required is not exposed in the browser view but is defined in the XML. Rank is for prioritizing separate records of the same geometry (assigning a preference for returning reference data attributes, for example, to return "Highway 123" from an input like "Main Street"). User_fld supports creating a locator with an additional output field taken from the reference data, for example, something you calculate into your reference data through a geoprocessing preprocess.

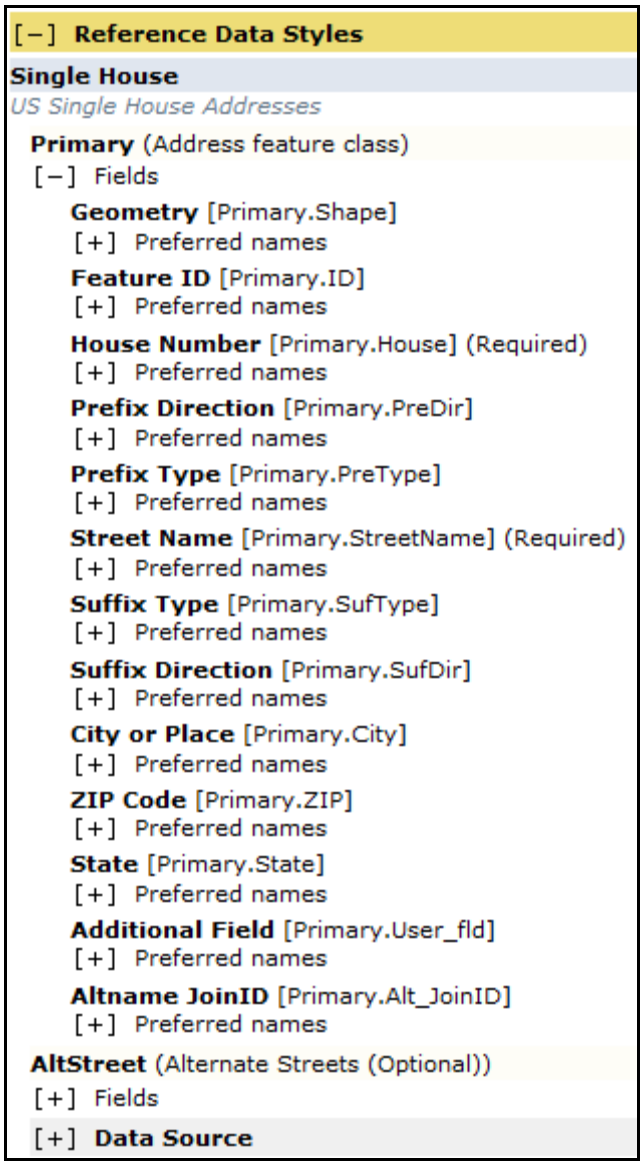
The Names element is not currently used and may be removed from the Mapping Schema definitions at a later release.

Filter criteria are SQL statements using the invariably present element StreetName to make sure reference data with no street names is not used for SingleAddress geocoding. Reference data commonly contains records not useful for geocoding but included for cartographic or network connectivity reasons.

(Locator) Reference Data Styles

This section defines the physical data model for reference data—table role, field role, and preferred field names to be automatically recognized when creating locators using the Create Address Locator geoprocessing tool. Data typing is not enforced by styles, except for geometry fields.

Here, we can see the Single House data style partially expanded:



Expanding one of the Primary role table fields, we can see the preferred field name list:

City or Place [Primary.City]
[−] Preferred names
PLACE
GEONAME
PLACENAME
CITY

Partially expanding the optional AltStreet and Data Source elements, we see the following:

AltStreet (Alternate Streets (Optional))
[−] Fields
JoinID [AltStreet.ID] (Required)
[+] Preferred names
Prefix Direction [AltStreet.PreDir]
[+] Preferred names
Prefix Type [AltStreet.PreType]
[+] Preferred names
Street Name [AltStreet.StreetName]
[+] Preferred names
Suffix Type [AltStreet.SufType]
[+] Preferred names
Suffix Direction [AltStreet.SufDir]
[+] Preferred names
[−] **Data Source**
Uses schema SingleAddressSearch
[−] **Query #1**
Tables: $Primary
Field mappings:

ShapePtPoly	$Primary.Shape
ID	$Primary.ID
House	$Primary.House
PreDir	$Primary.PreDir
PreType	$Primary.PreType
StreetName	$Primary.StreetName
SufType	$Primary.SufType
SufDir	$Primary.SufDir
City	$Primary.City
ZIP	$Primary.ZIP
State	$Primary.State
User_fld	$Primary.User_fld
Alt_JoinID	$Primary.Alt_JoinID

[+] **Query #2**

This shows how a 1:M join is implemented to a table with fields for FullStreetName grammar elements. The graphic above shows how the primary search query utilizes fields from the primary role table, while the graphic below shows a fallback query using the join to the alternate name role table and alternate FullStreetName mappings.

The Data Source reference to the SingleAddressSearch schema refers back to the Mapping Schemas section, where it ultimately links to the SingleAddress schema variable names for House, PreDir, PreType, StreetName, and so on.

```
[−] Query #2
Tables:          $Primary
                 $AltStreet
Field mappings:  ShapePtPoly  $Primary.Shape
                 ID           $Primary.ID
                 House        $Primary.House
                 PreDir       $AltStreet.PreDir
                 PreType      $AltStreet.PreType
                 StreetName   $AltStreet.StreetName
                 SufType      $AltStreet.SufType
                 SufDir       $AltStreet.SufDir
                 City         $Primary.City
                 ZIP          $Primary.ZIP
                 State        $Primary.State
                 User_fld     $Primary.User_fld
Join clause:     $Primary.Alt_JoinID = $AltStreet.ID
```

If there are any errors in the elements used in Mapping Schemas and Reference Data Styles to map data sources, schema fields, and grammar, then your style file may be valid XML but invalid to the geocoding engine. Mapping Schemas defines field names (e.g., "StreetName") that are used in Reference Data Styles. Take particular care when editing so that

- Mapping Schema **grammar_ref** attributes exactly match grammar element names.
- Mapping Schema index **dictionary** entries include all schema fields.
- Mapping Schema **relationships** include street name and all zone combinations.
- Mapping Schema **reverse relationships** use all schema fields.
- Mapping Schema **outputs** include all schema fields.
- Mapping Schema **reverse_geocoding** uses all schema fields.
- Mapping Schema **standardization** includes all schema fields.
- Reference Data **table roles** include a field role for all schema fields.
- Reference Data **field roles** use the field names defined in Mapping Schemas.
- Reference Data **data source** maps source names to all field roles.

The **standardization** section of Mapping Schemas has **build** and **tool** sections. The build section specifies how street names are parsed into the locator index when you create a locator. The tool section controls how the Standardize Addresses tool operates. Note that, by default, the build section does not convert parsed street name parts to standard values, but the tool does.

Output Formats

A section in the style file that is not currently rendered in a browser is the one for output formats; the XPath is **locators/locator/output_formats**. This defines how geocodes are reported by the engine when geocoding or reverse geocoding. Here is the collapsed view in Notepad++:

```
<!--This section defines the output address formats and the order of address
elements for normal addresses, intersections, and reverse geocoded addresses.-->

<output_formats>
  <format_definition name="format_score_per_component">
  <format_definition name="format_intersections">
  <format_definition name="format_intersection_name">
  <format_definition name="format_reverse_range_address">
  <format_definition name="format_normal_address">
</output_formats>
```

You may have noticed **format_ref** tags elsewhere in the style file, for example, the definition for FullNormalAddress top-level element has a result property that specifies the built-in search function to use and how to format the candidates—with **format_normal_address**.

```
<def name="FullNormalAddress">
  <alt>
    <elt ref="NormalAddress" weight="70"/>
    <elt ref="OptSeparator" weight="0"/>
    <elt ref="Zone" weight="15"/>
    <result tag="Address">
      <search_value ref="addressSearch"/>
      <format_ref ref="format_normal_address"/>
    </result>
  </alt>
</def>
```

Expanding **format_score_per_component,** we see the following:

```
<format_definition name="format_score_per_component">
  <value>;</value>
  <component_value component="component_score.House"   pre_separator="House="/>
  <component_value component="component_score.prefix"  pre_separator="; prefix=" />
  <component_value component="component_score.pretype" pre_separator="; pretype="/>
  <component_value component="component_score.StName"  pre_separator="; StName="/>
  <component_value component="component_score.suftype" pre_separator="; suftype="/>
  <component_value component="component_score.suffix"  pre_separator="; suffix="/>
  <component_value component="component_score.City"    pre_separator="; City="/>
  <component_value component="component_score.State"   pre_separator="; State="/>
  <component_value component="component_score.ZIP5"    pre_separator="; ZIP="/>
  <component_value component="component_score.prefix"  pre_separator="prefix1="    record="1"/>
  <component_value component="component_score.pretype" pre_separator="; pretype1=" record="1"/>
  <component_value component="component_score.StName"  pre_separator="; StName1="  record="1"/>
  <component_value component="component_score.suftype" pre_separator="; suftype1=" record="1"/>
  <component_value component="component_score.suffix"  pre_separator="; suffix1="  record="1"/>
  <component_value component="component_score.City"    pre_separator="; City1="    record="1"/>
  <component_value component="component_score.State"   pre_separator="; State1="   record="1"/>
  <component_value component="component_score.ZIP5"    pre_separator="; ZIP1="     record="1"/>
  <component_value component="component_score.prefix"  pre_separator="; prefix2="  record="2"/>
  <component_value component="component_score.pretype" pre_separator="; pretype2=" record="2"/>
  <component_value component="component_score.StName"  pre_separator="; StName2="  record="2"/>
  <component_value component="component_score.suftype" pre_separator="; suftype2=" record="2"/>
  <component_value component="component_score.suffix"  pre_separator="; suffix2="  record="2"/>
  <component_value component="component_score.City"    pre_separator="; City2="    record="2"/>
  <component_value component="component_score.State"   pre_separator="; State2="   record="2"/>
  <component_value component="component_score.ZIP5"    pre_separator="; ZIP2="     record="2"/>
</format_definition>
```

The **format_score_per_component** section controls the behavior of the **reportScorePerComponent** property when its value is "true." The entries with a record attribute deal with tied match scores; the ties should have differing values among the elements with record attributes.

The **format_intersections** section controls how intersection geocodes are reported:

```
<format_definition name="format_intersections">
  <field_value ref="PreDir" record="1"/>
  <field_value ref="PreType" record="1" pre_separator=" "/>
  <field_value ref="StreetName" record="1" pre_separator=" "/>
  <field_value ref="SufType" record="1" pre_separator=" "/>
  <field_value ref="SufDir" record="1" pre_separator=" "/>
  <value xml:space="preserve"> &amp;</value>
  <field_value ref="PreDir" record="2" pre_separator=" "/>
  <field_value ref="PreType" record="2" pre_separator=" "/>
  <field_value ref="StreetName" record="2" pre_separator=" "/>
  <field_value ref="SufType" record="2" pre_separator=" "/>
  <field_value ref="SufDir" record="2" pre_separator=" "/>
  <field_value ref="City" record="1" pre_separator=", "/>
  <field_value ref="State" record="1" pre_separator=", "/>
  <field_value ref="ZIP" record="1" pre_separator=" "/>
</format_definition>
```

The record attribute here pertains to the two records participating in the intersection geocode.

The **format_intersections_name** section is similar to the above except that it omits Zone fields:

```
<format_definition name="format_intersection_name">
  <field_value ref="PreDir" record="1"/>
  <field_value ref="PreType" record="1" pre_separator=" "/>
  <field_value ref="StreetName" record="1" pre_separator=" "/>
  <field_value ref="SufType" record="1" pre_separator=" "/>
  <field_value ref="SufDir" record="1" pre_separator=" "/>
  <value xml:space="preserve"> &amp;</value>
  <field_value ref="PreDir" record="2" pre_separator=" "/>
  <field_value ref="PreType" record="2" pre_separator=" "/>
  <field_value ref="StreetName" record="2" pre_separator=" "/>
  <field_value ref="SufType" record="2" pre_separator=" "/>
  <field_value ref="SufDir" record="2" pre_separator=" "/>
</format_definition>
```

Intersection addresses are commonly given without Zone values.

The next format section is **format_reverse_range_address**. This section controls how the text for reverse geocoding results is generated. The reverse geocoding plug-in returns a candidate object that has a calculated property set with house number, geometry, percent along, etc., and a reference data record property set for the candidate. By using record="1" in the formatting, it tells the engine when to use the original reference data property set values and not look in the calculated property set. If you have a different set of top-level elements in your style file, the names of your field references will be different.

```
<format_definition name="format_reverse_range_address">
  <component_value component="House" />
  <field_value ref="PreDir" pre_separator=" " record="1" />
  <field_value ref="PreType" pre_separator=" " record="1" />
  <field_value ref="StreetName" pre_separator=" " record="1" />
  <field_value ref="SufType" pre_separator=" " record="1" />
  <field_value ref="SufDir" pre_separator=" " record="1" />
</format_definition>
```

The last format section is **format_normal_address**. This controls how the text for normal geocoding candidates is generated. If you have a different set of top-level elements and zones for a normal address, your definition will be different.

```
<format_definition name="format_normal_address">
  <field_value ref="House" nulls="0"/>
  <field_value ref="PreDir" pre_separator=" "/>
  <field_value ref="PreType" pre_separator=" "/>
  <field_value ref="StreetName" pre_separator=" "/>
  <field_value ref="SufType" pre_separator=" "/>
  <field_value ref="SufDir" pre_separator=" "/>
  <field_value ref="City" pre_separator=", "/>
  <field_value ref="State" pre_separator=", "/>
  <field_value ref="ZIP" pre_separator=" "/>
</format_definition>
```

(Locator) Plugins

For completeness, an expanded view of the Plugins section is shown below. Customization with plug-ins will not be described in this document.

[–] Plugins

intersection
Description:
Type: Built-in

directed_offset
Description: *Calculates location given bearing and location.*
Parameters:
distance *(as double),*
units *(as esriUnits integer value),*
bearing *(decimal degrees angle, clockwise),*
location *(some location as IPropertySet)*
Type: Built-in

create_point
Description:
Type: Built-in

calculate_score
Description:
Type: Built-in

CalcZipRange
Description:
Type: Built-in

match_house_to_range
Description:
Type: Built-in

reverse_geocode_range
Description:
Type: Built-in

calc_polygon_centroid
Description: *Expects one parameter as a IGeometry object (polygon). Returns IGeometry (point) as a centroid of the passed polygon.*
Type: Built-in

no_op
Description: *No operation (does nothing) - returns value of the first parameter "as is"; other parameters ignored.*
Type: Built-in

concatenate
Description: *Converts all parameters to strings and returns concatenation of them.*
Type: Built-in

interpolate
Description: *Calculates address location along a polyline.*
Type: Built-in

reverse_geocode_intersections
Description: *Reverse geocode street intersections*
Type: Built-in

reverse_MGRS
Description: *Reverse geocode location and return result as MGRS string*
Type: Built-in

Appendix A: Example of Editing Locator Properties

Here is a table of locator properties that may be used in a style:

Category	Property Name	Description and Purpose	Type	Default Value if the Property Is Not Specified	Remarks
General Options	MinimumMatchScore	The minimum match score that determines if an input address is matched against the candidate in batch geocoding	Float	90	
	MinimumCandidateScore	The minimum score whereby the candidates will be displayed in interactive matching	Float	10	
	SpellingSensitivity	Spelling sensitivity	Float	80	
	MatchIfScoresTie	Automatically matches to the first best candidate if the scores are tied	Boolean	TRUE	
	SideOffset	The value that determines how far away from the left or right side of a line feature an address location should be placed	Integer	20	
	SideOffsetUnits	The units used for Side Offset	esriUnits	Feet	
	EndOffset	The value that determines how far away from the end of a line an address location should be placed	Integer	3	Percentage of line segment length
	EndOffsetUnits	The units used for End Offset	esriUnits	Percent	
	UseRelativePaths	Stores relative paths to the reference data in the locator	Boolean	TRUE	
Matching Options	supportsIntersections	Enables support for intersection matching if the reference features are centerlines	Boolean	TRUE	
	IntersectionConnectors	Intersection connectors for delimiting two streets in intersection matching	String	&	Requires that supportIntersections=TRUE

Category	Property Name	Description and Purpose	Type	Default Value if the Property Is Not Specified	Remarks
	supportsEmptyHouseNumber	Enables matching address without house number if the supporting grammar is specified	Boolean	FALSE	Set to false for improving performance
	Interpolate.SideValue.Left	The value in the reference data mapped as Left side in a Nickel data schema	String	L	
	Interpolate.SideValue.Right	The value in the reference data mapped as Right side in a Nickel data schema	String	R	
Outputs	WriteXYCoordFields	Writes x and y coordinate fields in the result feature class	Boolean	FALSE	Batch match
	WriteReferenceIDField	Writes Reference data ID field in the result feature class	Boolean	FALSE	Batch match
	WritePercentAlongField	Writes Percent Along field in the result feature class	Boolean	FALSE	Batch match
	WriteAdditionalOutputFields	Writes Additional Output fields in the result feature class	Boolean	TRUE	Batch match
	ShowElapsedTime	Adds a field showing the elapsed time per each record in the result feature class	Boolean	FALSE	Batch match
	reportScorePerComponent	Adds a field showing the score per component in the result feature class	Boolean	FALSE	Batch match
Performance	RuntimeMemoryLimit	Sets Runtime Memory Limit (byte) for optimizing batch matching performance	Integer	512000000	Bytes
	BatchPresortInputs	Presorts input table by fields for improving matching performance		No default value	Batch match
	<State>	<field name>			Property list
	<ZIP>	<field name>			Property list
	<City>	<field name>			Property list
	BatchPresortCacheSize	Presorts Cache Size (number of records)	Integer	50000	Batch match

J-9969

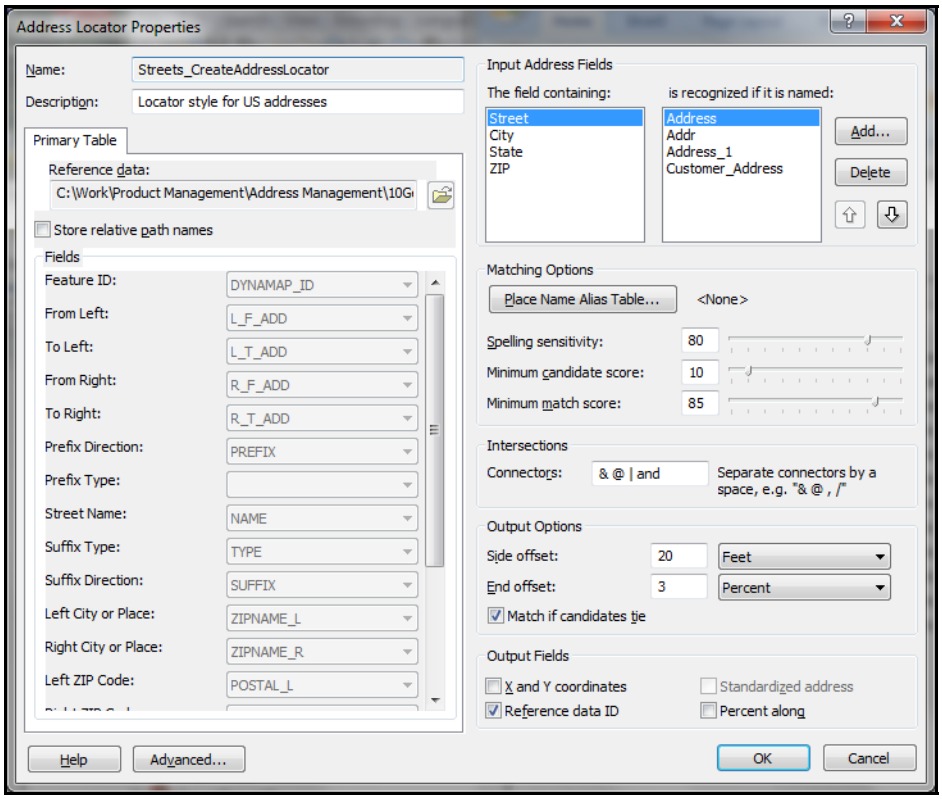
Category	Property Name	Description and Purpose	Type	Default Value if the Property Is Not Specified	Remarks
	MaxCandidates	Maximum number of candidates to be searched for in interactive matching; used to optimize performance for interactive matching	Integer	200	Interactive match
	MaxPerfectCandidates	Maximum number of candidates with 100 score to be searched for and shown in interactive matching; used to optimize performance for interactive matching	Integer	20	Interactive match
	SearchTimeout	Sets time-out for searching (in second) for interactive matching	Float	5	Interactive match
Building Locators	StoreStandardizedRefData	Prestandardizes the reference data and stores standardized value in the locator indexes	Boolean	FALSE	
	UseManagedDictionaries	Managed dictionaries in locator for optimizing performance	Boolean	FALSE	It is recommended that you not change the value.
	StorageSegmentSizeKB	Internal unit for data page size (byte)	Integer	262144	It is recommended that you not change the value.
Reverse Geocoding	SearchDistance	Sets search distance for finding nearest addresses for the input point; used by the Address Inspector and Reverse Geocode tool	Integer	100	
	SearchDistanceUnits	The units used for Search Distance	esriUnits	Meters	

Properties for all locators supported by a style file may be set at this XPath in the style file:

locators/locator/properties

```
<properties>
  <prop name="MinimumMatchScore" type="float">85</prop>
  <prop name="MinimumCandidateScore" type="float">10</prop>
  <prop name="SpellingSensitivity" type="float">80</prop>
  <prop name="MatchIfScoresTie" type="Boolean">true</prop>
  <prop name="WriteXYCoordFields" type="Boolean">false</prop>
  <prop name="WriteStandardizedAddressField" type="Boolean">false</prop>
  <prop name="WriteReferenceIDField" type="Boolean">true</prop>
  <prop name="WriteAdditionalOutputFields" type="Boolean">true</prop>
  <prop name="WritePercentAlongField" type="Boolean">false</prop>
  <prop name="RuntimeMemoryLimit" type="Int">512000000</prop>
  <prop name="ShowElapsedTime" type="Boolean">false</prop>
  <prop name="reportScorePerComponent" type="Boolean">false</prop>
  <prop name="MaxCandidates" type="Int">200</prop>
  <prop name="MaxPerfectCandidates" type="Int">200</prop>
  <prop name="SearchTimeout" type="float">1</prop>
</properties>
```

Many of these properties may be set or adjusted after locator creation with the property sheet for a Locator, which you can access from ArcCatalog™ or the Catalog window in ArcMap™:



Some properties are set in the Mapping Schemas section. For example, the custom zone definition designed in appendix F has these properties, altered from City, State, ZIP in the original:

```
<properties>
  <prop name="StorageSegmentSizeKB" type="Int">128</prop>
  <prop name="supportsEmptyHouseNumber" type="Boolean">false</prop>
  <prop name="supportsIntersections">false</prop>
  <prop name="StoreStandardizedRefData">false</prop>
  <prop_list name="BatchPresortInputs">
    <value>Region</value>
    <value>PostCode</value>
    <value>Locality</value>
  </prop_list>
</properties>
```

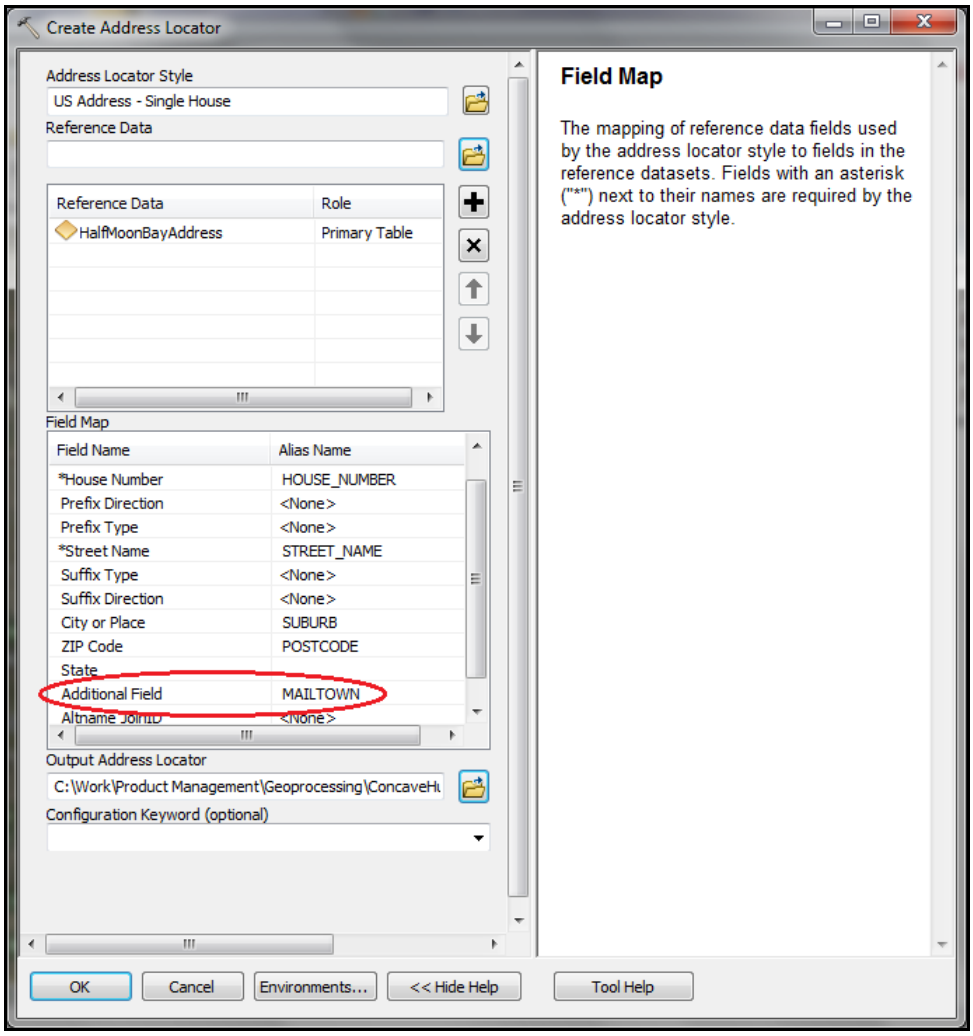
The mapping schema for address range interpolation styles like Nickel supports properties relevant to centerline interpolation:

```
<properties>
  <prop name="supportsEmptyHouseNumber" type="Boolean">false</prop>
  <prop name="supportsIntersections" type="Boolean">true</prop>
  <prop name="SideOffset">20</prop>
  <prop name="SideOffsetUnits">Feet</prop>
  <prop name="EndOffset">3</prop>
  <prop name="EndOffsetUnits">Percent</prop>
  <prop name="Interpolate.SideValue.Left">L</prop>
  <prop name="Interpolate.SideValue.Right">R</prop>
  <prop name="IntersectionConnectors">&amp; @ | and</prop>
  <prop name="WritePercentAlongField" type="Boolean">true</prop>
</properties>
```

Nickel is a term coined for single-range centerline data, as a counterpart to the dual-range Dual Independent Map Encoding (DIME) format developed by the U.S. Census Bureau in the 1960s.

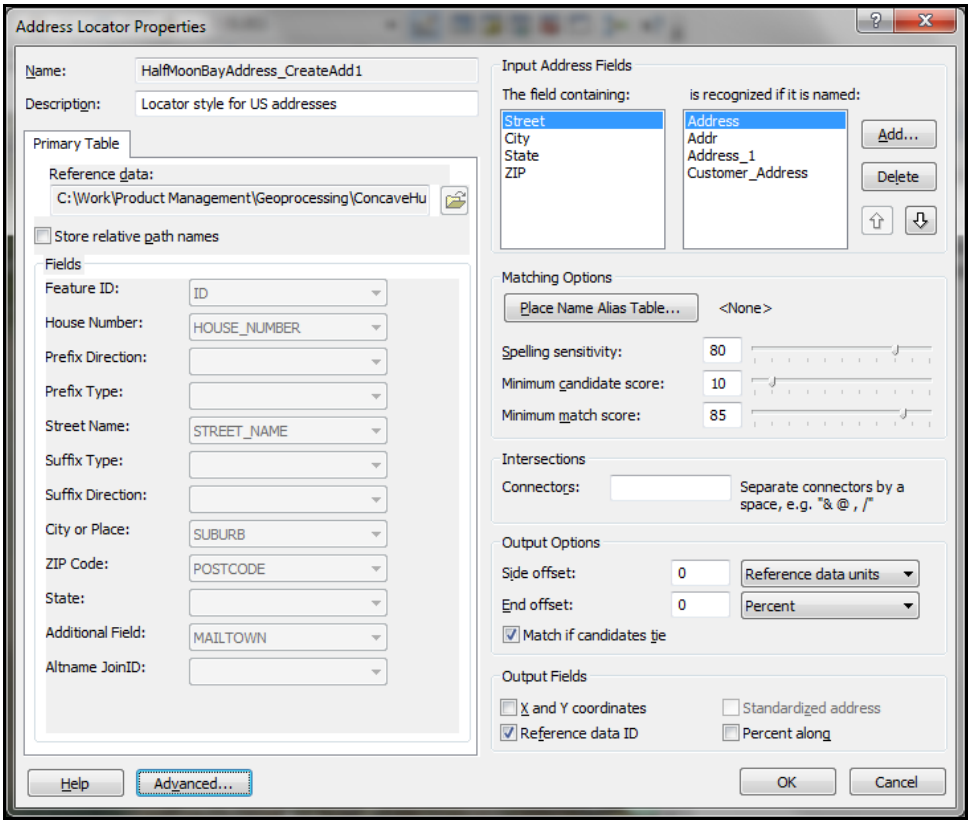
Appendix B: Example of a Runtime Property

Not all locator customization is controlled entirely by the style file; some locator properties may be set during locator creation or at run time. An example is whether or not a locator writes an additional field from the reference data when batch geocoding. This capability is switched on or off by the WriteAdditionalOutputFields property. If this is set to true, then the Create Address Locator dialog box allows the user to select a reference dataset field for output in geocoding, as below for "MAILTOWN".



J-9969

Locators have a set of properties that can be altered at run time. If you open a locator's properties dialog box, you will see some editable properties:



It is very useful to be able to adjust spelling sensitivity or candidate and match scores when rematching addresses.

Appendix C: Examples of Adding Aliases

Aliases provide a mechanism to support word substitution beyond that provided by spelling substitution. Whichever alias form is found in reference or input data will be treated as equivalent.

The first example of modifying an alias will be to add an entry to the CityAliases list. These are words commonly used in city names, where colloquial usage often includes abbreviations and misspellings. In this case, the city Carmel by the Sea is usually called "Carmel," so we simply insert an entry in alphabetic order in the alias list.

CityAlias added for Carmel:

```
<alias_def>
  <alt>brook</alt>
  <alt>brk</alt>
</alias_def>
<alias_def>
  <alt>carmel</alt>
  <alt>carmel by the sea</alt>
</alias_def>
<alias_def>
  <alt>camp</alt>
  <alt>cmp</alt>
  <alt>cp</alt>
</alias_def>
```

brook	"brook" "brk"
carmel	"carmel" "carmel by the sea"
camp	"camp" "cmp" "cp"

Adding aliases is a simple process for any alias list.

Another example would be StateAliases edited to support the full name for Rhode Island:

```
<alias_def>
  <alt>ri</alt>
  <alt>rhode island</alt>
  <alt>rhode island and providence plantations</alt>
</alias_def>
```

pr	"pr" "puerto rico" "puerto"
ri	"ri" "rhode island" "rhode island and providence plantations"
sc	"sc" "s car" "south carolina" "s carolina" "so carolina"

Appendix D: Examples of Adding Alternate Values

The grammar defines alternate values for many elements; the richest variability is in prefix types, suffix types, and unit names, and these will likely need some attention for your reference data. Adding alternate values is straightforward, but you need to be aware of how they are used. For example, in the US style, prefix types are grouped into various forms by usage. For example, the various forms for the prefix type "County" are given in an element _cty, and _cty is referenced in _ctyln and _cord so that all alternates for "County" are included in the other elements.

We will add an additional prefix type, "Paseo," which we understand is not referenced to other prefix types, so it can go in the **pre_type_no_sthwy** section. The usage we look to support is the street name "Paseo del Norte."

Add a reference in the **pre_type_no_sthwy** list:

```
<alt ref="_parkrd"/>
<alt ref="_paseo"/>
<alt ref="_ramp"/>
```

As *Paseo* may also be a suffix type, add the reference definition into suffix types below _park:

```
<def name="_paseo" is_alias_list="true">
  <alt>Paseo</alt>
  <alt>Pseo</alt>
  <alt>Pso</alt>
</def>
```

```
_park    :  "Park"
         |  "Parks"
         |  "Prk"
         |  "Pk"
         ;
_paseo   :  "Paseo"
         |  "Pseo"
         |  "Pso"
         ;
_pass    :  "Pass"
         ;
```

A simpler case of adding a suffix type would be to support "Close".

```
<def name="_cll" is_alias_list="true">
  <alt>Cll</alt>
  <alt>Calle</alt>
</def>
<def name="_cls" is_alias_list="true">
  <alt>Cls</alt>
  <alt>Close</alt>
</def>
<def name="_cmn" is_alias_list="true">
  <alt>Cmn</alt>
  <alt>Common</alt>
</def>
```

```
_cll     :  "Cll"
         |  "Calle"
         ;
_cls     :  "Cls"
         |  "Close"
         ;
_cmn     :  "Cmn"
         |  "Common"
         ;
```

Appendix E: Example of Defining a New House Number Format

Your reference data house numbers may not be parsed into separate fields for the base house number and other qualifiers. In this case, you will need to recognize the values you have. For example, the following are some irregular forms of house number:

House Number Example	Source	Interpretation
1/101–11/101	New Zealand	Units 1 to 11 at house number 101
10bis	Italy	House number with Latin ordinal numerical adverb suffix (bis, ter, quater, quinquies, sexies, septies, octies, novies, decies)
11/rosso	Italy	House number with door color suffix, black (*nero*) for residential, red (*rosso*) for businesses
IV/24	Hungary	House number 24 with residence floor number 4 in Roman numerals

These forms of house number may be handled by adding alternate values to the AlphaNumericHouse element in the House Numbers section of the style file. The values will be defined by regular expressions matching the required forms.

When using regular expressions in locator definitions, you do not have the full power of patterns found in many programming languages and text editors; there is only a subset of functionality available. The expression syntax allowed is limited to

Pattern Syntax	Description
.	Any single character
+	The preceding expression one or more times
[]	Enclosing a set of characters
\t	The literal TAB character
\	Escape the next special character
$	End of line

Thus, while in a programming language, we might be able to express the Hungarian numbering case as "(?:XC|XL|L?X{0,3})(?:IX|IV|V?I{0,3})/[0-9]+" (being very generous about building heights in Hungary), we are limited to expressing the character set expected to be found. For the cases above, then, appropriate expressions would be

House Number Form	Matching Regular Expression
1/101–11/101	[0-9]+/[0-9]+-[0-9]+/[0-9]+
10bis	[0-9]+(?:bis|ter|quater|quinquies|sexies|septies|octies|novies|decies)
11/rosso	[0-9]+/(?:nero|rosso)
IV/24	[IVXLDCM]+/[0-9]+

Note: For the Italian examples, the alternate literals would best be encoded in the XML as

```
<alt>
   <elt post_separator="none">`[0-9]+`</elt>
   <elt ref="Cont" />
</alt>
<def name="Cont">
   <alt>A</alt>
   <alt>B</alt>
   <alt>C</alt>
   <alt>D</alt>
</def
```

This is because the (|) pattern is not supported in locator style files.

We will show, adding only one complete example of the above to the AlphaNumericHouse element, the example for Hungary. We will call the house number form HungarianNumber and search for it only in a fallback situation.

First, add a basic element section definition for HungarianNumber after the number element:

```
<def name="number">
  <alt>`[0-9]+`</alt>
</def>
<def name="HungarianNumber">
  <alt>`[IVXLDCM]+/[0-9]+`</alt>
</def>
```

Then, add it as a fallback option for AlphaNumericHouse in the House Numbers section:

```
<alt fallback="true" ref="Fraction" />
<alt fallback="true" ref="HungarianNumber" />
```

The new house number definition will now be visible in the browser view:

```
number          :  "`[0-9]+`"
                ;
HungarianNumber :  "`[IVXLDCM]+/[0-9]+`"
                ;
```

```
AlphaNumericHouse  :  number OptFraction
                   |  alpha
                   |  alpha ↔OptHyphen↔number
                   |  number ↔Hyphen↔alpha
                   |  number→alpha
                   |  number "-" number alpha
                   |  Fraction
                   |  HungarianNumber
                   ;
```

Appendix F: Example of Defining Custom Zone Elements and a Supporting Schema

Most countries promulgate a postal code schema used by the postal agency for mail sorting purposes. The importance of postcodes to postal agencies is to facilitate aggregating addresses for their business processes; likewise, we use zone elements like postcode for our purpose—to filter the number of records considered for match candidacy. Zone values are the point of entry for a geocode.

Some postcode schemes are very powerful filters and eliminate the need for all other zone information; others are weak and depend on one or more additional zone elements to make an address unambiguous. We will design zone element definitions for a weak postcode case where three zone elements are used: **Locality**, **Region**, and **Postcode**. A locality might be a suburb or township; a region might be a larger political area.

Your postal agency will have designed a zone component set into its addressing guidelines, stating whether each component is essential or optional for postage. Regardless of this, you should allow for the possibility of zone values being missing or faulty, because people will often record an address not knowing one or more zone values.

Locator styles have two sections where zone elements are defined and referenced; the XPaths are **locators/locator/grammar/Zones** and **locators/locator/grammar/ZonesNoSearch**. Elements defined (not just referenced) in Zones, when found in an address, are used to restrict the index search to records with an agreeing zone field value. Elements defined in ZonesNoSearch will be used for scoring but will not be used to restrict index search for match candidates. Zone fields that have a high value in disambiguation should always be defined in Zones with a high score weight and redefined with a different name in ZonesNoSearch. Where ZonesNoSearch elements are used in the Zone element in Zones, the Zone option should have a reduced score total.

Let's suppose our jurisdiction depends more on the combination of locality and postcode to disambiguate addresses, perhaps because it is situated entirely within a larger political entity or has large rural spaces where the region name is not useful for disambiguation.

All this leads us to a design decision for the Zone element (superscripts are score weights):

- Locality^{40} Region^{10} Postcode^{50}—with all elements restricting the search; if no candidates, then:

- Locality^{30} Region^{10} Postcode^{50}—with only Postcode restricting the search; if no candidates, then:

J-9969

- Locality^{50} Region^{10} Postcode^{30}—with only Locality restricting the search; if no candidates, then:

- Locality^{30} Region^{10} Postcode^{50}—with only Region restricting the search

In the fallback cases (2nd, 3rd, and 4th cases), the search-restricting elements will use a nonempty definition; note that the total score weight (90) is reduced compared to the primary query (100). This design supports geocoding addresses where two or more zone elements are wrong or missing, albeit with a lowered score.

Below is our starting point for how zones are defined:

```
[-] Zones
Zone        :  OptCity^20 OptState^30 OptGenZIP^50
            |  OptCityNoSearch^10 OptStateNoSearch^30 GenZIP^50
            |  City^50 OptState^30 OptZipNoSearch^10
            ;
OptState    :  State
            |
            ;
OptCity     :  City
            |
            ;
OptGenZIP   :  GenZIP
            |
            ;
City        :  wordlist
            ;
State       :
            ;
GenZIP      :  ZIP5 OptPlus4
            ;
ZIP5        :  "`[0-9][0-9][0-9][0-9][0-9]`"
            ;
ZIP4        :  "`[0-9][0-9][0-9][0-9]`"
            ;
ZIPPlus4    :  ZIP5^100 OptHyphen^0 ↔ZIP4^50
            ;
OptPlus4    :
            |  OptHyphen^0 ↔ZIP4^50
            ;
[-] ZonesNoSearch
OptCityNoSearch   :  CityNoSearch
                  |
                  ;
CityNoSearch      :  wordlist
                  ;
OptStateNoSearch  :  StateNoSearch
                  |
                  ;
StateNoSearch     :  wordlist
                  ;
OptZipNoSearch    :  GenZIPNoSearch
                  |
                  ;
GenZIPNoSearch    :  ZIP5NoSearch OptPlus4
                  ;
ZIP5NoSearch      :  "`[0-9][0-9][0-9][0-9][0-9]`"
                  ;
```

There is also a reference to a Zone element in top-level elements:

```
Postal  :  GenZIP
           { ['#ZIPSearch'] }
        ;
```

The XML editing strategy we will adopt is to add our definitions into the style and remove existing ones. We also need to check where existing Zone elements are referenced elsewhere in the style.

We will use a postcode form of four integers, for example, 0128 or 3447. The regular expression for exactly four integer characters is "[0-9]{4}". Leading zeroes are supported by this expression.

Let's start with defining PostCode and OptPostCode:

```
<!--PostCode-->

<def name="PostCode">
  <alt>`[0-9]{4}`</alt>
</def>
```

```
<def name="OptPostCode">
  <alt ref="PostCode" />
  <alt />
</def>
```

Here is the browser view:

```
OptPostCode  :  PostCode
             |
             ;
City         :  wordlist
             ;
State        :
             ;
PostCode     :  "`[0-9]{4}`"
             ;
```

We will continue adding definitions or redefining existing ones. The completed browser view of this work is below, but there are a couple of edits not displayed in the browser. The CityAliases top-level element has been renamed to LocalityAliases and is used by both Locality and Region:

```
<!--Locality Parameter defined as word and using LocalityAliases for alias names-->

<def name="Locality">
  <alias_list_ref ref="LocalityAliases"/>
  <alt ref="wordlist"/>
</def>

<!--Region Parameter defined as word and sharing Locality Alias list-->

<def name="Region">
  <alias_list_ref ref="LocalityAliases"/>
  <alt ref="wordlist"/>
</def>
```

There is also a change to a top-level elements function call:

```
Postal                          :  PostCode
                                   { ['#PostalSearch'] }
                                ;
```

J-9969

The source style used the "ZIPSearch" search context for Postal, but we will use "PostalSearch." There is also a search context called "CitySearch." These search contexts are defined by the engine and manage a set of tests for the element.

The final form of the **locators/locator/grammar/Zones** and **locators/locator/grammar/ZonesNoSearch** sections looks like this:

```
[-] Zones
Zone          :  OptLocality40 OptRegion10 OptPostCode50
              |  OptLocalityNoSearch30 OptRegionNoSearch10 PostCode50
              |  Locality30 OptRegionNoSearch10 OptPostCodeNoSearch50
              |  OptLocalityNoSearch30 Region10 OptPostCodeNoSearch50
              ;
OptLocality   :  Locality
              |
              ;
OptRegion     :  Region
              |
              ;
OptPostCode   :  PostCode
              |
              ;
Locality      :  wordlist
              ;
Region        :  wordlist
              ;
PostCode      :  "`[0-9]{4}`"
              ;

[-] ZonesNoSearch
OptLocalityNoSearch  :  LocalityNoSearch
                     |
                     ;
LocalityNoSearch     :  wordlist
                     ;
OptRegionNoSearch    :  RegionNoSearch
                     |
                     ;
RegionNoSearch       :  wordlist
                     ;
OptPostCodeNoSearch  :  PostCodeNoSearch
                     |
                     ;
PostCodeNoSearch     :  "`[0-9]{4}`"
                     ;
```

There is more work to do for our zone elements. Because this style will not need any concept of US States as zones, we will remove **Locators/locator/grammar/US States**.

There are, additionally, a number of references to US States in **locators/locator/grammar/Aliases/StateAliases**, treated just as words. Let's suppose some streets in our jurisdiction may use the names of U.S. states ("Nevada Street"), so we can move the definitions up into the StreetNameAliases section and remove StateAliases.

Now that we have changed the way zone grammar is defined, we must change the way reference data supports our new definitions. For example, the Mapping Schemas section for SingleAddress references City, ZIP, and State definitions, and there are sections for styles built for USZIP and derivatives, so we will change or remove these sections:

```
[−] Mapping Schemas
SingleAddress
Description:            Single house addresses (points)
Geometry type:          point
Fields:                 [−]
                        Shape      Shape field
                        No preferred names defined
                        ID         Unique ID field
                        No preferred names defined
                        House      House number
                        Maps to grammar element House
                        Names:     [+]
                        PreDir     Prefix Direction
                        Maps to grammar element prefix
                        Names:     [+]
                        PreType    Prefix Type
                        Maps to grammar element pretype
                        Names:     [+]
                        StreetNameStreet Name
                        Maps to grammar element StName
                        Names:     [+]
                        SufType    Suffix Type
                        Maps to grammar element suftype
                        Names:     [+]
                        SufDir     Suffix Direction
                        Maps to grammar element suffix
                        Names:     [+]
                        Locality   Locality, Suburb or Township
                        Maps to grammar element Locality
                        Names:     [+]
                        PostCode   PostCode
                        Maps to grammar element PostCode
                        Names:     [+]
                        Region     Region or Major Locality
                        Maps to grammar element Region
                        Names:     [+]
                        Rank       Specifies rank (primary or secondary) for alternate records of the same geometry
                        Names:     [+]
                        User_fld   Additional field
                        No preferred names defined
Filter criteria (default):   $StreetName <> '' AND $StreetName <> ' ' AND UPPER($StreetName) <> 'UNNAMED' AND
                             UPPER($StreetName) <> 'UNNAMED STREET'
Filter criteria (MSAccess):  $StreetName <> '' AND $StreetName <> ' ' AND $StreetName <> 'UNNAMED' AND $StreetName
                             <> 'UNNAMED STREET'
Filter criteria (Oracle):    $StreetName IS NOT NULL AND $StreetName <> ' ' AND UPPER($StreetName) <> 'UNNAMED'
                             AND UPPER($StreetName) <> 'UNNAMED STREET'
```

New SingleAddress Mapping Schema Definition after Adding the Zone Fields

Not visible in the browser view but also needing attention in the Mapping Schemas section is how the table field roles and indexes are defined to use the zone elements. In our case, we replace City, State, and ZIP with Locality, Region, and PostCode in a number of places. Then we add the zone fields to the primary table index and the query fields set.

```
<index>
  <dictionary ref="UnitType"/>
  <dictionary ref="UnitNumber"/>
  <dictionary ref="House"/>
  <dictionary ref="PreDir"/>
  <dictionary ref="PreType"/>
  <dictionary ref="StreetName" search="true"/>
  <dictionary ref="Urbanization"/>
  <dictionary ref="SufType"/>
  <dictionary ref="SufDir"/>
  <dictionary ref="Locality" search="true" input_field_ref="Locality" />
  <dictionary ref="Region" search="true" input_field_ref="Region" />
  <dictionary ref="PostCode" search="true"  input_field_ref="PostCode" />
```

Mapping Schemas Index (with Search)

J-9969

```
<relationship>
  <field_ref ref="StreetName"/>
  <field_ref ref="Region"/>
</relationship>
<relationship>
  <field_ref ref="StreetName"/>
  <field_ref ref="Region"/>
</relationship>
<relationship>
  <field_ref ref="StreetName"/>
  <field_ref ref="Locality"/>
  <field_ref ref="Region"/>
</relationship>
<relationship>
  <field_ref ref="StreetName"/>
  <field_ref ref="Locality"/>
  <field_ref ref="PostCode"/>
</relationship>
<relationship>
  <field_ref ref="StreetName"/>
  <field_ref ref="Region"/>
  <field_ref ref="PostCode"/>
</relationship>
<relationship>
  <field_ref ref="StreetName"/>
  <field_ref ref="Region"/>
  <field_ref ref="Locality"/>
  <field_ref ref="PostCode"/>
</relationship>
```

Relationships

```
<reverse_relationship>
  <field_ref ref="UnitType"/>
  <field_ref ref="UnitNumber"/>
  <field_ref ref="House"/>
  <field_ref ref="PreDir"/>
  <field_ref ref="PreType"/>
  <field_ref ref="StreetName"/>
  <field_ref ref="Urbanization"/>
  <field_ref ref="SufType"/>
  <field_ref ref="SufDir"/>
  <field_ref ref="Locality"/>
  <field_ref ref="PostCode"/>
  <field_ref ref="Region"/>
  <field_ref ref="ID"/>
  <field_ref ref="User_fld" />
</reverse_relationship>
```

Reverse Relationships

```
<field ref="SufDir" field_role_ref="Primary.SufDir" />
<field ref="Locality" field_role_ref="Primary.Locality" />
<field ref="PostCode" field_role_ref="Primary.PostCode" />
<field ref="Region" field_role_ref="Primary.Region" />
<field ref="User_fld" field_role_ref="Primary.User_fld" />
```

Primary Query Fields

```
<field ref="SufDir"      field_role_ref="AltStreet.SufDir" />
<field ref="Locality"    field_role_ref="Primary.Locality" />
<field ref="PostCode"    field_role_ref="Primary.PostCode" />
<field ref="Region"      field_role_ref="Primary.Region" />
<field ref="User_fld"    field_role_ref="Primary.User_fld" />
```

Fallback Query Fields

Now that we have created a new logical model to fit our zoning definitions, we need to edit the physical model definitions in the Reference Data Styles section.

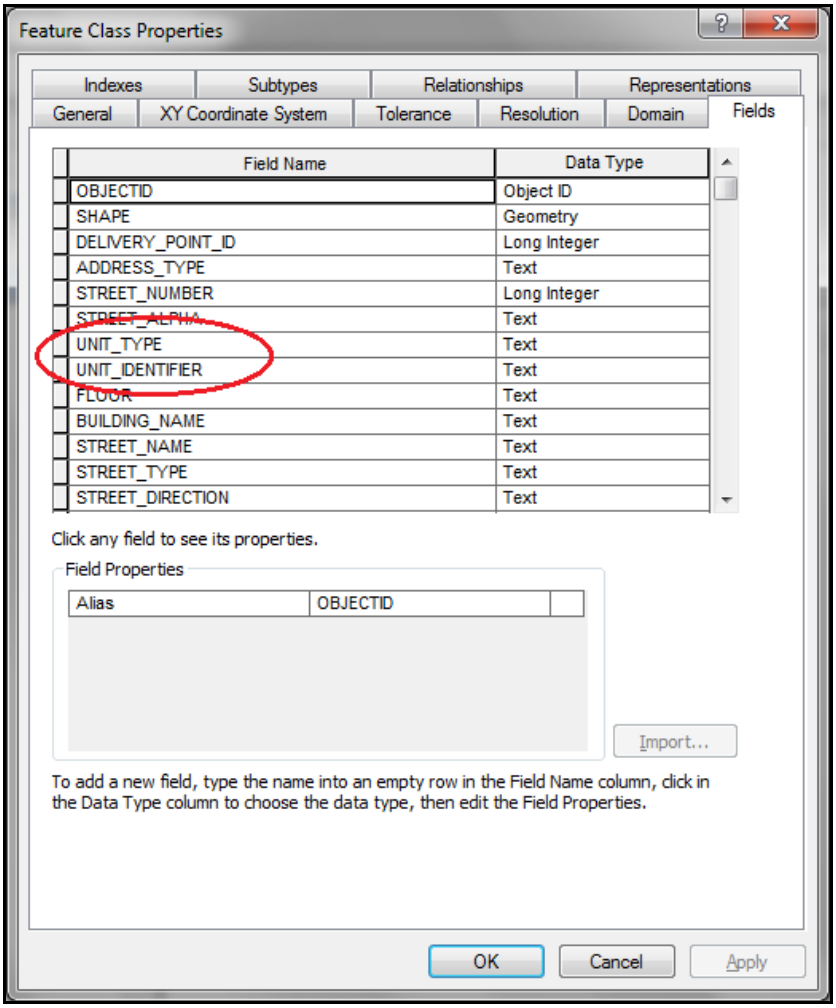
```
[−] Reference Data Styles
Single House
MY Single House Addresses
  Primary (Address feature class)
  [−] Fields
      Geometry [Primary.Shape]
      [+] Preferred names
      Feature ID [Primary.ID]
      [+] Preferred names
      Unit Type [Primary.UnitType]
      [+] Preferred names
      Unit Number [Primary.UnitNumber]
      [+] Preferred names
      House Number [Primary.House] (Required)
      [+] Preferred names
      Prefix Direction [Primary.PreDir]
      [+] Preferred names
      Prefix Type [Primary.PreType]
      [+] Preferred names
      Street Name [Primary.StreetName] (Required)
      [+] Preferred names
      Suffix Type [Primary.SufType]
      [+] Preferred names
      Suffix Direction [Primary.SufDir]
      [+] Preferred names
      Locality, Suburb or Township [Primary.Locality]
      [+] Preferred names
      Postcode [Primary.PostCode]
      [+] Preferred names
      Region [Primary.Region]
      [+] Preferred names
```

Editing the reference data styles is described in appendix G.

Appendix G: Example of Adjusting a Mapping Schema

Let's suppose our style needs to support reference data that includes fields for housing units, such as apartments or building floors. The reference data might have z-coordinate (height) values for units on different floors, or separate apartments might have different x,y coordinates for the same house number on a street. Addresses might have the form "Studio 7, 134 Main St, Georgetown, Barbados" or "123 Kansas St, Apartment 2, Redlands, CA." The reference data has a field for unit type and a field for unit number.

Here is an example of such reference data:



In this case, we need the unit information—unit type and unit number—to participate in the match. Therefore, we need to edit the mapping schema.

First, to support the forms of address given, we adjust the top-level element NormalAddress to recognize unit information on either side of the house and street data:

```
NormalAddress        :  House15 ≡FullStreetName60 OptionalUnit0
                     |  OptionalUnit0 ≡House15 ≡FullStreetName60
                     ;
```

In the browser view, we can see that OptionalUnit is defined to be either empty or a reference to unitAndNumber, while unitAndNumber references the elements HousingUnitType and AlphaNumericUnit. These are the grammar elements that need to be mapped in the schema.

```
unitAndNumber        :  HousingUnitType OptNumSeparator AlphaNumericUnit
                     |  ↔NumSeparator↔AlphaNumericUnit
                     |  ↔NumSeparator↔latinAlphaWord
                     ;
```

Here, we insert the mapping schema changes into SingleAddress ahead of the House element:

```
<mapping_schema name="SingleAddress" geom_type="point">
  <desc>Single house addresses (points)</desc>
  <desc>For point address datasets.</desc>
  <fields>
    <field name="Shape" type="geometry">
      <desc>Shape field</desc>
    </field>
    <field name="ID">
      <desc>Unique ID field</desc>
    </field>
    <field name="UnitType" grammar_ref="HousingUnitType">
      <desc>Unit Type</desc>
      <preferred_name>UNITTYPE</preferred_name>
      <preferred_name>UNIT_TYPE</preferred_name>
    </field>
    <field name="UnitNumber" grammar_ref="AlphaNumericUnit">
      <desc>Unit number</desc>
      <preferred_name>UNITNUMBER</preferred_name>
      <preferred_name>UNIT_IDENTIFIER</preferred_name>
    </field>
    <field name="House" grammar_ref="House">
      <desc>House number</desc>
      <preferred_name>HN</preferred_name>
      <preferred_name>ADDRESS</preferred_name>
    </field>
```

In the browser view, this looks like the image below:

```
[−] Mapping Schemas
SingleAddress
Description:        Single house addresses (points)
Geometry type:      point
Fields:             [−]
                    Shape        Shape field
                    No preferred names defined
                    ID           Unique ID field
                    No preferred names defined
                    UnitType     Unit Type
                    Maps to grammar element HousingUnitType
                    Names:       [+]
                    UnitNumberUnit number
                    Maps to grammar element AlphaNumericUnit
                    Names:       [+]
                    House        House number
                    Maps to grammar element House
                    Names:       [+]
```

It is now necessary to edit the Reference Data Styles section so that Single House reference data fields will be logically connected to the mapping schema. Edit the style file to insert the unit type and number references into the mapping schema fields section:

```
<field_role name="Primary.UnitType" required="false">
  <display_name>Unit Type</display_name>
  <preferred_name>UNITTYPE</preferred_name>
  <preferred_name>UNIT_TYPE</preferred_name>
  <preferred_name>APARTMENT_TYPE</preferred_name>
</field_role>
<field_role name="Primary.UnitNumber" required="false">
  <display_name>Unit Number</display_name>
  <preferred_name>UNITNUMBER</preferred_name>
  <preferred_name>UNIT_NUMBER</preferred_name>
  <preferred_name>APARTMENT_NUMBER</preferred_name>
</field_role>
```

In the browser view, we can see the result:

```
[−] Reference Data Styles
Single House
MY Single House Addresses
  Primary (Address feature class)
  [−] Fields
      Geometry [Primary.Shape]
      [+] Preferred names
      Feature ID [Primary.ID]
      [+] Preferred names
      Unit Type [Primary.UnitType]
      [−] Preferred names
          UNITTYPE
          UNIT_TYPE
          APARTMENT_TYPE
      Unit Number [Primary.UnitNumber]
      [−] Preferred names
          UNITNUMBER
          UNIT_NUMBER
          APARTMENT_NUMBER
      House Number [Primary.House] (Required)
      [+] Preferred names
```

This connects the new fields to the logical schema, but we also need to include the fields in the index for the schema.

```
<mapping_schema name="SingleAddress" geom_type="point">
  <desc>Single house addresses (points)</desc>
  <desc>For point address datasets.</desc>
  <fields>
  <selection_clause dbms="default">
  <selection_clause dbms="MSAccess">
  <selection_clause dbms="Oracle">
  <index>
    <dictionary ref="UnitType"/>
    <dictionary ref="UnitNumber"/>
    <dictionary ref="House"/>
```

We must also add a reverse relationship for the index:

```
<reverse_relationship>
  <field_ref ref="UnitType"/>
  <field_ref ref="UnitNumber"/>
  <field_ref ref="House"/>
```

We must then include the fields in the schema outputs:

```
<outputs>
  <output component="Shape" type="geometry"/>
  <output component="Status" candidate_mode="false" length="1"/>
  <output component="Score" type="float" decimal_digits="2"/>
  <output component="Match_addr" length="120"/>
  <output ref="UnitType" batch_mode="false" length="12"/>
  <output ref="UnitNumber" batch_mode="false" length="6"/>
  <output ref="House" batch_mode="false" length="12"/>
```

This completes the schema mapping edits, but the reference data styles will also need editing to support the new fields.

UnitType and UnitNumber will need to be added to the primary table field roles.

```
<ref_data_style>
  <name>Single House</name>
  <desc>MY Single House Addresses</desc>
  <table_roles>
    <table_role name="Primary">
      <display_name>Primary Table</display_name>
      <desc>Address feature class</desc>
      <field_roles>
        <field_role name="Primary.Shape" is_geometry="true">
        <field_role name="Primary.ID" required="false">
        <field_role name="Primary.UnitType" required="false">
          <display_name>Unit Type</display_name>
          <preferred_name>UNITTYPE</preferred_name>
          <preferred_name>UNIT_TYPE</preferred_name>
          <preferred_name>APARTMENT_TYPE</preferred_name>
        </field_role>
        <field_role name="Primary.UnitNumber" required="false">
          <display_name>Unit Number</display_name>
          <preferred_name>UNITNUMBER</preferred_name>
          <preferred_name>UNIT_NUMBER</preferred_name>
          <preferred_name>APARTMENT_NUMBER</preferred_name>
        </field_role>
        <field_role name="Primary.House" required="true">
```

They must also be added to the primary and fallback queries:

```
<data_source type="indexed">
  <mapping_schema ref="SingleAddressSearch" />
  <queries>
    <query>
      <tables>
        <table role_ref="Primary" />
      </tables>
      <fields>
        <field ref="ShapePtPoly" field_role_ref="Primary.Shape" />
        <field ref="ID" field_role_ref="Primary.ID" />
        <field ref="UnitType" field_role_ref="Primary.UnitType" />
        <field ref="UnitNumber" field_role_ref="Primary.UnitNumber" />
        <field ref="House" field_role_ref="Primary.House" />
```

```
<query>
  <tables>
    <table role_ref="Primary" />
    <table role_ref="AltStreet" />
  </tables>
  <fields>
    <field ref="ShapePtPoly" field_role_ref="Primary.Shape" />
    <field ref="ID"          field_role_ref="Primary.ID" />
    <field ref="UnitType" field_role_ref="Primary.UnitType" />
    <field ref="UnitNumber" field_role_ref="Primary.UnitNumber" />
    <field ref="House"       field_role_ref="Primary.House" />
```

Appendix H: Example of Adjusting the Scoring Weights

Score weights are easily tuned within the top-level and zone element sections of the style file. There should be little need to adjust scoring, except in unusual situations. One such situation would occur when you expect input addresses to be missing an address component, but you nevertheless want the engine to report high scores. This might happen when a national style is used only for data for a single city and the incoming addresses usually omit the zone data, even though your locator is created with zone fields populated. Local governments may experience this situation when using subsets of national scale reference data and working with addresses for their constituents, who assume the local government knows the city name and state.

Given the above scenario, you might do some exploratory geocoding of your reference data against your locator built from it, omitting a mixture of zone elements you expect to be missing in incoming addresses. Then adjust the Zone score in your FullNormalAddress until you get the scores you are looking for.

If this is the starting point for score weight:

```
FullNormalAddress        :  NormalAddress^70 OptSeparator^0 Zone^30
                            { ['#addressSearch'] }
                         ;
```

Adjust the weight for Zone downwards by editing the original score:

```
<def name="FullNormalAddress">
  <alt>
    <elt ref="NormalAddress" weight="70"/>
    <elt ref="OptSeparator" weight="0"/>
    <elt ref="Zone" weight="15"/>
    <result tag="Address">
      <search_value ref="addressSearch"/>
      <format_ref ref="format_normal_address"/>
    </result>
  </alt>
</def>
```

Yielding a score weight of 15 for Zone:

```
FullNormalAddress        :  NormalAddress^70 OptSeparator^0 Zone^15
                            { ['#addressSearch'] }
                         ;
```

Now, the Zone elements will have less of a contribution to the total score, but as you are confident the addresses pertain to a limited set of zone values, your scores reflect the confidence you have that the matches are correct.

Appendix I: Example of Adding a Top-Level Element

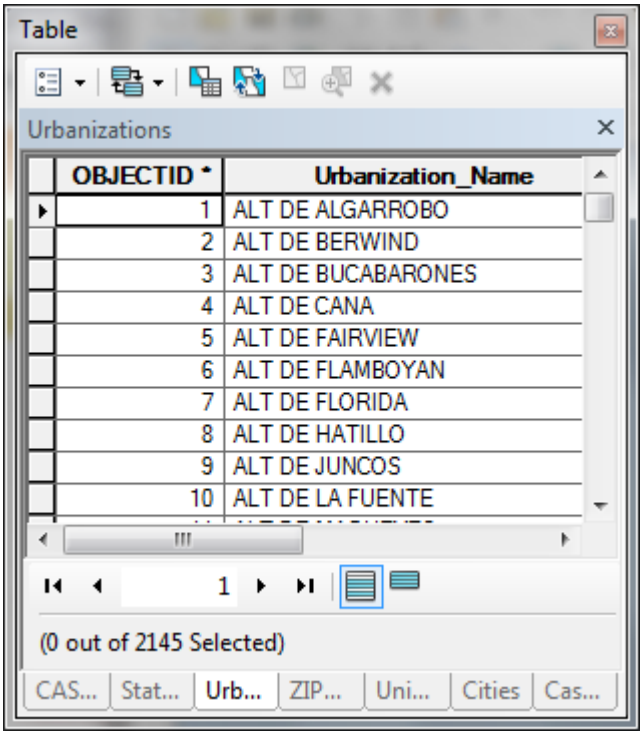
A new top-level element can be required to support a new form of address. Our example will be the Puerto Rican "Urbanization," an identifier used in addition to zone values. It may be used in place of a street name or appended to the full address.

MR JOHN DOE
1234 URB LOS OLMOS
PONCE PR 00731-1235

MRS JANE DOE
URB LAS GLADIOLAS
150 CALLE A
SAN JUAN PR 00926-0221

Let's assume that Urbanization will be an available match field in the reference data, so the mapping schema will need to be updated too, but first we'll describe adding the element.

The reference data is analyzed to determine a set of 2,145 recognized Urbanizations:



J-9969

Inspection of the Urbanization names shows that there is a reasonably limited number of what look like Urbanization prefix words—"ALT DE", "BRISAS DE", and so on—so one approach might be to define an Urbanization as these literal values followed by other words. For this exercise, however, we will simply use the entire set of full names as literals. This might seem to be an unwieldy number of values to edit into the style, but in fact we can automate this process with some Python scripting.

The following Python code writes an XML file that defines a new top-level element and provides values for it, taken from a geodatabase table.

```
# Create an XML document defining the Urbanization element
import arcpy
import xml.dom.minidom as minidom

#Source table
urbTable = r"C:\Temp\CASS.gdb\Urbanizations"

#Output XML path
includeXML = r"C:\Program Files
(x86)\ArcGIS\Desktop10.0\Locators\Urbanizations.xml"

#Create a new XML document
includeDoc = minidom.Document()

#Define the Urbanization element
def_el = includeDoc.createElement("def")
def_el.setAttribute("name","Urbanization")
includeDoc.appendChild(def_el)

#Append alt values for each source value
urbList = list(set([row.Urbanization_Name.strip() for row
in arcpy.SearchCursor(urbTable)]))
urbList.sort()
for urb in urbList:
    alt_el = includeDoc.createElement("alt")
    def_el.appendChild(alt_el)
    u = includeDoc.createTextNode(urb)
    alt_el.appendChild(u)

#Write the XML document
xmlFile = open(includeXML,"w")
xmlFile.write(includeDoc.toprettyxml(indent="  "))
xmlFile.flush()
xmlFile.close()
```

The output file looks like this in the XML editor:

```
<?xml version="1.0" ?>
<def name="Urbanization">
  <alt>
    ALT DE ALGARROBO
  </alt>
  <alt>
    ALT DE BERWIND
  </alt>
  <alt>
    ALT DE BUCABARONES
  </alt>
  <alt>
    ALT DE CANA
  </alt>
  <alt>
    ALT DE FAIRVIEW
  </alt>
```

We could also use Python to write new values into the appropriate XPath section in the style file, but in case of script errors, we write out a new file. Normal XML practice at this point would be to use an XML `<include>` tag in the style file, at the end of the top-level elements section, to include the new data:

```
<!-- Include Urbanizations -->
<include uri="Urbanizations.xml" xpath="/def"/>
```

This would make it easy to merge any later changes in source data into the style by updating the include file. However, the included data is not visible in the browser view of the style. While using an include tag will work when creating your new locator, you will probably want to validate your changes in the browser view, so we will copy and paste the new data into the style and comment out the include tag (you can uncomment it after testing the style and remove the added data).

After the copy-paste, the style file will look like this in the XML editor:

```
<!-- Include Urbanizations
<include uri="Urbanizations.xml" xpath="/def"/>  -->
<def name="Urbanization">
  <alt>
    ALT DE ALGARROBO
  </alt>
  <alt>
    ALT DE BERWIND
  </alt>
  <alt>
    ALT DE BUCABARONES
  </alt>
```

The browser view will then be (at the end of the Top level elements section):

```
MultiLineUnitAndNumberPrefix :  HousingUnitType OptNumSeparator AlphaNumericUnit
                             |  AlphaNumericUnit HousingUnitType
                             |  ↔NumSeparator↔AlphaNumericUnit
                             |  wordlist
                             ;
Urbanization                 :  " ALT DE ALGARROBO "
                             |  " ALT DE BERWIND "
                             |  " ALT DE BUCABARONES "
```

The next task to plug in the new Urbanization element is to add definitions of how it is going to be used in an address. Since an Urbanization may be appended to an address, we add it in the Location top-level element:

```
<section desc="Top level elements">
  <def name="Location">
    <alt>
      <elt ref="FullAddress" separator_list=".,;" />
    </alt>
    <alt>
      <elt ref="Urbanization"/>
      <elt ref="FullAddress"/>
    </alt>
    <alt ref="Coordinates"/>
    <alt ref="SpatialOperator"/>
  </def>
```

```
[-] Top level elements
Location                     :  FullAddress
                             |  Urbanization FullAddress
                             |  Coordinates
                             |  SpatialOperator
                             ;
```

An Urbanization element may also function as a street name, so we will add Urbanization to the StName options:

```
<def name="StName">
  <alias_list_ref ref="StreetNameAliases"/>
  <alt ref="name"/>
  <alt ref="Urbanization"/>
</def>
```

```
StName                       :  name
                             |  Urbanization
                             ;
```

Since we're adding support for an element used in Puerto Rican addresses, we will add Puerto Rico into the US States section. This is a simple edit, so we'll just show the browser view result (there is also a reference defined above the alias list):

```
st_pa  :  "pa"
       |  "penn"
       |  "penna"
       |  "pennsylvania"
       ;
st_pr  :  "pr"
       |  "puerto rico"
       ;
st_ri  :  "ri"
       |  "rhode island"
       ;
```

Another consideration with adding a new top-level element is how it participates in the index structure and searching, which is defined in the mapping schema. If the new element is in a separate field in the reference data, it must be included in the index structure to be used. In our example, an Urbanization can be

- Noise data ahead of an otherwise sufficient address
- Substituted for the street name

In the first case, we can exclude the element from participation in index searching, because we have enough other data, but in the second case, we need to use the value as a street name.

First, we add Urbanization to the logical schema for SingleAddress:

```
StreetName  Street Name
Maps to grammar element StName
Names:      [+]
Urbanization Urbanization
Maps to grammar element Urbanization
Names:      [-]
            URBANIZATION
```

Because our reference data has the Urbanization and street name elements in separate fields, we can easily build a relationship between Urbanization values and street name values. Street name values already have relationships to all the other address elements. To add the relationship between Urbanization and street names, edit the index section of the style file mapping schema. The XPath to this section is **locators/locator/mapping_schemas/mapping_schema/index**.

Add a dictionary for Urbanization:

```
<index>
  <dictionary ref="House"/>
  <dictionary ref="PreDir"/>
  <dictionary ref="PreType"/>
  <dictionary ref="StreetName" search="true"/>
  <dictionary ref="Urbanization"/>
```

Add a relationship between Urbanization and StreetName. This will create a forward lookup structure from StreetName to Urbanization:

```
<relationship>
  <field_ref ref="StreetName"/>
  <field_ref ref="Locality"/>
</relationship>
<relationship>
  <field_ref ref="StreetName"/>
  <field_ref ref="Urbanization"/>
</relationship>
```

Add Urbanization into the reverse relationships so the index will support finding StreetName from Urbanization (which is what we need in our case):

```
<reverse_relationship>
  <field_ref ref="House"/>
  <field_ref ref="PreDir"/>
  <field_ref ref="PreType"/>
  <field_ref ref="StreetName"/>
  <field_ref ref="Urbanization"/>
  <field_ref ref="SufType"/>
  <field_ref ref="SufDir"/>
  <field_ref ref="Locality"/>
  <field_ref ref="PostCode"/>
  <field_ref ref="Region"/>
  <field_ref ref="ID"/>
  <field_ref ref="User_fld" />
</reverse_relationship>
```

Now, we need to add Urbanization to the reference data styles where we expect to find it. If we have reference data for dual-range addresses that also has Urbanization in it, we add a reference in that section:

```
Street Name [Primary.StreetName] (Required)
[+] Preferred names
Urbanization [Primary.Urbanization]
[−] Preferred names
    URBANIZATION
```

Additionally, for the Query #1:

```
[−] Data Source
    Uses schema DIME
    [−] Query #1
    Tables:          $Primary
    Field mappings:  Shape          $Primary.Shape
                     ID             $Primary.ID
                     FromLeft       $Primary.FromLeft
                     ToLeft         $Primary.ToLeft
                     FromRight      $Primary.FromRight
                     ToRight        $Primary.ToRight
                     PreDir         $Primary.PreDir
                     PreType        $Primary.PreType
                     StreetName     $Primary.StreetName
                     Urbanization   $Primary.Urbanization
```

Appendix J: Example of Customizing Inputs

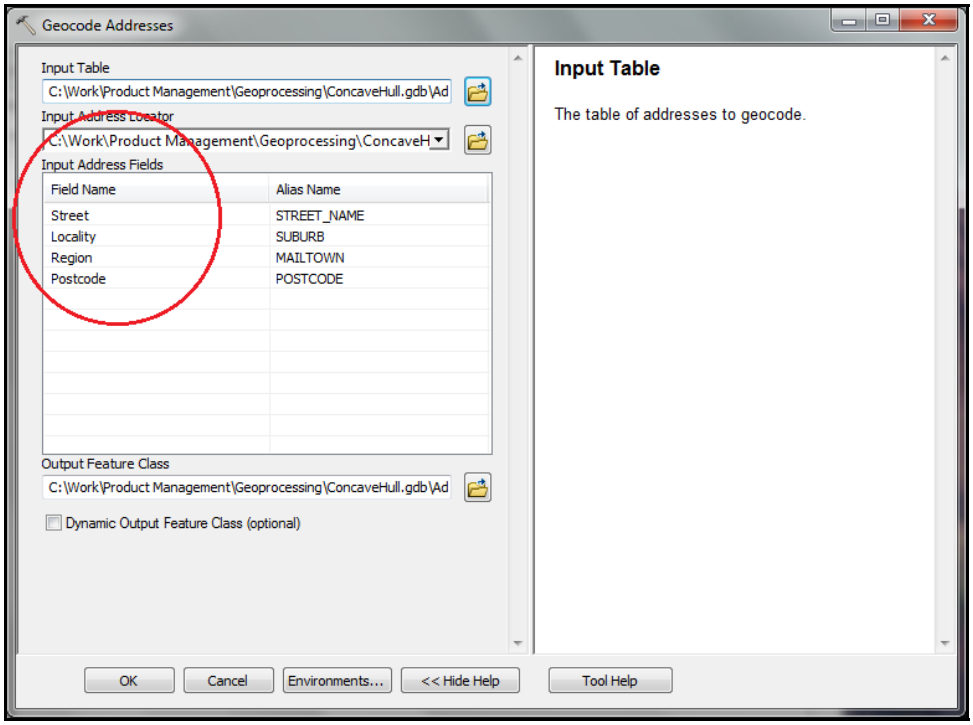
We saw in appendix F how to cater for a custom zoning schema; in the situation where an input field changes, the inputs section of the style needs to be edited. The XPath for this section is **locators/locator/inputs**.

Following the appendix F example, let's suppose our style requires Locality, Region, and Postcode inputs. The inputs section to support this might be as follows:

```
<inputs>
  <default_input name="Single Line Input" length="100" grammar_ref="Location">
    <caption xml:lang="en">Full Address</caption>
    <std_elt standard="FGDC">CompleteAddress</std_elt>
    <recognized_name>Address</recognized_name>
    <recognized_name>Addr</recognized_name>
    <recognized_name>Address_1</recognized_name>
    <recognized_name>Customer_Address</recognized_name>
  </default_input>
  <input name="Street" length="100" required="true">
    <caption xml:lang="en">Street or Intersection</caption>
    <recognized_name>Address</recognized_name>
    <recognized_name>Addr</recognized_name>
    <recognized_name>Address_1</recognized_name>
    <recognized_name>Customer_Address</recognized_name>
  </input>
  <input name="Locality" length="30" required="false">
    <caption xml:lang="en">Locality or Suburb</caption>
    <recognized_name>Locality</recognized_name>
    <recognized_name>Suburb</recognized_name>
    <recognized_name>Place</recognized_name>
    <recognized_name>Placename</recognized_name>
    <recognized_name>Town</recognized_name>
  </input>
  <input name="Region" length="30" required="false">
    <caption xml:lang="en">Region</caption>
    <recognized_name>Region</recognized_name>
    <recognized_name>Majorlocality</recognized_name>
    <recognized_name>City</recognized_name>
  </input>
  <input name="Postcode" length="4" required="false">
    <caption xml:lang="en">Postcode</caption>
    <recognized_name>PostCode</recognized_name>
    <recognized_name>PCode</recognized_name>
  </input>
</inputs>
```

J-9969

Now, when you use a locator with your style, the tool dialog box will appear as below, with the input element name properties visible in the dialog box.



Appendix K: Example of a New Intersection Type

Spatial Operators support offset style addresses like "100 meters north from 380 New York Street Redlands CA 92373" to geocode at an offset from the reference position for the base address. The scenario we want to support with this customization is a style of address commonly used to record traffic accidents. These look like "Daily Dr 100' West of Carmen Dr," which means the accident was on Daily Drive 100 feet west of the intersection of Daily Drive and Carmen Drive.

A rigorous approach to this type of geocode would be to write a plug-in that uses the built-in interpolation plug-in to find a coordinate that is exactly *on* Daily Drive. Using a directed offset will result in a coordinate that lies off the reference data centerline unless the road concerned runs exactly along the compass bearing given. However, for our purposes, we will assume that a given bearing is acceptable or that postprocessing to find linear reference route positions will result in an acceptable coordinate.

To support this new style of location, we will define a new intersection type, **IntersectionOffset**, and allow it to be a **FullAddress** (which is a Location).

We add IntersectionOffset to the supported FullAddress definitions:

```
<def name="FullAddress">
  <alt>
    <elt ref="FullNormalAddress" search_context="addressSearch"/>
  </alt>
  <alt selector="supportsIntersections" ref="FullIntersection" fallback_score="10"/>
  <alt selector="supportsIntersections" ref="IntersectionOffset" fallback_score="10"/>
</def>
```

The browser view is then the following:

```
FullAddress        :  FullNormalAddress
                   |  FullIntersection      | used only if no candidates with score above 10 has been previously found
                                            | used only if supportsIntersections is true
                   |  IntersectionOffset    | used only if no candidates with score above 10 has been previously found
                                            | used only if supportsIntersections is true
                   ;
```

Now, we define IntersectionOffset to call the directed_offset function with an embedded call to the intersection function:

```
<def name="IntersectionOffset">
  <alt>
    <elt ref="FullStreetName" search_context="leftStreet" weight="40"/>
    <elt ref="positiveRealNumber" weight="0"/>
    <elt ref="LinearUnits" weight="0"/>
    <elt ref="Bearing" weight="0"/>
    <elt ref="From" weight="0"/>
    <elt ref="FullStreetName" search_context="rightStreet" weight="40"/>
    <elt ref="OptSeparator"/>
    <elt ref="Zone" weight="20"/>
    <result tag="OffsetFromIntersection">
      <method ref="directed_offset">
        <parameter>
          <component_value component="_2"/>
        </parameter>
        <parameter>
          <component_value component="_3"/>
        </parameter>
        <parameter>
          <component_value component="_4"/>
        </parameter>
        <parameter>
          <method ref="intersection">
            <parameter>
              <search_value ref="leftStreet" original_values="true"/>
            </parameter>
            <parameter>
              <search_value ref="rightStreet" original_values="true"/>
            </parameter>
          </method>
        </parameter>
      </method>
      <format_ref ref="format_offset_from_intersections"/>
    </result>
  </alt>
</def>
```

The browser view becomes clearer:

```
IntersectionOffset      :  FullStreetName^40 positiveRealNumber^0 LinearUnits^0 Bearing^0 From^0 FullStreetName^40 OptSeparator Zone^20
                           { @directed_offset($2, $3, $4, @intersection( ['#leftStreet'] , ['#rightStreet'] ) ) }
                        ;
```

J-9969

This gives us a Location result when this type of intersection is given. Notice that in the definition for IntersectionOffset, we have referenced an output format, "format_offset_from_intersections." This is necessary to support output of the new address form in geocodes. In the output_formats section, we add the following:

```
<format_definition name="format_offset_from_intersections">
  <field_value ref="PreDir" record="1.1"/>
  <field_value ref="PreType" record="1.1" pre_separator=" "/>
  <field_value ref="StreetName" record="1.1" pre_separator=" "/>
  <field_value ref="SufType" record="1.1" pre_separator=" "/>
  <field_value ref="SufDir" record="1.1" pre_separator=" "/>
  <component_value component="Distance" pre_separator=" "/>
  <component_value component="Units" pre_separator=" "/>
  <value xml:space="preserve"> bearing </value>
  <component_value component="Bearing"/>
  <value xml:space="preserve"> from </value>
  <field_value ref="PreDir" record="1.2" pre_separator=" "/>
  <field_value ref="PreType" record="1.2" pre_separator=" "/>
  <field_value ref="StreetName" record="1.2" pre_separator=" "/>
  <field_value ref="SufType" record="1.2" pre_separator=" "/>
  <field_value ref="SufDir" record="1.2" pre_separator=" "/>
  <field_value ref="City" record="1.1" pre_separator=", "/>
  <field_value ref="State" record="1.1" pre_separator=", "/>
  <field_value ref="ZIP" record="1.1" pre_separator=" "/>
</format_definition>
```

This completes adding support for this type of intersection address.

Appendix L: Adjusting Spatial Operators

The default style supports specific wording for offsets from locations, for example:

"100 meters bearing 90 from 380 New York St Redlands CA 92373"

Common usage may need to allow other forms of given offset:

"100 meters heading 90 from…"
"100 meters heading west of…"

To support these forms, we need to edit the **Bearing** and **From** elements in the Spatial Operators section.

To support the preposition "of" as a **From** element is straightforward:

```
<def name="From">
  <alt>from</alt>
  <alt>of</alt>
</def>
```

```
From          :  "from"
              |  "of"
              ;
```

The **Bearing** element must return a numeric result. This means that adding "heading" as another alternate like this will support "100 meters heading 90 from…".

```
|   "bearing" realNumber
    { $2 }
|   "heading" realNumber
    { $2 }
;
```

However, this will not support "100 meters heading west…", as even if you inserted "dir" into the grammar, it will not return a numeric value. To support "100 meters heading west…", we will decouple the Bearing element from the DirectedOffset element by defining a new element, OffsetDirection:

J-9969

```
<def name="OffsetDirection">
  <alt>
    <elt>bearing</elt>
    <elt ref="Bearing">
    </elt>
  </alt>
  <alt>
    <elt>heading</elt>
    <elt ref="Bearing">
    </elt>
  </alt>
  <alt>
    <elt ref="Bearing">
    </elt>
  </alt>
</def>
```

The browser view becomes the following:

```
OffsetDirection  :  "bearing" Bearing
                 |  "heading" Bearing
                 |  Bearing
                 ;
```

Then, we edit **DirectedOffset** and **Bearing**:

```
<def name="DirectedOffset">
  <alt>
    <elt ref="positiveRealNumber" weight="0"/>
    <elt ref="LinearUnits" weight="0"/>
    <elt ref="OffsetDirection" weight="0"/>
    <elt ref="From" weight="0"/>
    <elt ref="Location"/>
    <result tag="SpatialOperator">
      <method ref="directed_offset">
        <parameter>
          <component_value component="_1"/>
        </parameter>
        <parameter>
          <component_value component="_2"/>
        </parameter>
        <parameter>
          <component_value component="_3"/>
        </parameter>
        <parameter>
          <component_value component="_5"/>
        </parameter>
      </method>
      <format>
        <component_value component="Distance"/>
        <component_value component="Units" pre_separator=" "/>
        <value xml:space="preserve"> bearing </value>
        <component_value component="Bearing"/>
        <value xml:space="preserve"> from </value>
        <component_value component="Match_addr" record="1"/>
      </format>
    </result>
  </alt>
</def>
```

For **Bearing**, just remove the literal element "bearing" and adjust the component selection to "1" to pick up the compass angle:

```
</alt>
<alt>
  <elt ref="realNumber">
  </elt>
  <result>
    <component_value component="_1"/>
  </result>
</alt>
```

J-9969

We now have the definitions we need for the offset address forms required:

```
[-] Spatial Operators
SpatialOperator :  DirectedOffset
                ;
DirectedOffset  :  positiveRealNumber⁰ LinearUnits⁰ OffsetDirection⁰ From⁰ Location
                   { @directed_offset($1, $2, $3, $5) } format: { ['Distance'] ['Units'] " bearing " ['Bearing'] " from " ['Match_addr'] }
                ;
From            :  "from"
                |  "of"
                ;
OffsetDirection :  "bearing" Bearing
                |  "heading" Bearing
                |  Bearing
                ;
Bearing         :  north
                   {0.}
                |  "nne"
                   {22.5}
                |  northeast
                   {45.}
                |  "ene"
                   {67.5}
                |  east
                   {90.}
                |  "ese"
                   {112.5}
                |  southeast
                   {135.}
                |  "sse"
                   {157.5}
                |  south
                   {180.}
                |  "ssw"
                   {202.5}
                |  southwest
                   {225.}
                |  "wsw"
                   {247.5}
                |  west
                   {270.}
                |  "wnw"
                   {292.5}
                |  northwest
                   {315.}
                |  "nnw"
                   {337.5}
                |  realNumber
                   { $1 }
                ;
```