Using GIS to Assist with the Maintenance of the Election Office Street File (ELVIS)

SHAWNEE COUNTY
KANSAS

About me:

William Trimble

Shawnee County GIS

GIS Analyst
2018 - Present

True North Geographic Technologies, Inc
Murfeesboro, TN

GIS Analyst
2015 – 2018

Statewide NextGen 911 implementation.

Weakley County (TN) 911

GIS Specialist
2012 - 2015

County NextGen 911 system and general GIS tasks.

Can we use GIS to help QC the Election Office’s ELVIS street file list?

SHAWNEE COMMISSIONER OF ELECTIONS COUNTY

Election Street File (ELVIS)

Table

ELVISstreetfile

OBJECTID *	key_street_segment *	cde_street_dir_prefix	text_street_name	cde_street_type	cde_street_dir_suffix	text_low_street_addr	text_high_street_addr	ind_odd_even	text_low_unit_nbr	text_high_unit_nbr
1	619095	SE	101st	St	<Null>	300	1799	B	<Null>	<Null>
2	619096	SE	101st	St	<Null>	1800	2799	B	<Null>	<Null>
3	619097	SE	101st	St	<Null>	2800	6999	B	<Null>	<Null>
4	1022864	SE	101st	St	<Null>	7000	8299	B	<Null>	<Null>
5	619098	SW	101st	St	<Null>	4600	5999	B	<Null>	<Null>
6	619099	SW	101st	St	<Null>	9800	12399	B	<Null>	<Null>
7	619100	SW	101st	St	<Null>	12400	13999	B	<Null>	<Null>
8	619101	SE	102nd	St	<Null>	2000	2399	B	<Null>	<Null>
9	619102	SW	103rd	St	<Null>	1400	3999	B	<Null>	<Null>
10	619103	SW	103rd	St	<Null>	7000	8399	B	<Null>	<Null>

text_high_unit_nbr	precinct_name	text_city	text_zip5	zone_type_desc	ind_permanent
<Null>	0066.01	Wakarusa	66546		Y
<Null>	0066.01	Berryton	66409		Y
<Null>	0058.02	Berryton	66409		Y
<Null>	0058.02	Overbrook	66524		Y
<Null>	0065.01	Wakarusa	66546		Y
<Null>	0074.03	Auburn	66402		Y
<Null>	0074.02	Harveyville	66431		Y
<Null>	0066.01	Berryton	66409		Y
<Null>	0065.01	Wakarusa	66546		Y
<Null>	0073.01	Wakarusa	66546		Y
<Null>	0058.02	Berryton	66409		Y

Fields of interest to the project:

Segment key	*Key value*
Direction prefix/suffix Street name Street type Low address High address Even or odd	*Centerline Information*
Precinct Name	*Election Boundary*

Challenges:

No spatial reference.

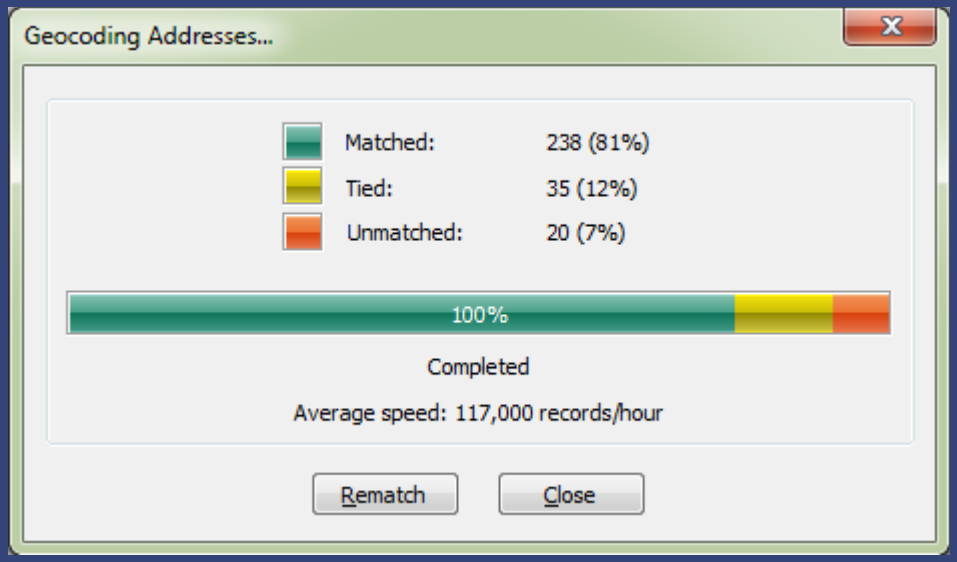
No key field relating to GIS data.

Ranges relate only to precinct boundaries.

Can we create useable results?

How do we get a spatial reference to the ELVIS data?

Geocode!



How much data do we want to process?

Consider the number of:

- Centerline Segments
- Elvis Entries
- Precincts

Initial Solution:

StartAddress	EndAddress	MiddleAddress
300 SE 101st St	1799 SE 101st St	1049 SE 101st St

Geocode 3 parts of each ELVIS entry…

Start & End (provided)
Middle (computed)


GO BIG
OR
GO HOME

Final Solution:

Geocode every possible address in the ELVIS street file list.

Shawnee County > 1.7 million addresses

Table

ELVISstreetfileExpanded

OBJECTID *	text_low_street_addr	ind_odd_even	precinct_name	address	streetID	FULLADDRESS
1	300	B	0066.01	SE 101st St	619095	300 SE 101st St
2	301	B	0066.01	SE 101st St	619095	301 SE 101st St
3	302	B	0066.01	SE 101st St	619095	302 SE 101st St
4	303	B	0066.01	SE 101st St	619095	303 SE 101st St
5	304	B	0066.01	SE 101st St	619095	304 SE 101st St
6	305	B	0066.01	SE 101st St	619095	305 SE 101st St
7	306	B	0066.01	SE 101st St	619095	306 SE 101st St
8	307	B	0066.01	SE 101st St	619095	307 SE 101st St
9	308	B	0066.01	SE 101st St	619095	308 SE 101st St
10	309	B	0066.01	SE 101st St	619095	309 SE 101st St
11	310	B	0066.01	SE 101st St	619095	310 SE 101st St
12	311	B	0066.01	SE 101st St	619095	311 SE 101st St
13	312	B	0066.01	SE 101st St	619095	312 SE 101st St
14	313	B	0066.01	SE 101st St	619095	313 SE 101st St
15	314	B	0066.01	SE 101st St	619095	314 SE 101st St
16	315	B	0066.01	SE 101st St	619095	315 SE 101st St
17	316	B	0066.01	SE 101st St	619095	316 SE 101st St

Create the list:

Two cursors – Search and Write

Search cursor to read through the original ELVIS table

Write cursor to create the new ELVIS expanded table

One cursor cant do both.

In python that looks like this:

```
### Iterate through original ELVIS data and create expanded data ###
print ("Create expanded ELVIS data...")
# create write cursor for expanded ELVIS data
expanded = arcpy.da.InsertCursor(ELVISstreetfileExpanded, ELVISfieldsExpanded)

# create search cursor for ELVIS data
with arcpy.da.SearchCursor(ELVISstreetfile, ELVISfields) as cursor:
    for row in cursor:
        print (row[0])
        # check side of road
        side = str(row[3])
        # populate even and odd points
        if side == 'B':
            startadd = int(row[1])
            endadd = int(row[2])
            while startadd <= endadd:
                expanded.insertRow((row[0],startadd,row[3],row[4],row[5]))
                startadd = startadd + 1
        # populate even points
        elif side == 'E':
            startadd = int(row[1])
            endadd = int(row[2])
            # check remainder to fix incorrect start value for range :/
            if (startadd % 2 != 0):
                startadd = startadd + 1
            while startadd <= endadd:
                expanded.insertRow((row[0],startadd,row[3],row[4],row[5]))
                startadd = startadd + 2
        # populate odd points
        else:
            startadd = int(row[1])
            endadd = int(row[2])
            # check remainder to fix incorrect start value for range :/
            if (startadd % 2 == 0):
                startadd = startadd + 1
            while startadd <= endadd:
                expanded.insertRow((row[0],startadd,row[3],row[4],row[5]))
                startadd = startadd + 2
del expanded
```

FULLADDRESS
300 SE 101st St
301 SE 101st St
302 SE 101st St
303 SE 101st St
304 SE 101st St
305 SE 101st St
306 SE 101st St
307 SE 101st St
308 SE 101st St
309 SE 101st St
310 SE 101st St
311 SE 101st St
312 SE 101st St
313 SE 101st St
314 SE 101st St
315 SE 101st St
316 SE 101st St
317 SE 101st St
318 SE 101st St
319 SE 101st St
320 SE 101st St
321 SE 101st St
322 SE 101st St
323 SE 101st St
324 SE 101st St
325 SE 101st St
326 SE 101st St
327 SE 101st St
328 SE 101st St
329 SE 101st St
330 SE 101st St
331 SE 101st St
332 SE 101st St
333 SE 101st St
334 SE 101st St
335 SE 101st St
336 SE 101st St
337 SE 101st St
338 SE 101st St

Calculate Field: FULLADDRESS

Create a single field with the complete address.

Use this new field for geocoding to the NG911 road centerlines.

New list is called ELVIS street file expanded.

Create a geolocator for the Kansas NG911 road centerlines.

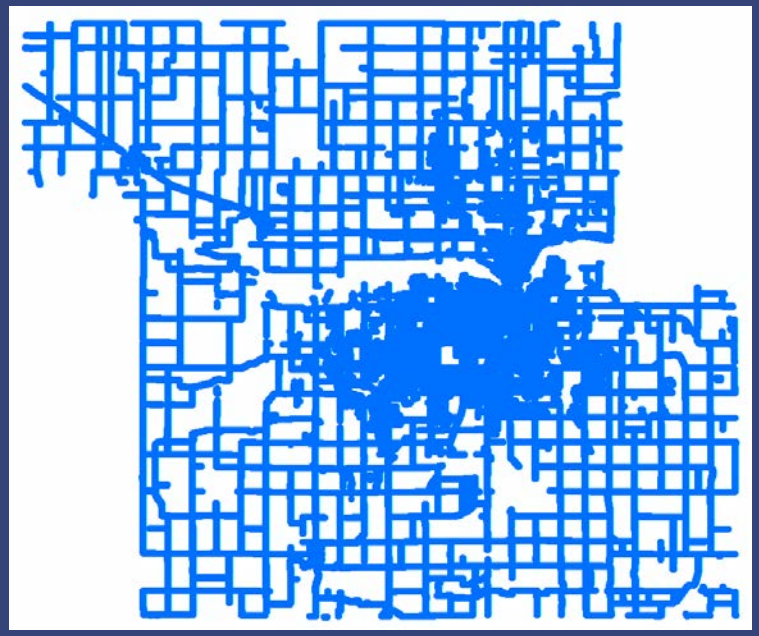
WHY?

Statewide standard that can be replicated in other counties across the state.

Geocode the ELVIS street file expanded data.

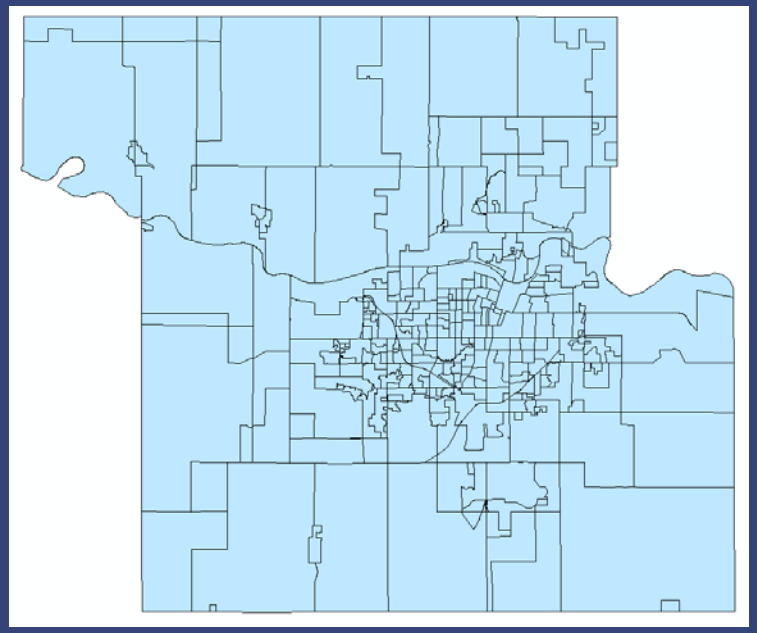
(and find something else to do for awhile)

Spatial Join:



ELVIS Expanded geocoding results

VS



Precinct boundaries

Two error sets:

Matched Errors

100% geocode score
ELVIS precinct <> precinct polygon

Unmatched Errors

Geocode score of 0
Also includes:
Ties, Geocode score <> 100%

Great, we have errors to report!

BUT

How do we turn 10s of thousands of point errors and table errors into useable data?

Problems:

It needs to be easy to interpret and use by a non-GIS professional.

There can be multiple errors on a road centerline or ELVIS segment. They can be E, O, B or a mix of the three.

How do you divide the errors and what format do you use to create a report?

Biggest breakthrough in brainstorming…

Assign a unique ID to each ELVIS street file entry.

The unique key also serves as a join field for the geocoding, error reports, and ELVIS.

```
## Matched Error Report
print ("Create matched error report...")
# variables
ELVISstreetfileErrors = "C:\\Elections\\ELVIS_Analysis\\Elvis_Analysis.gdb\\ELVISstreetfileErrors"
ELVISfieldsErrors = ["address","low_number", "high_number", "ind_odd_even", "precinct_name",
"key_street_segment"]
SpatialGeocodedELVISexpandedfields =
["USER_address","USER_text_low_street_addr","USER_ind_odd_even","USER_precinct_name", "USER_streetID"]
# row writer data variables
start = 0
errorstart = "0"
errorend = "0"
errorID = "0"
errorinterval = ""
errorstreet = ""
errorprecinct = ""
errortype = 0
current = ""

# create write cursor for ELVIS error data
errorwriter = arcpy.da.InsertCursor(ELVISstreetfileErrors, ELVISfieldsErrors)

# create search cursor for ELVIS expanded data
with arcpy.da.SearchCursor(SpatialGeocodedELVISexpandedErrors, SpatialGeocodedELVISexpandedfields) as
cursor2:
    for row2 in cursor2:
       # if first row, set variables to first row data
       if (start == 0):
           errorstart = row2[1]
           errorID = row2[4]
           errorinterval = row2[2]
           errorstreet = row2[0]
           errorprecinct = row2[3]
           start = 1
           current = row2[1]
```

```
else:
            # if in a B row and errors are only E or O, change interval to match
            if (errorinterval == "B") and (int(row2[1]) - int(current) == 2):
                if ((int(row2[1]) % 2) == 0):
                    errorinterval = "E"
                else:
                    errorinterval = "O"
            # if row matches start ID, do one of three below
            if (row2[4] == errorID):
                # both interval is not 1, write new row and set variables
                if (errorinterval == "B") and (int(row2[1]) - int(current) != 1):
                    errorend = current
                    errorwriter.insertRow((errorstreet,errorstart,errorend,errorinterval,errorprecinct,errorID))
                    errorstart = row2[1]
                    errorID = row2[4]
                    errorinterval = row2[2]
                    errorstreet = row2[0]
                    errorprecinct = row2[3]
                    current = row2[1]
                # even/odd interval is not 2, write new row and set variables
                elif (errorinterval == "E" or errorinterval == "O") and (int(row2[1]) - int(current) != 2):
                    errorend = current
                    errorwriter.insertRow((errorstreet,errorstart,errorend,errorinterval,errorprecinct,errorID))
                    errorstart = row2[1]
                    errorID = row2[4]
                    errorinterval = row2[2]
                    errorstreet = row2[0]
                    errorprecinct = row2[3]
                    current = row2[1]
                # continue to move through list
                else:
                    current = row2[1]
            # row doesnt match ID, write row and set new variables
            else:
                errorend = current
                errorwriter.insertRow((errorstreet,errorstart,errorend,errorinterval,errorprecinct,errorID))
                errorstart = row2[1]
                errorID = row2[4]
                errorinterval = row2[2]
                errorstreet = row2[0]
                errorprecinct = row2[3]
                current = row2[1]
    # write last row in dataset
    errorwriter.insertRow((errorstreet,errorstart,row2[1],errorinterval,errorprecinct,errorID))
```

The error output table:

Table

ELVISstreetfileErrors

OBJECTID *	address	low_number	high_number	ind_odd_even	precinct_name	key_street_segment *
6586	SE 105th St	6007	6237	O	0058.01	619106
6587	SE 10th Ave	295	299	O	0211.01	618205
6588	SE 10th Ave	561	565	O	0404.01	618312
6589	SE 10th Ave	1175	1199	O	0404.01	618312
6590	SE 10th Ave	600	656	E	0208.02	618313
6591	SE 10th Ave	2271	2355	O	0402.01	618317
6592	SW 10th Ave	1900	1998	E	0707.01	618325
6593	SW 10th Ave	1901	1919	O	0801.01	618326
6594	SW 10th Ave	3100	3110	E	0709.01	618330
6595	SW 10th Ave	3800	3899	B	0709.01	618330
6596	SW 10th Ave	4460	4598	E	1001.02	618331
6597	SW 10th Ave	4601	5099	O	1003.02	618332
6598	SW 10th Ave	5835	5899	O	1011.03	618335
6599	SW 10th Ave	5684	5698	E	1014.03	2384342
6600	SW 10th Ave	6500	6518	E	1291.01	618339
6601	SW 10th Ave	6542	6798	E	1291.01	618339
6602	SW 10th Ave	6501	6517	O	1281.01	618340
6603	SW 10th Ave	6795	6799	O	1219.01	618342
6604	SE 10th St	3101	3107	O	0401.02	618450

1 (0 out of 1046 Selected)

ELVISstreetfileErrors

Key_street_segment joins/searchable to matching street in ELVIS street file

We now have two error tables, matched errors and unmatched errors.

Strict on the matched error criteria…

Important note:

We are not an authority on the data so we did not want to make decisions for the reviewer.

Clean up the error features:

Points are busy and present too much information. They are also difficult to link errors to.

Convert to line features based on ELVIS unique ID field.


WITHOUT MAPS
WE ARE
SPATIALLY BLIND

Where and How?

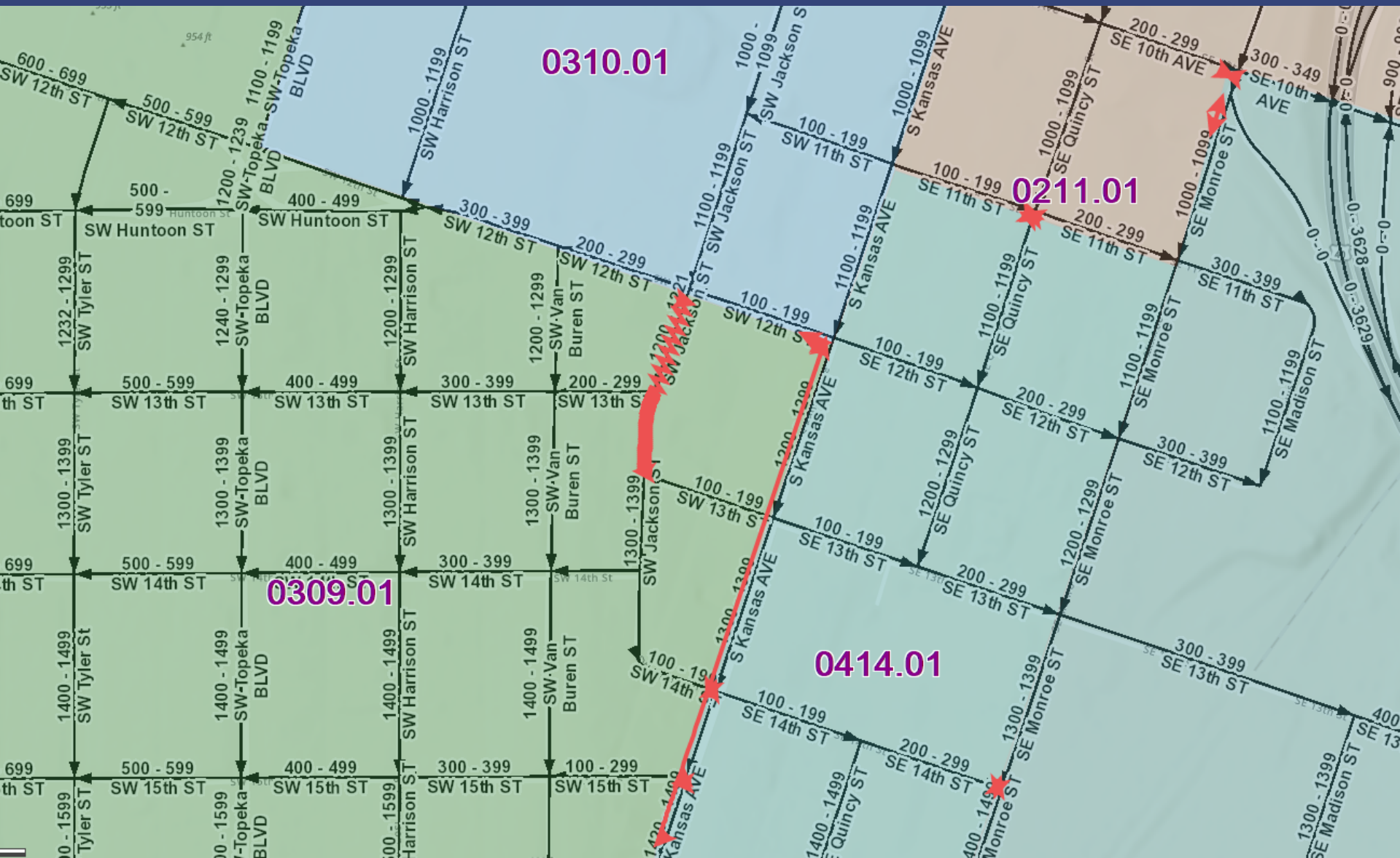
ArcGIS Online

No software required.

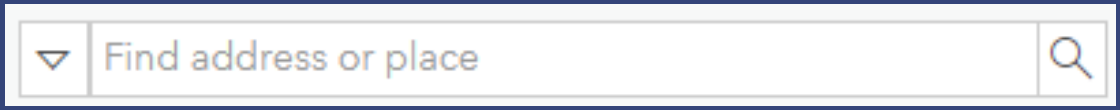
No knowledge of ArcMap or Pro needed.

Easy to share, search, and edit data.

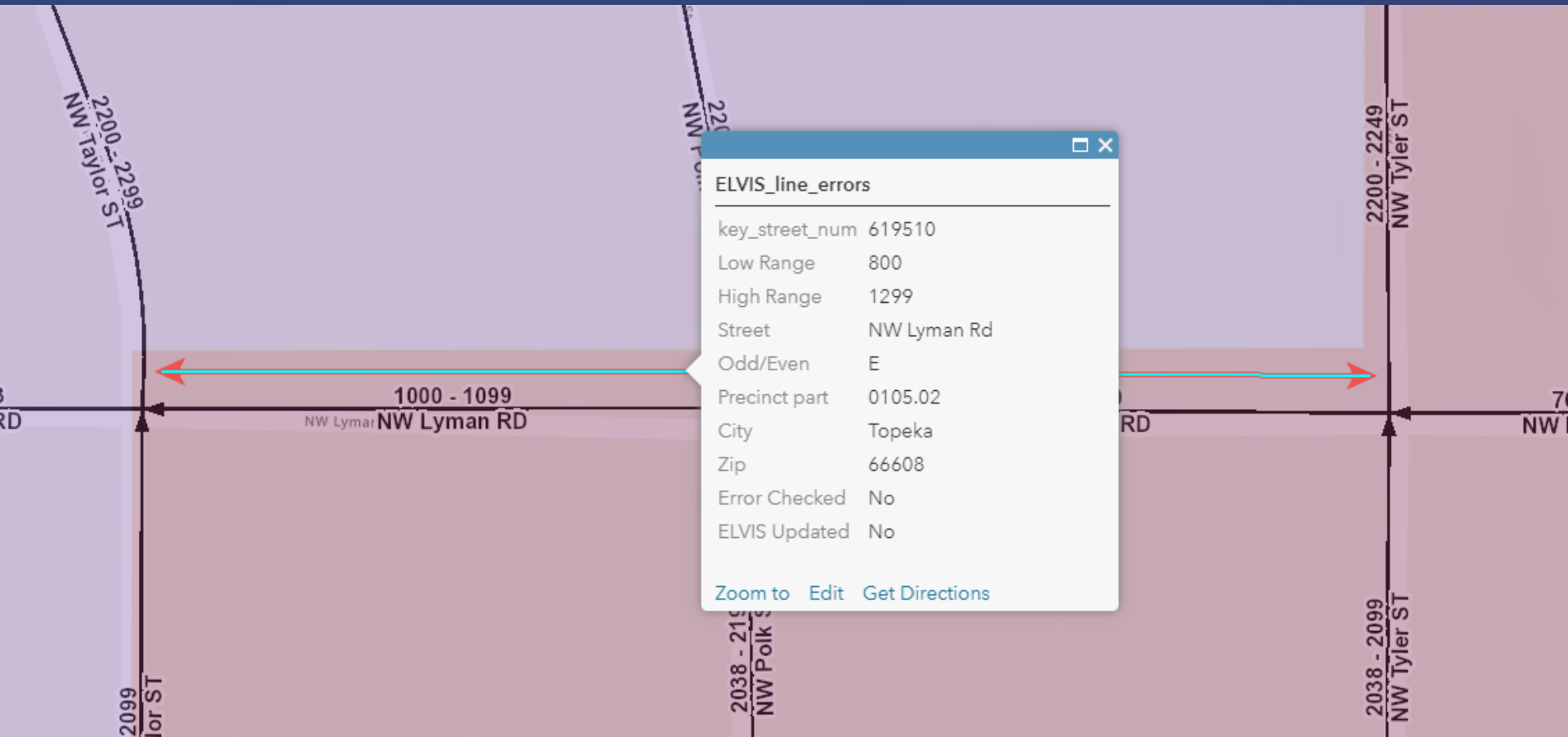
Map interface for the Election Office



Search by unique ELVIS street file ID



Popup window shows the original ELVIS information.

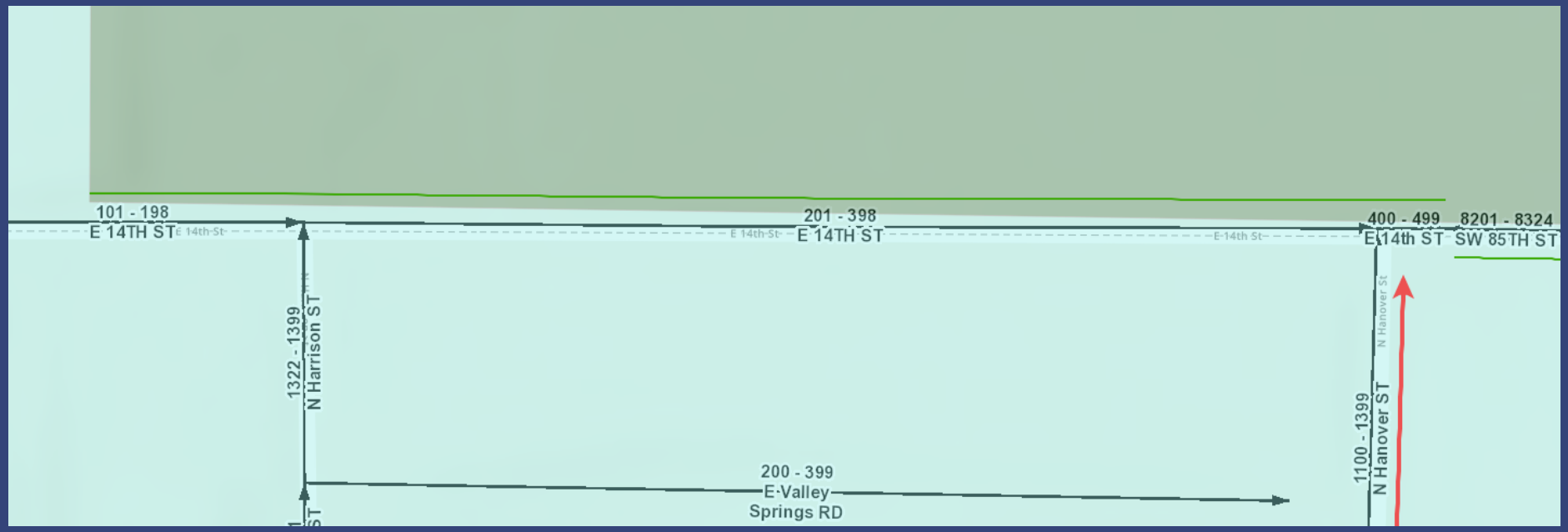


Types of errors:

- Precinct Slivers

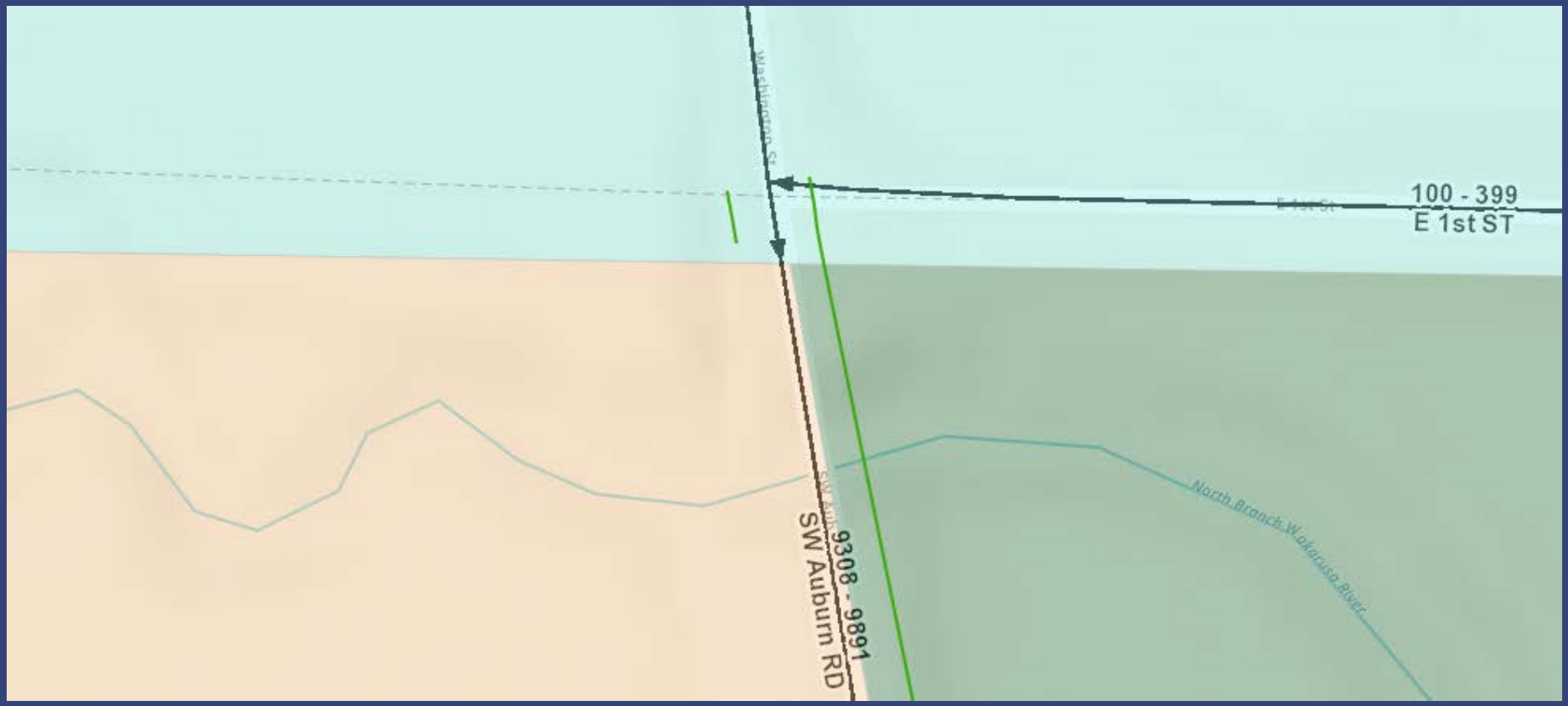
Types of errors:

- Reverse Slivers



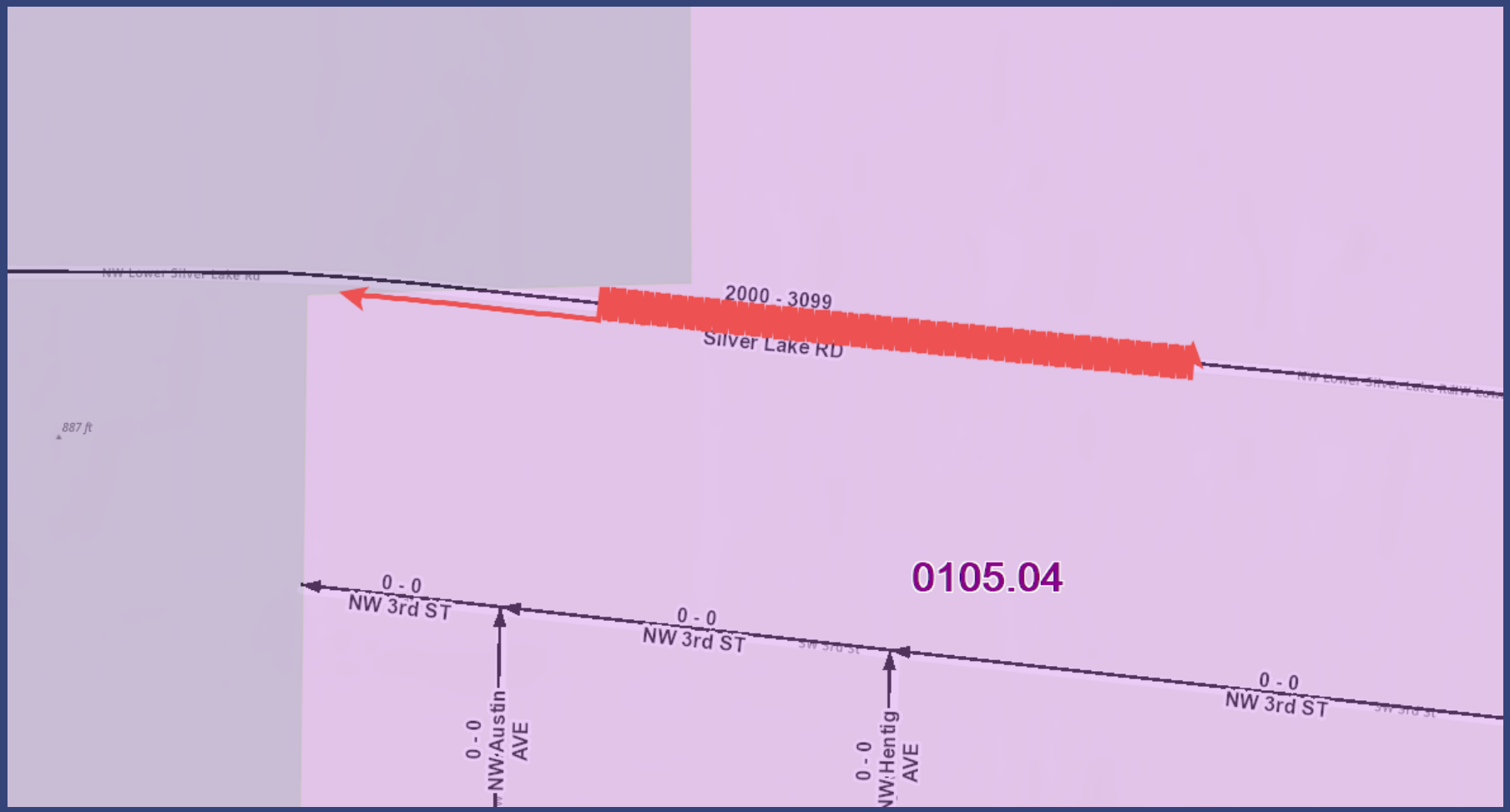
Types of errors:

- Missing Intersections



Types of errors:

- Incorrect Precinct Value



Future Plans:

- Adjust ArcGIS online error checking interface based on user feedback.

- Optimize some segments of the python code.

- Complete user input features for easy implementation at any county in KS.

    - user input file locations
    - convert local data into tables
    - NG911 based locator creation

Questions?

Want a copy of the python code?

william.trimble@snco.us