

# Geocoder Web Service

*A MetroGIS Project for funding year 2007*

## Final Report

November 26, 2008

For MetroGIS Coordinating Committee review

Submitted by Nancy Read, Metropolitan Mosquito Control District (MMCD)

### Summary:

Many web-based mapping applications use an address look-up (geocoder). In this project a group of MetroGIS participants identified a common need for a service that could take a request from an application and return a set of likely matching addresses and locations, using both address information in the Regional Parcel Dataset (and/or eventually the Address Points Dataset) and address ranges in the TLG Street Centerlines dataset.

After identifying requirements and sending out an RFP (through MMCD), the group chose to fund modifying the "Postal Address Geo-Coder" (PAGC, <http://www.pagcgeo.org/>), an open-source geocoding application used for batch geocoding. Walter Sinclair, developer of PAGC, made the extensive changes required and wrote documentation for installation and use of the service, and LMIC staff installed the service and related data. The service was then put into production use by projects at MMCD (see example in site at <http://www.mmcd.org/treatentrypage.htm>) MN-DNR, and the GCGI Emergency Preparedness committee (RNC application) and also tested by Carver Co. and Met. Council staff. After the first month some revisions and corrections were requested, which are now in place and documented.

The team worked with Metropolitan Council staff to set up an informational web site on the Geocoder, with links to the web service, general instructions, and full documentation (see <http://www.metrogis.org/data/apps/geocoder/>).

The service is fully functional for both street address and intersection look-up in the Metro area, and is in active use. It returns not only x,y coordinates (lat-lon) and a standardized situs address and mailing city, but also parcel ID (if a parcel match was found) (see web site for test form, or use in app at MMCD link above).

The Team has updated the street and parcel data used as reference by the geocoder and is working on automating those updates, aiming for weekly update of street data and (at least) quarterly update for parcels (parcel data update limited by pre-processing requirements at counties and Metropolitan Council).

Tools and examples are available to help in using the service, including a SOAP wrapper for .NET programming, and an ArcTools extension to use the service in a desktop mapping environment.

Presentations on the geocoder were made at MN GIS/LIS meetings in 2007 and 2008, and an article was published in the MN GIS/LIS newsletter.

We hope that other organizations needing address look-up will use the service or code, and save many hours of programming and data maintenance.

### Geocoder Team Members:

Jim Maxwell (TLG), Mark Kotz (Metro Council), Gordy Chinander (Metro Emergency Services Board), Bob Basques (City of St. Paul), Chris Cialek, Jim Dickerson, and Pete Olsen (LMIC), Dave Bitner (MAC), Kent Treichel (MN Dept. of Revenue) and Nancy Read (MMCD, contact for correspondence, [nancread@mmcd.org](mailto:nancread@mmcd.org), 651-643-8386).

**CONTENTS:**

**Introduction**

**Defining the Project**

**Project Steps**

**Examples of the Geocoder Service**

**Continuing Work**

**Lessons Learned**

**Recommendations**

**Appendix A: How the Geocoder Works**

**Appendix B: Plan and Specifications (June, 2007)**

**Appendix C: Data Pre-processing**

## **Introduction**

Many participants in MetroGIS, both governmental and private, are building web-based mapping applications to help citizens or staff find data related to an address. An address look-up (geocoder) is often the first step for access to these sites. A clear need existed for a service that could take a request from an application and return a set of likely matching addresses and locations, using both address information in the Regional Parcel Dataset (and/or eventually the Address Points Dataset) and address ranges in the TLG Street Centerlines dataset.

## **Defining the Project**

A group of MetroGIS participants identified the need for a Geocoder Web Service and recognized that we had most of the key ingredients needed to make it a reality:

- The project was technically possible (and there was at least one potential contractor willing to bid on it)
- The datasets were available - TLG was willing to have the street centerlines used in such an application, and it was within the acceptable uses of the Regional Parcel Dataset.
- A host site was available – LMIC agreed to host the service on their server.
- An organization had enough need for the resulting service that it was willing to take on project management (MMCD), and others saw enough value to be willing to serve on a team to guide the project.

What was needed was dedicated programming time to develop the web service. Funding from MetroGIS enabled the group to fill that need.

## **Project Steps**

1. Project proposal was submitted to MetroGIS Coordinating Committee and Policy Board, approved July 2007.
2. Geocoder team met and refined functional requirements and developed a “Scope of Work” that could be used as a basis for request for proposals (Appendix B). RFP was sent out and 6 proposals received. Team chose to fund a modification of the “Postal Address Geo-Coder” (PAGC), an open-source geocoding application originally developed for batch geocoding large research datasets.
3. Contract was finalized between Metropolitan Council and MMCD for funding Dec. 31, 2007.
4. A contract was established between MMCD and Walter Sinclair, developer of PAGC, and in negotiations it was agreed that the geocoder service would include both house address and intersection look-up, as well as documentation and some maintenance procedures, for \$14,000. In the first deliverable, 30 d after the contract started, Walter evaluated alternatives for the project and presented a plan to the team. He then proceeded with development of the application. At 3 months the team again met by teleconference with Walter to discuss any issues that had arisen during development.
5. By the end of May, 2008, Walter set up a demonstration of the web service hosted on his own server, and the team began testing and discussing bug fixes and other

revisions. Walter then developed documentation for installation and use of the service, and LMIC staff installed the service and related data in June. The service was then put into production use by projects at MMCD and MN-DNR and also tested by Carver Co. and Met. Council staff.

6. After this production testing, the team met with Walter again for a review. Several small but important items were identified by the team as high priority for revision. Some of these were considered to be in fulfillment of the original specifications, but some were items the team had not originally identified but recognized as important once the application was in full use. The team decided to extend the contract to include these changes.
7. The team also discovered several issues with the source datasets that needed to be resolved in order for the geocoder to work correctly. Fixes were applied that made the datasets usable, and team members began assessing longer-term solutions to prevent these problems (see Appendix C).
8. By early October, 2008, the revised Geocoder software was installed at LMIC, along with revised source datasets.
9. The team worked with Metropolitan Council staff to set up an informational web site on the Geocoder, with links to the web service, general instructions, full documentation, and the complete source code (available under the LGNU License).
10. Presentations on the geocoder were made at MN GIS/LIS meetings in 2007 and 2008, and an article was published in the MN GIS/LIS newsletter.

The service is fully functional and is in active use by several organizations.

## Examples Using the Geocoder Service:

The following screen shots show applications that are using the geocoder service.

### 1. In a public web site: Metropolitan Mosquito Control District Built by Brian Fischer, Houston Engineering, using PHP.

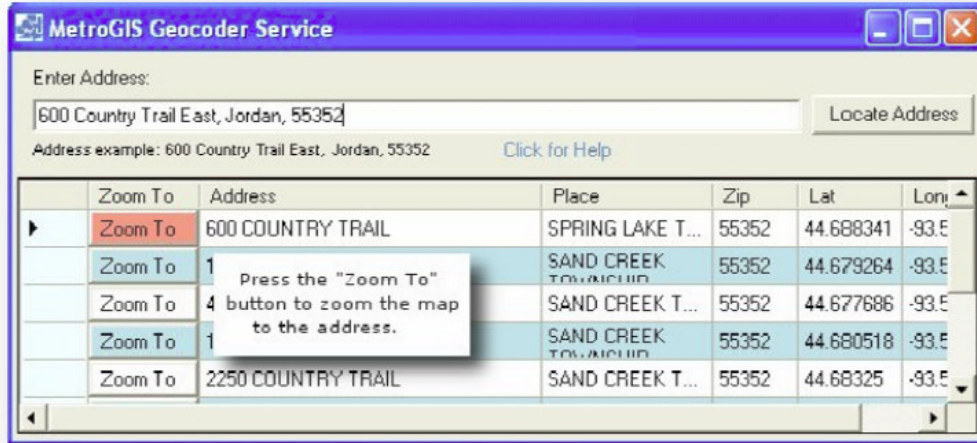
The screenshot displays the MMCD web application interface. The top navigation bar includes 'Home', 'Contact', and the date 'Saturday, November 29, 2008'. The main interface features a 'Zoom to Address' section with input fields for 'Address', 'City', and 'Zip Code', and a 'Find' button. Below this, there is an 'Intersection Geocoding' section with 'Street 1' and 'Street 2' input fields and another 'Find' button. A map of the Minneapolis area is shown, with labels for cities like ANOKA, HENNEPIN, RAMSEY, WASHINGTON, CARV, SCOTT, and DANOMA. A callout box points to the 'Address' field with the text 'Address Geocoding (for Metro, City-Zip optional)'. Another callout box points to the 'Street 1' and 'Street 2' fields with the text 'Intersection Geocoding'. At the bottom, an 'INFORMATION' table lists search results.

Address	Street	City	Zip
2099	University Avenue West	Saint Paul	55104
2099	University Avenue West	Saint Paul	55104
2099	University Avenue Northeast	Minneapolis	55418
2099	University Avenue Southeast	Minneapolis	55414
2019	University Avenue Northeast	Minneapolis	55418
2709	University Avenue Northeast	Minneapolis	55418
2609	University Avenue Southeast	Minneapolis	55414
2500	University Avenue	Minneapolis	55414

Example of Results

## 2. In an ArcMap Tool:

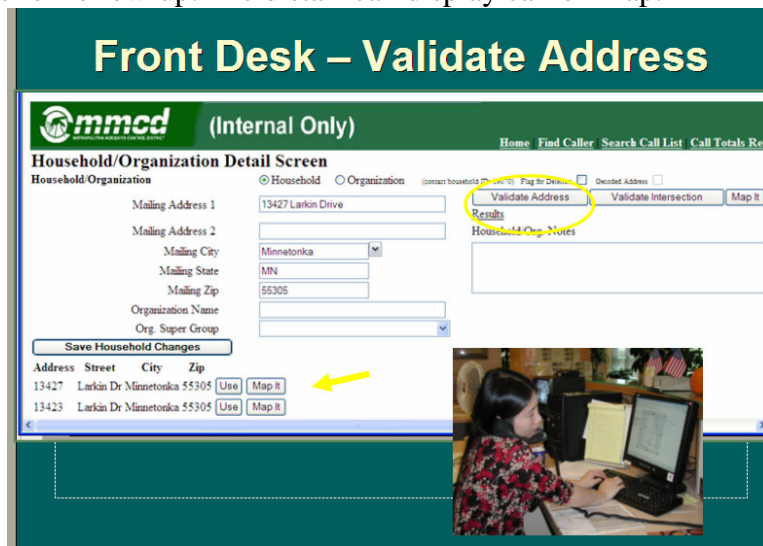
Tool was developed by Steve Jakala, Scott Co. to allow ArcMap users easy access to Geocoder service. Once installed, a user types in the address, hits "Locate Address", and 10 candidate results are returned. User can then zoom to location.



## 3. In a custom application: Mosquito Control District - Call Tracking System

As calls are received, front desk staff enters address and validates using the Geocoder, embedded in a data entry form. Both Situs City and Mailing City are used as returned by Geocoder if a Parcel match is found. (Mailing City not currently available for TLG Streets.)

MMCD's application uses geocoded location to spatially assign call to Facility and Foreman Areas for follow-up. Field staff can display call on map.



### **Continuing Work:**

1. Team members are working on automating procedures for updating the street and parcel datasets and pre-processing parcel data (Appendix C).
2. We are experimenting with the customizable settings in the Geocoder to adjust some aspects of performance.
3. LMIC's hosting is meeting current needs well and is not overburdening LMIC staff. However, if the service starts receiving very heavy traffic we may need to consider long-term hosting arrangements, and requirements for up-time.
4. City of St. Paul is working on a batch geocoding service using this code.

### **Lessons Learned:**

1. Contracts – It took over 4 months after the Policy Board approved the project before the contract between Metropolitan Council and MMCD was signed. The major issues were a. legal questions about licensing for Open Source software development, and b. availability of Met. Council legal staff to handle revisions. We hope that the language and principles developed for this contract can be re-used in future projects and reduce this time needed.
2. Working with a single contractor – When dealing with a single person on a contract, it is helpful to allow time for potential delays and/or plan alternatives in case the contractor is unavailable for some reason – for example, in this case the contractor had the flu and the project ended up delayed several weeks.
3. Web meetings and Teleconferences – This entire project was completed without the programmer ever coming to Minnesota to meet face-to-face with the team. While this may not be preferred, it is certainly workable, and it was also helpful to have screen-sharing over the web (as in WebX) as well.
4. Data quality – We participants of MetroGIS like to think our datasets are wonderful. When we start using them in applications, small discrepancies or problems may become evident. This is especially likely in data sets like the parcel data that come from many sources and are originally designed to meet in-house needs (see Appendix C). We need to be prepared to deal with the issues that come up regarding data quality and its implications for data providers and/or data custodians. Data sets may adhere to standards for format, but still have content issues that make it difficult to use them in applications.
5. Licensing – Other than the initial issues with the Met. Council contract, we encountered no problems related to licensing or intellectual property regarding this Open Source software development. This makes distribution very simple.
6. Encouraging Use – Although basic documentation of the service is complete, several potential users have asked for examples of calling the service from web applications. We will need to add some information or links on the Geocoder web site that can make it easy for web application developers to see how others have built in calls to the Geocoder.

7. Ongoing support – This application will benefit from having a web site where users can share what they’ve learned and where we can announce changes or get feedback from users. We would also like to establish a contact list of those who are using the service so we can notify users of any pending changes.
8. Project management – Our team recognizes it was important to have a dedicated project manager whose organization had a clear benefit from the project, and involvement of key players from other organizations with a wide range of relevant experience and (in some cases) a vested interest. The value of this “donated” time and expertise needs to be recognized when planning projects.

### **Recommendations**

1. Development of Web Services offers tremendous value to MetroGIS participants. It expands on the basic value of “build once, use many times”. Our experience with the Geocoder project we believe demonstrates the importance of this approach. We encourage MetroGIS to do more in this line, and appreciate support of extensions to the Geocoder such as the Points-of-Interest (Landmarks) development project.
2. Data content quality and automated data updates –MetroGIS and data producers will need to deal with these issues if we want to use this data as the base for high quality applications and services.
3. Licensing – Having Open Source licensing has made it easy to handle distribution, and does not seem to have caused any problems (except for some initial questions from Metro. Council’s legal department).
4. Hosting – This project would not have been possible without an organization willing to host the service. We appreciate LMIC’s contribution. Having hosting capability available will be a key component in expanding jointly-developed services.
5. Project “Commons” – This project currently uses the MetroGIS web site as its main information-sharing tool. It is becoming evident that we need a place for developers and users of a particular service to share news, tools, suggestions and questions. This will have to be further explored (especially in the context of an Open Source package that may be used anywhere in the world).

## Appendix A.

### **PAGC Postal Address Geocoder and the Geocoder Service – How It Works**

(summary extracted and interpreted Nov. 2008 by N. Read from 9/16/08 documentation produced by Walter Sinclair)

#### **Reference Data**

The Geocoder requires at least one shapset (.shp, .shx, .dbf) to use as a Reference. It can use both a precise dataset (points, such as parcel centroids) and a street dataset for interpolated location, finding the best match from either. For intersection geocoding only the street dataset is used.

To use reference data with the Geocoder, special data and index files must be “built” from the original shapefile. This build is only done when the geocoder is first set up, or if new reference data becomes available.

#### **Overview of “Build” Process**

The postal address related information is extracted from the Reference shapset's xbase (.dbf) attribute table, indexed, and standardized in preparation for matching against user inputs.

1. Schema
  - a. A Schema is used to set up how the address information is extracted (“Name Fields”).
  - b. Schema sets up parameters for how the reference information is to be compared with user inputs to find a match (“Comparison Types”), and the Match/Mismatch Weights (“M and U”).
  - c. Schema can also set
    - cross-street name source fields to use for intersection look-up
    - coordinate source field names, if you wish to use of coordinates directly from the xbase file instead of from the shapefile
    - occupancy fields (e.g., Apt #) to include in standardized record without comparing to user inputs
    - other flags for options such as generating statistics, how to read shape file, etc.

2. Standardization

Address records use a variety of abbreviations and can sometimes have the same information in different order (e.g., West Lake St., Lake St. West). Standardizing is a way to evaluate words, phrases and abbreviations so when user input is compared with the reference dataset appropriate parts are compared. Applying the same standardization to the reference file and user input improves matching.

- a. The standardizer “tokenizes” its input by describing words and numbers as a series of Input tokens (e.g., Number, Word, Direct, Type)
- b. The Lexicon and Gazetteer helps translate a particular input string into possible standardized text options, based on the token type (example: ST as token “TYPE” standardizes as “Street”; ST as token “Stopword” standardizes as “Saint”)
- c. “Rules” are used to establish how a given set of input tokens may translate into postal attributes, and how likely that rule is to occur (rank). (example: Main St could tokenize as “Word” “Direct” which translates into the Postal Attributes “Street” “Sufdir”)
- d. Rules may have different characteristics for “MICRO” or “MACRO” attributes

- MICRO –House #, Streetname, Type and Direction – on a reference file of streets with address ranges, this would not change for either side of a street segment
  - MACRO –City, State, and Postal codes – might be different on different sides of a street segment in a reference file of streets.
- e. The standardizer produces possible tokenized “candidate standardizations” for each record and compares their ranks generated using the rules. A maximum of 6 candidates with the highest ranks are retained. These are compared with the original unstandardized data; if a postal attribute present in the original is missing in the standardized, or one not present in the original appears in the standardized version, the standardization candidate is downgraded. The best standardization, by final rank, becomes the record in the “normalized” main data file and links to the original shapeseq using the original shapeseq entity number. [?? is this right??]
3. Files produced in “Build” (all in Berkeley b-tree format)
- a. Main data file (“normalized” data record, with original shapeseq entity number) (.pgx)
  - b. Shape information (x-y for a point, pair of x-y’s for a line) with original shapeseq entity number (.ix5)
  - c. Index files as follows (each returns lookup name of record in Main normalized data file)
    - full street name (.ix0)
    - root street name (defined as name without direction or type) (.ix1)
    - soundex of root street name (using standard soundex encoding) (.ix2)
    - approximate street name, using root and maximum edit distance (a paging trie that uses Berkeley memory pool facility) (.ix3)
    - point index for start and end points of block (optional) (.ix4)
    - concatenated intersection names (optional, for intersection lookup) (.ix6)
    - soundex of concatenated intersection names (optional, for intersection lookup) (.ix7)
    - approximate concatenated intersection names (optional, for intersection lookup) (.ix8)

### How the Geocoder Service Works

The following is a brief description of how the Geocoder does its work. For more details consult the c source code.

#### Initialization

In CGI mode the responder is re-launched by the Apache httpd webserver as each new request is received. In FastCGI mode it is launched by Apache when Apache starts and stays running, waiting for requests. Requests are relayed to **geocode\_response** by the fastcgi module, mod\_fastcgi, as they are received.

The program initializes by establishing where it is (the current working directory) and where its data files are located. Once it has possession of this information it opens the **PAGC** library and creates a **PAGC** schema record for each of the parcels and the streets data sets, opening the database files and indices. In CGI the responder will then create a **PAGC** matching context. In FastCGI it creates a number of matching contexts, each of which awaits a request.

### **Request handling – validate, concatenate**

When a request arrives, the variable-value pairs are retrieved. (See “Geocode Request API” for description of parameters).

Each is checked against the appropriate constraints to ensure that this value is a valid value for this variable.

Then the address data is concatenated into a form suitable for standardization and the results dispatched, bound to the matching context (e.g., precise or interpolated dataset), to **PAGC**.

### **PAGC processes**

#### **Standardize query (with Rules) –**

PAGC standardizes the query address strings, producing up to 5 different standardizations.

Each standardization, starting with the most likely (according to weights assigned to the rules), is used to produce index lookup keys for the query.

The kinds and number of keys produced will vary depending on whether this is an intersection or site address query.

#### **Search for Match (Candidates)**

A site address query looks for:

1. exact match on the complete streetname.
2. exact match on the base string street name (i.e., without directionals, type or modifiers)
3. approximate matches, within an edit distance of 2 (within 2 deletions, insertions or transpositions) of the base street name
4. key created from the soundex keys of each (non-numeric) word in the base street name

For intersections:

The same sequence of name, approximate name and soundex key searches is conducted.

1. the streets environment contains indices formed from a concatenation of the base street name of the record and the base street name of the cross-street; it searches these indices first.
2. if no candidates at all are found, it retrieves records matching one street name and those matching the other and joins them based on their coordinates.

With each index lookup a standardized address record is retrieved that serves as a candidate for matching.

#### **Calculate Candidate’s Score**

For each candidate, the query address and candidate address are compared, part by part, for a match.

- Each part of the address, if it matches, contributes a positive weighted value.
- If it doesn't match it contributes a negative weighted value.
- For some fields a similarity value may be calculated that results in a weight between the match and non-match value

The sum of these values constitutes the candidate's score. (see M and U scores)

#### **List**

The scored candidate is then placed in score order on the matching context's candidate list.

If the list is full the candidate will displace another candidate if its score is greater or equal to the score of the last candidate on the list. Otherwise it is chucked.

If at any point in the candidate generation process, a candidate is found that has the maximum possible score, the search is terminated and that candidate is returned.

However, except in that circumstance, the search continues until it has a list of the top 100 candidates for the matching context.

At this point control returns from **PAGC** to the geocoder.

### **Responder Merged List**

The responder now creates its own candidate list summoning **PAGC** on each of the highest scoring standardizations to geocode the address or intersection.

The candidates are scored and stored purely on the basis of the correspondence between their addresses and the query address.

- The streets database consists of blockrange records. It may be the fact that a candidate address, representing a blockrange, scores well enough on other attributes to make it onto the candidate list, but the query address number does not, in fact, fall into the interval given by the blockrange. An address that is non-geocodable in this manner is not added to the responder's list.
- (any equivalent for parcels?)

The score of each candidate that is kept is **normalized** by subtracting the lowest possible value and dividing by the difference between the highest and lowest possible value (determined by the sum of the M or U weights for all attributes in the schema), giving a value between 0.00 (least likely) and 1.00 (most likely) and is formatted according to the format specified by the request.

The responder retains the top 30 candidates (or max as set in request).

For site address:

- **PAGC** matching context is first bound to the parcels schema and looks for a "precise" match.
- responder evaluates if the top candidate fails to reach or exceed a certain score (this is cascading trigger; score is set in `data_cap.h` header, e.g., `#define ACCEPTABLE_SCORE .9 (default)`)
- if score not met or exceeded, responder rebinds the context to the streets schema record and looks for an "interpolated" match, once again summoning **PAGC** to produce candidates.
- The products of this are sorted into the geocoder's candidate list. (Note - because the schema of parcels and streets is slightly different, the maximum and minimum scores can be different, thus requiring this normalization)

The procedure for an intersection address is performed in a like manner, but using only the streets (linear) data (without cascading from the parcel point data).

### **Final response formatted, assembled**

When the responder has candidate list in its possession each candidate will have been formatted, geocoded and scored.

The list is then combined into the appropriate geocode list format.

That list is combined with the other elements of the response – the original requested address, the response header and a list of faults, if any – and the response is returned (via Apache and `mod_cgi` or `mod_fastcgi`) to the user-agent that generated the request.

## Appendix B.

### **MetroGIS Geocoder Project Outline for Coordinating Committee Review 6/27/2007**

Project Participants: Dave Bitner (MAC), Nancy Read (MMCD), Mark Kotz (Met.Co.), Jim Maxwell (TLG), Gordy Chinander (MESB), Chris Cialek & Jim Dickerson (LMIC), Bob Basques (St. Paul), Kent Treichel (MN Dept. of Revenue).

Focus of project:

1. Develop geocoding software that meets the following requirements:
  - Parse: take a given "initial address" character string and transform that into something that can be used to search against a database
  - Geocoding Engine: search a database (streets, parcels, or some other locational db) and return a list of lat/lon coordinates (point) of possible matches, and estimate of quality of match
  - Cascade: if Engine can't find a match in primary dataset, search next, etc. Priority and number of datasets searched should be configurable. Data returned on quality of match should indicate which dataset used for match.
  - Database "template" needs to match Geocoding Engine toolset; original data could be shapefile or PostGRE/GIS or some other data format.
2. Set up the above software on a host site with associated data and any supporting software such that geocoding can be provided as a web service for individual requests from other web applications.

Scope and Design issues:

1. Start with single requests, not batch.
  - a. Software could be used in-house by participants to do in-house batch geocoding against datasets they are already licensed to have.
  - b. a batch geocoding service (free OR charge) could be set up by a participant, depending on licensing issues.
2. Final product is web service that returns initial address string, parsed corrected address(es), lat/lon coordinates, and match quality info.
  - a. It is up to the developers of the web sites consuming this service to handle translation from lat/lon to other coordinate systems (including custom systems like King Map Book or systems like Military Grid), to handle match options and match quality display. If there are sufficient resources, code samples for doing these chores could be included, or may consider adding the most common conversion (UTM) to service.
  - b. Returned data format should reflect industry standards for geocoding services (e.g., standard schemas for XML transfer).
  - c. setting up a mapping site directly usable by the public is not within scope of this project.
3. The corrected addresses (text) returned could meet some national standard... [?]
4. Geocoder engine could use any dataset with US-style address. As part of project we plan to make data templates more specific to locally-available data: TLG streets, Metro Parcels, and eventually Occupiable Units. We plan to launch the web service using TLG streets and Metro Parcels.
5. Prefer that all parts of software are freely available/sharable, include comments in code, and documentation for anyone to install and use.

6. The complete process of submitting an initial address string, parsing, running geocoder engine, and returning list of matches should have a fast response time.
7. Software design should recognize potential future needs for enhancements, including intersection look-up and reverse geocoding (lat/lon to address).

**Total \$ Amount requested:** Not to exceed **\$14,000**.

Activity	
1. <b>define functional requirements</b> of a geocoding service for the MetroGIS community, scope of current project and develop RFP's	- to be done by team
2. <b>develop parsing code and geocoder engine</b> - evaluate existing geocoding code offered by MAC or available from other sources, assess changes needed to meet MetroGIS community needs, and use funding for programming to make those changes and/or develop new code as needed.	- RFP #1a - \$10,000 We expect to hire a consulting firm that can coordinate the evaluation of existing resources, with review by the group, and can perform or subcontract programming, possibly including code contributions from group members.
3. develop <b>documentation</b> for those planning to build applications that use the service or those wishing to use the geocoder code, either in open-source or ArcIMS environments	- RFP #1b - \$1000 (expect to be done with 1a)
4. define draft roles and responsibilities of "regional custodian" of service (the host organization) as well as source data providers (e.g. parcels & TLG)	- to be done by team and prospective host(s), as details of needs become clearer
5. find an organization willing to <b>host</b> the service and set up service on their server	- LMIC has offered to host. Probably no charge; will need to know what assumptions are made about host environment. Could also do as RFP #2, in which case would need another ca. \$1000. May also consider a multi-node setup, especially since some organizations may want to attach their own data to the address points for querying. This could also providing a means to load-balance.
6. <b>maintenance procedures</b> for TLG street data and other data used, such as translating to template form, rebuilding indexes, conforming to standards (Av vs Ave etc).	- Possibly RFP #3 - \$1000? Will need to determine with host and data providers. Some existing code from City of Saint Paul might be used.
7. add street <b>intersection</b> look-up 8. add <b>landmark</b> look-up	- add-on to RFP #1 - \$1000 Could start with existing intersection code for TLG dataset from City of St. Paul. Note that if code base is relatively generic, would make the end product much more valuable overall. Landmark lookup is one type of datasource, but there are many others. Not much work to increase the return on investment.

MMCD has agreed to serve as administrator as needed for handling funding.

## Appendix C.

### Data Pre-processing

#### Conversion to Lat-Long

Because the Geocoder service is designed to work with many different users, we chose to return coordinates in lat-long instead of a particular projection or coordinate system. This leaves it to the user to convert to whatever local system they are using (or add a national reference such as the National Grid). Thus the input data is translated to lat-long for use within the Geocoder. (Note that if a Coordinate Transformation Service was available, results from the Geocoder could easily be sent to that service to return other values.)

For the TLG Street file, this conversion is the only pre-processing needed.

#### Parcel Data Adjustments

The Metro Geocoder currently uses the “parcels\_all7\_points” file produced quarterly by Metropolitan Council staff using data provided by the 7 Metro Counties. We discovered that some aspects of this data result in challenging issues for the geocoder.

##### **1. City and City\_USPS names not consistent**

- mix of “St.”, “ST”, or “Saint”, more than one spelling for the same City
- Washington Co. has “City of” or “Town of” + Name in City field
- Scott Co. adds “CITY” to all City Names (e.g., “SAVAGE CITY”)
- Dakota and Scott Co. add “TWP” to all Township names; other counties spell out “TOWNSHIP”

For the Geocoder to score matches correctly when a user enters a City, the content of the City field should be predictable and consistent. For some users it is important for the returned value of City names to be consistent as well.

To make the City names compatible with the MetroGIS-endorsed boundary file, we would need to convert all “ST” and “Saint” in City to “St.”, make all townships end with “Twp.”, and eliminate “City of” or “CITY” (none of the cities in the 7-county metro have “City” as part of their official name).

##### **2. Street Name, Type, Directional not parsed into appropriate fields**

Parsing is currently done by all counties except Hennepin. Having inconsistency in this data makes it more difficult for the geocoder to make an appropriate match. After discussions with Hennepin Co., a couple of geocoder team members worked out a script for parsing this information while retaining original spellings (using the “standardizer” in ArcMap would have applied abbreviations as well as parsing). This script would need to be run on the Hennepin Co. portion of the parcel data prior to applying any parcel data updates to the geocoder. As of this writing we are working out some issues in determining which words are street types and which are part of street name for some rarely-used potential types (such as Cove, Bend, Heights), dealing with compound Types (such as “STCT” for Street Court) and making sure the preprocessing matches the geocoder lexicon.

**3. House Numbers containing Letters or “1/2”**

The Oct. 17, 2008 parcels\_all7\_points file contains a grand total of 1,176 records out of 1,088,804 that are not integers. It would be simpler to set up matching in the geocoder if the integer and non-integer portions of House Number were separate. (Also note that non-integer house numbers are not recommended by the National Emergency Management Association, NENA)

234 records with House Numbers that contain a letter:

Circle Pines	228
Lino Lakes	1
Hastings	1
Lakeville	2 (end with “XX”)
Rosemount	1 (ends with “XX”)
Burnsville	1

942 records with House Numbers that contain “1/2”:

city	Total
APPLE VALLEY	1
BLOOMINGTON	2
BROOKLYN PARK	1
NEWPORT	1
DOUGLAS TWP	1
EAGAN	51
EAST BETHEL	1
EDINA	2
EXCELSIOR	1
FARMINGTON	1
HASTINGS	2
HOPKINS	12
LONG LAKE	3
MEDICINE LAKE	2
MENDOTA	
HEIGHTS	1
MET AIRPORT	3
MINNEAPOLIS	843
OSSEO	1
RICHFIELD	2
ROBBINSDALE	4
SOUTH ST PAUL	3
ST. LOUIS PARK	3
WEST ST PAUL	1

- 4. Assorted oddities and omissions** make it difficult for geocoder to function well.  
 Examples: 5301 E County Line N - omits “Road”  
 2475 Tournament Players Cir N – Street file says “Players”, Parcel file says “Plays”