Travel Model Two Development: Roadway Supply

Technical Paper

Metropolitan Transportation Commission with Parsons Brinckerhoff, Inc.

September 9, 2013

m:\development\travel model two\supply\deliverables\roadways\2013 09 09 release roadway network.docx
1 Overview

MTC is rebuilding the representation of supply in our travel model. When complete, the new representations of space, roadways, transit service, sidewalks, and bicycle ways will become part of the Travel Model Two modeling system. For an overview of the model design, please see the Travel Model Two: Strategic Supply Design technical paper¹.

This technical paper outlines the Travel Model Two roadway network. Specifically, it discusses the steps for building the network from the TeleAtlas North America (TANA) data set. The processes are implemented via a collection of Microsoft DOS batch files, Python scripts, and Cube Voyager scripts. This paper describes these processes in detail.

As shown in Figure 1 below, The Travel Model Two roadway network captures an order of magnitude more detail than the roadway network included in Travel Model One. This increased detail should support more accurate representations of vehicle movements in the travel model.

Figure 1 – Example Travel Model One and Travel Model Two Highway Network Resolution

¹http://analytics.mtc.ca.gov/foswiki/pub/Main/Documents/2012_08_24_RELEASE_Strategic_Design.pdf
2 Inputs

An automated process has been developed to create a Cube network, which is compatible with the Travel Model Two software, from the TeleAtlas (now owned by TomTom) North America database. Prior to executing the steps, the following inputs are required:

1. **ca_tana2011.gdb**: The California TeleAtlas North America 2011 all roads highway network. This geo-database contains all links and junctions, as well as their attributes. The feature classes `ca_jc` and `ca_nw` must be included, which are the California network nodes and links, respectively. Specific documentation on the network and attributes can be found in the MultiNet Shapefile Format Specification document provided with the network. This document is available from MTC upon request.

2. **abag.gdb**: Geo-database containing the feature class `ABAG_blockgroups_Dissolve`, which defines the MTC model area and is used to extract the TeleAtlas nodes and links for the MTC model area.

3. **TAZ and MAZ Zone Centroid Shapefiles**: County-specific travel analysis zone (TAZ) and micro-analysis zone (MAZ) centroid locations. Each county’s shapefile must be projected in NAD83 CA State Plane FIPS VI geometry and contain the following fields:
   a. **TAZ or MAZ** (depending on whether the zones are MAZs or TAZs): the zone number for each MAZ or TAZ. Zone numbering is sequential and follows the scheme described in Zone Numbering System below.
   b. **COUNTY**: the county number of the MAZ and TAZ centroids. This is used to renumber each MAZ and TAZ according to the scheme described in Zone Numbering System.
   c. **POINT_X**: the X coordinate (in State Plane feet) of the centroid
   d. **POINT_Y**: the Y coordinate (in State Plane feet) of the centroid
3 Zone Numbering System

The highway network uses a numbering system whereby each county has a reserved block of nodes. Within each county’s block, nodes 1 through 9,999 are reserved for TAZs, 10,001 through 89,999 are for MAZs, and 90,001 through 99,999 for transit access points or TAPs. The blocks are assigned to the nine counties per MTC’s numbering scheme, as shown in Table 1. TeleAtlas network nodes are numbered by county as well and range from 1,000,000 to 10,000,000 as shown in Table 2. In Table 2, HOV lane nodes are those nodes corresponding to general purpose lane nodes.

Table 1 - County Zone Numbering System

<table>
<thead>
<tr>
<th>County</th>
<th>Reserved TAZ</th>
<th>Reserved MAZ</th>
<th>Reserved TAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 San Francisco</td>
<td>1 – 9,999</td>
<td>10,001 – 89,999</td>
<td>90,001 – 99,999</td>
</tr>
<tr>
<td>2 San Mateo</td>
<td>100,001 – 109,999</td>
<td>110,001 – 189,999</td>
<td>190,001 – 199,999</td>
</tr>
<tr>
<td>3 Santa Clara</td>
<td>200,001 – 209,999</td>
<td>210,001 – 289,999</td>
<td>290,001 – 299,999</td>
</tr>
<tr>
<td>4 Alameda</td>
<td>300,001 – 309,999</td>
<td>310,001 – 389,999</td>
<td>390,001 – 399,999</td>
</tr>
<tr>
<td>5 Contra Costa</td>
<td>400,001 – 409,999</td>
<td>410,001 – 489,999</td>
<td>490,001 – 499,999</td>
</tr>
<tr>
<td>6 Solano</td>
<td>500,001 – 509,999</td>
<td>510,001 – 589,999</td>
<td>590,001 – 599,999</td>
</tr>
<tr>
<td>7 Napa</td>
<td>600,001 – 609,999</td>
<td>610,001 – 689,999</td>
<td>690,001 – 699,999</td>
</tr>
<tr>
<td>8 Sonoma</td>
<td>700,001 – 709,999</td>
<td>710,001 – 789,999</td>
<td>790,001 – 799,999</td>
</tr>
<tr>
<td>9 Marin</td>
<td>800,001 – 809,999</td>
<td>810,001 – 889,999</td>
<td>890,001 – 899,999</td>
</tr>
</tbody>
</table>
Table 2 - County Network Node Numbering System

<table>
<thead>
<tr>
<th>County</th>
<th>Network Node</th>
<th>HOV Lane Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Francisco</td>
<td>1,000,000 - 1,500,000</td>
<td>5,500,000 - 6,000,000</td>
</tr>
<tr>
<td>San Mateo</td>
<td>1,500,000 - 2,000,000</td>
<td>6,000,000 - 6,500,000</td>
</tr>
<tr>
<td>Santa Clara</td>
<td>2,000,000 - 2,500,000</td>
<td>6,500,000 - 7,000,000</td>
</tr>
<tr>
<td>Alameda</td>
<td>2,500,000 - 3,000,000</td>
<td>7,000,000 - 7,500,000</td>
</tr>
<tr>
<td>Contra Costa</td>
<td>3,000,000 - 3,500,000</td>
<td>7,500,000 - 8,000,000</td>
</tr>
<tr>
<td>Solano</td>
<td>3,500,000 - 4,000,000</td>
<td>8,000,000 - 8,500,000</td>
</tr>
<tr>
<td>Napa</td>
<td>4,000,000 - 4,500,000</td>
<td>8,500,000 - 9,000,000</td>
</tr>
<tr>
<td>Sonoma</td>
<td>4,500,000 - 5,000,000</td>
<td>9,000,000 - 9,500,000</td>
</tr>
<tr>
<td>Marin</td>
<td>5,000,000 - 5,500,000</td>
<td>9,500,000 - 10,000,000</td>
</tr>
</tbody>
</table>
4 Build Scripts
The highway build process is performed by a series of Python (.py) and Cube Voyager scripts (.s). Each of the following scripts is described below in more detail:

1. mtc_networks_data.py
2. buildNetwork.s
3. mergeCentroids.py
4. mergeMazTaz.s
5. getTAZVertices.py
6. buildConnectorsMAZ.py
7. buildConnectors.s
8. buildConnectorsTAZAll.py
9. buildConnectorsTAZ_Part1.s
10. taz_usage.s
11. finalConnectors.py
12. buildConnectorsTAZ_Part2.s

In addition, a series of post-processing scripts are run after all of the network build processes (highway, transit, bike, pedestrian) have completed. These scripts are:

13. pre_postprocess.py
14. netToShapefileForPostprocess.s
15. postprocess.py
16. build_final_network.s

4.1 mtc_networks_data.py

*Purpose:* Creates highway network links and nodes shapefiles, re-numbers network nodes, and creates additional link and node attributes.

This script begins the network build process by creating feature classes for relevant links and nodes from the California statewide TeleAtlas 2011 network. It intersects the statewide “all roads” network links and nodes with the dissolved ABAG block groups feature class, ABAG_blockgroups_Dissolve (from abag.gdb) to obtain the MTC model area relevant links and nodes. It also re-numbers the TeleAtlas network nodes by trimming the first seven digits of the node ID number and adding an offset of three million. This offset helps identify TeleAtlas nodes from MAZs, TAZs, and TAPs. In addition, new fields
are created and calculated for the links and nodes. For links, the following new fields are created and assigned.

**Table 3 - New Link Attributes**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>F_JNCTID_h</td>
<td>Recode of the FROM node ID field, F_JNCTID</td>
<td>Long Integer</td>
</tr>
<tr>
<td>T_JNCTID_h</td>
<td>Recode of the TO node ID field, T_JNCTID</td>
<td>Long Integer</td>
</tr>
<tr>
<td>oneway_recode</td>
<td>Recode of the ONEWAY field for building the network late:</td>
<td>Short Integer</td>
</tr>
<tr>
<td></td>
<td>'FT' = 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>'TF' = -1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Blank or 'N' = 2</td>
<td></td>
</tr>
<tr>
<td>ASSIGNABLE</td>
<td>TRUE if FRC is not 7 or ONEWAY is not N</td>
<td>Short Integer</td>
</tr>
<tr>
<td>FEET</td>
<td>Conversion of the METERS field to feet</td>
<td>Double</td>
</tr>
</tbody>
</table>

For nodes, the following new fields are created and assigned.

**Table 4 - New Node Attributes**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>TeleAtlas node ID</td>
<td>Long Integer</td>
</tr>
<tr>
<td>ELEV</td>
<td>Elevation (in feet) of the node, to be assigned later</td>
<td>Long Integer</td>
</tr>
<tr>
<td>ID_hash</td>
<td>Recode of the ID field</td>
<td>Long Integer</td>
</tr>
<tr>
<td>FEATTYP</td>
<td>TeleAtlas feature type (junction or railway junction)</td>
<td>Short Integer</td>
</tr>
<tr>
<td>JNCTTYP</td>
<td>TeleAtlas junction type</td>
<td>Short Integer</td>
</tr>
</tbody>
</table>

Finally, the new feature classes are projected to NAD83 California State Plane FIPS VI and written to Shapefile for building a Cube network (performed in the following script).
**Inputs:** ca jc and ca nw from ca_tana2011.gdb
ABAG_blockgroups_Dissolve from abag.gdb
Intersect.gdb

**Outputs:** ca jc relevant_sp.shp
ca nw Intersect_sp.shp

**Required Python Modules:** ArcPy

### 4.2 buildNetwork.s

**Purpose:** Builds a Cube network from the link and node Shapefiles created in mtc_networks_data.py. This script executes the Cube `SHAPE2NETWORK` function and builds a highway network file from the link and nodes shapefiles created in mtc_networks_data.py. It specifies that the link’s from and to nodes be set to the renumbered from and to node fields (F_JNCTID_h and T_JNCTID_h, respectively) and specifies to use the `oneway_recode` field for specifying direction for auto travel.

**Inputs:** ca jc relevant_sp.shp
ca nw Intersect_sp.shp

**Outputs:** tana_sp.net

### 4.3 mergeCentroids.py

**Purpose:** Merges each county-specific MAZ and TAZ centroids Shapefiles into a single Shapefile to insert into the network. Adds a new field, N, and sets it to the renumbered zone number, according to the zone numbering scheme in Table 1.

This script locates all of the MAZ and TAZ centroids Shapefiles in the paths specified and merges the results to a new feature class. It begins by creating the new target feature class, using the first MAZ feature class found as a template. It adds a new field, N, to store the calculated zone number, and then merges all MAZs found into the new feature class. For MAZs the N field is then calculated to be:

\[ MAZ + (\text{COUNTY} - 1) \times \text{COUNTY_OFFSET} + \text{MAZ_OFFSET} \]
where MAZ and COUNTY are fields that must exist in each MAZ centroids shapefile. COUNTY_OFFSET and MAZ_OFFSET are global variables, set to 100,000 and 10,000, respectively. Similarly, TAZs are merged, and N is set to the following for TAZs:

\[ TAZ + (COUNTY - 1) \times COUNTY\_OFFSET \]

**Inputs:** MAZ and TAZ centroid Shapefiles for each county. Each MAZ Shapefile requires numeric MAZ and COUNTY fields, where COUNTY is set according to the county numbering scheme shown in Table 1.

**NOTE:** All input files must be in the same projection, specifically: NAD83 California State Plane FIPS VI.

**Outputs:** maz_taz_centroids.shp

**Required Python Modules:** ArcPy

### 4.4 mergeTazMaz.s

**Purpose:** Integrates the newly created centroids with the existing TeleAtlas highway network created in buildNetwork.s. Writes the results to a new network file and also writes out the links and nodes to separate DBFs for use in generating connectors in buildConnectorsMAZ.py.

**Inputs:** tana_sp.net

maz_taz_centroids.dbf (from maz_taz_centroids.shp)

**Outputs:** tana_sp_with_maz_taz_centroids.net
tana_sp_with_maz_taz_links.dbf
tana_sp_with_maz_taz_nodes.dbf

### 4.5 getTAZVertices.py

**Purpose:** Writes out the TAZ polygon vertices (i.e. shape points that make up the polygon) to a text file for use in building TAZ connectors.

**Inputs:** TAZ polygon Shapefiles for each county

**Outputs:** TAZVertices.csv - COUNTY,TAZ,X,Y fields

**Required Python Modules:** ArcPy
4.6 buildConnectorsMAZ.py

Purpose: Builds text file of MAZ connectors used as input by Cube to generate new links.

MAZ connectors are created by loading the TANA links and nodes DBFs and determining which highway network nodes are eligible for consideration as a connector end node. If the link is a freeway (FREEWAY = 1), a dedicated ferry link (FT = 0), or it is not assignable (ASSIGNABLE = 0), then the link’s A node is added to a set of highway nodes to exclude for consideration. Next, spatial indexes of highway nodes to consider are created for MAZs. All of the nodes are read in, and if COUNTY is 0, the node is added to the spatial indexes if it is not a node that was determined to be ineligible. Also, all of the MAZ nodes are read into a hash table for easily looking up values later.

Then, the hash table of MAZs is read and processed. The MAZ spatial index is queried, and the nearest two (configurable via the global variable NUM_NEAREST_NODES_MAZ) highway nodes are selected for connectors. By utilizing the local street network, MAZs only need to connect to the network at a couple locations. The MAZ connectors are written to a CSV file, along with the distance. The fields are A, B, distance. To make Cube recognize the connector as bidirectional, the connector is written twice, reversing A and B.

Inputs: tana_sp_with_maz_taz_links.dbf
tana_sp_with_maz_taz_nodes.dbf

Outputs: connectorsMAZ.csv, a CSV file with A, B, and distance columns containing two connectors for MAZs.

Required Python Modules: rtree, dbfpy

4.7 buildConnectorsTAZAll.py

Purpose: Builds text file of TAZ connectors used as input by Cube to generate new links.

The initial set of TAZ connectors are created by loading the TANA links and nodes DBFs and determining which highway network nodes are eligible for consideration as a connector end node. If the link is assignable (ASSIGNABLE = 1) and is a collector or major arterial (FT = 4 or FT = 7) then the link’s nodes will be considered for TAZ connectors. Next, the TAZ vertices are read from the CSV file and are assigned a nearest network node via spatial index. A spatial index of the nearest node of each
TAZ vertex is created for each TAZ. Also, all of the TAZ nodes are read into a hash table. All of the valid TAZ connectors are then written to a CSV file, along with the distance. The fields are A, B, distance. To make Cube recognize the connector as bidirectional, the connector is written twice, reversing A and B.

**Inputs:**
- tana_sp_with_maz_taz_links.dbf
- tana_sp_with_maz_taz_nodes.dbf
- TAZVertices.csv

**Outputs:**
- connectorsTAZ.csv, a CSV file with A, B, and distance columns containing four connectors for TAZs.

**Required Python Modules:** rtree, dbfpy

### 4.8 buildConnectorsTAZ_Part1.s

**Purpose:** Builds connectors for the TAZ nodes in the input network and creates a new network file with the connectors.

Sets the connector type field, CNTYPE to TAZ and makes the connectors assignable (sets ASSIGNABLE = 1) and then reads in the network with TAZ centroids and text file with (origin, destination, distance) for the connectors and merges them.

**Inputs:**
- tana_sp_with_maz_taz_centroids.net

**Outputs:**
- tana_sp_with_maz_taz_centroids_connectors.net

### 4.9 taz_usage.s

**Purpose:** Assigns a matrix of ones to the network in order to identify which TAZ connectors are most useful.

The script first re-numbers the TAZ nodes from 1 to num zones since Cube requires sequential zone numbers for assignment. Then it creates a matrix of 1s and assigns it based on link length (FEET). Next it deletes connectors which have no assigned volume.

**Inputs:**
- tana_sp_with_maz_taz_centroids_connectors.net

**Outputs:**
- tana_sp_with_maz_taz_centroids_connectors_assigned.net
4.10 finalConnectors.py

*Purpose:* Builds text file of the final best N TAZ connectors.

Loops through the connector links and builds a list of all connectors for each TAZ. Then sorts the list of connectors by assigned volume, and writes out the top eight (as configured by the `MAX_CONNECTORS` argument). The TAZ connectors are written to a CSV file, along with the distance. The fields are A, B, distance. To make Cube recognize the connector as bidirectional, the connector is written twice, reversing A and B.

*Inputs:*  
tana_sp_with_maz_taz_centroids_connectors_assigned_links.dbf  
tana_sp_with_maz_taz_centroids_connectors_assigned_nodes.dbf

*Outputs:* connectorsTAZFinal.csv, a CSV file with A, B, and distance columns containing four connectors for TAZs.

*Required Python Modules:* rtree, dbfpy

4.11 buildConnectorsTAZ_Part2.s

*Purpose:* Builds connectors for the TAZ nodes in the input network and creates a new network file with the final set of TAZ connectors.

Like buildConnectorsTAZ_Part2.s, reads in the TAZ connector links and sets the connector type field, CNTYPE to TAZ and makes the connectors assignable (sets ASSIGNABLE = 1). This script also then drops the computer generated Alameda County connectors and reads in the Alameda County connectors coded by MTC. The TAZ connectors are also written to a CSV file, along with the distance. The fields are A, B, distance. To make Cube recognize the connector as bidirectional, the connector is written twice, reversing A and B.

*Inputs:*  
tana_sp_with_maz_taz_centroids.net  
AlamedaConnectors.txt

*Outputs:*  
tana_sp_with_maz_taz_centroids_connectors.net  
finalConnectorsOldNumbers.txt
4.12  buildConnectors.s

*Purpose:* Builds connectors for the MAZ nodes in the input network and creates a new network file with the connectors.

Sets the connector type field, CNTYPE, to either MAZ or TANA, and makes the connectors assignable (sets ASSIGNABLE = 1) and then reads in the network with MAZ and TAZ centroids and text file with (origin, destination, distance) for the connectors and merges them.

*Inputs:* tana_sp_with_maz_taz_centroids_connectors.net

*Outputs:* tana_sp_with_maz_taz_centroids_connectors.net

4.13  runAllHighway.bat

*Purpose:* Executes all necessary scripts to build the network as a sequence of other batch files. The batch files, in order, are as follows:

1. runCreateShapeFiles.bat: **Executes** mtc_networks_data.py
2. runBuildNetwork.bat: **Executes** buildnetwork_sp.s
3. runMergeTazMaz.bat: **Executes** mergeCentroids.py, followed by mergeTazMaz.s
4. runBuildConnectors.bat: **Executes** getTAZVertics.py followed by buildConnectorsMAZ.py, and buildConnectorsTAZAll.py, followed by buildConnectorsTAZ_Part1.s, followed by taz_usage.s, followed by finalConnectors.py, followed by buildConnectorsTAZ_Part2.s, followed by buildConnectors.s

4.14  pre_postprocess.py

*Purpose:* Clean out any remnant files from an existing post-process run.

4.15  netToShapefileForPostprocess.s

*Purpose:* Builds link and node shapefiles from the existing Cube network.

*Inputs:* tana_sp_with_maz_taz_tap_centroids_connectors_osm_bike_routes.net

*Outputs:* postprocess_temp_link.shp
            postprocess_temp_node.shp
4.16 postprocess.py

*Purpose:* Renumbers nodes so that they are consistent with the numbering system described in *Zone Numbering System*. This script also adds non-bus transit links explicitly to the network, adjusts TAP nodes to be 25 feet (instead of 1 foot) from their primary connector node, and updates the transit line file so that it uses these updated nodes. Finally, the script writes the link and node input files for building the final network in Cube.

*Inputs:* postprocess_temp_link.shp

postprocess_temp_node.shp

*Outputs:* postprocess_link.csv

postprocess_node.csv

transitLines.lin

4.17 build_final_network.s

*Purpose:* Builds the final network in Cube, including transit network, and exports it to shapefiles.

*Inputs:* postprocess_link.csv

postprocess_node.csv

transitLines.lin

transitFactors.fac

*Outputs:* mtc_final_network.net

mtc_final_network_links.shp

mtc_final_network_nodes.shp
5 User Class, Toll, and Other Link Attributes

After building the TANA network in Cube, the existing Travel Model One user class (USER) link attributes and number of lanes were joined (conflated) to the TANA network. This process involved the following steps:

1. Select the existing Travel Model One links with special user class coding. These are links with the following attribution:
   a. USE = 2; HOV 2+
   b. USE = 3; HOV 3+
   c. USE = 4; No combination trucks

2. Manually identify the start and end TANA node of each special user class corridor (i.e. the Eastbound I-580 carpool lane). Also note the user class code, the start and end node X and Y, and the number of lanes to remove from the general purpose link and/or to add to the user class link. These are stored in the use_links.xlsx file.

3. Run the Cube shortest path routine to get the TANA node sequence from the start node to the end node for each corridor. This is done with the shortestPath_UseLinks.s script. This is the set of new links to add to the network. The script outputs use_links_nodes.csv.

4. Copy the new links to the use_links_nodes.xls file and join link attributes to create the new link records to add to the TANA network in Cube. The following files are then export from this Excel file:
   a. use_links_cube.csv – new user class links
      i. NEWA – new A node which is the general purpose node number + 2 million (which is later changed in the post-processing step to be consistent with the county node numbering system)
      ii. NEWB – new B node which is the general purpose node number + 2 million (which is later changed in the post-processing step to be consistent with the county node numbering system)
      iii. OFFAX – new offset A X coordinate
      iv. OFFAY – new offset A Y coordinate
      v. OFFBX – new offset B X coordinate
      vi. OFFBY – new offset B Y coordinate
vii. **USECLASS** – user class code  
viii. **FRC** – TANA Functional Road Class  
ix. **NEWLANES** - lanes  
x. **FEET** – link length  
xi. **SPEEDCAT** – TANA speed category  
xi. **KPH** – TANA KPH

b. *use_links_connectors_cube.csv* – new connector links offset by 50 ft from the general purpose links  
i. **CONNECTORA** – A node  
ii. **CONNECTORB** – B node  
iii. **CONNECTORA_REV** – A node for reverse link  
iv. **CONNECTORB_REV** – B node for reverse link  
v. **USECLASS** – user class code  
vi. **FRC** – TANA FRC  
vii. **NEWLANES** - lanes  
viii. **FEET** – link length  
ix. **SPEEDCAT** – TANA speed category  
x. **KPH** – TANA KPH  
xi. **CNTYPE** – ‘USE’

c. *use_links_cube_gp.csv* – revisions to the TANA general purpose link lanes  
i. **A** – A node  
ii. **B** – B node  

iii. **LANES** – revised lanes

5. The TANA network includes the number of lanes for significant facilities such as freeways, highways, and many major arterials. However, it does not include the number of lanes for all links with **LANES** greater than one. To remedy this, the *num_lanes.csv* file was created to update the TANA network where lane data was missing. This file includes the number of lanes for the identified TANA network links and it was created by spatially joining and then manually reviewing and adjusting the results. The file has the following fields: **A**, **B**, **LANES**.
6. Run `mergeUseLinks.s` to add the new links and to code various model specific attributes. The script does the following:

   a. Imports the new user class links with `USECLASS` set to 2, 3, or 4.
   b. Adds the user class link connectors. See Figure 2 below for an example of the new user class links and link connectors.
   c. Revises the number of lanes on the general purpose links as needed.
   d. Codes the link `TOLLBOOTH` attribute using the `Travel Model One` `TOLL` attribute values\(^2\).
   e. Codes link facility type (FT) based on the TANA `FRC` and `RAMP` to `Travel Model One` Facility Type crosswalk defined in Table 4 below. This will allow the link capacity to be calculated based on the `Travel Model One` link capacity lookup table\(^3\). See Table 5 below for a count of the network links by facility type.
   f. Codes the link free flow speed (FFS) in MPH based on the TANA KPH.
   g. Sets the `LANES` attribute for TANA network links in the `num_lanes.csv` file.
   h. Sets the `NUMLANES` attribute to `LANES` or to 1 if `LANES` is undefined. This allows for reviewing the different components of the `LANES` coding later if needed.

Table 5 – Facility Type Crosswalk

<table>
<thead>
<tr>
<th>FRC</th>
<th>FRC Description</th>
<th>RAMP</th>
<th>FT</th>
<th>FT Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>0</td>
<td>Connector (MAZ, TAZ, TAP, USE)</td>
</tr>
<tr>
<td>0,1,2</td>
<td>Freeway, Major Road, Other Major Road</td>
<td>T</td>
<td>1</td>
<td>Freeway to Freeway</td>
</tr>
<tr>
<td>0,1,2</td>
<td>Freeway, Major Road, Other Major Road</td>
<td>F</td>
<td>2</td>
<td>Freeway</td>
</tr>
<tr>
<td>3</td>
<td>Secondary Road</td>
<td>F</td>
<td>3</td>
<td>Expressway</td>
</tr>
<tr>
<td>6,7,8</td>
<td>Local Road, Insignificant Local Road, Other Road</td>
<td>F</td>
<td>4</td>
<td>Collector</td>
</tr>
<tr>
<td>&gt;2</td>
<td>Ramp</td>
<td>T</td>
<td>5</td>
<td>Ramp</td>
</tr>
<tr>
<td>4,5</td>
<td>Local Connecting Road, Significant Local Road</td>
<td>F</td>
<td>7</td>
<td>Major Arterial</td>
</tr>
</tbody>
</table>

\(^2\) [http://mtcgis.mtc.ca.gov/foswiki/Main/HighwayNetworkCoding](http://mtcgis.mtc.ca.gov/foswiki/Main/HighwayNetworkCoding)

\(^3\) The link capacity lookup table is based on facility type, area type, and signal coordination/ramp metering. Link area type, which is based on TAZ (or potentially MAZ) population and employment density, will be dynamically coded during the model run. In order to calculate a more even area type surface, the calculated TAZ (or potentially MAZ) area type will consider neighbor zones. Once area type is coded, each link will be attributed with its capacity. Adjustments to the link capacity due to signal coordination/ramp metering will be addressed when revising the model software.
Figure 2 – Example User Class Link Coding*

*User Class Links=Orange, User Class Connectors=Red, MAZ/TAZ Connectors=Grey, General Purpose/Other Links=Blue

Table 6 – Facility Type Link Count

<table>
<thead>
<tr>
<th>Facility Type (FT)</th>
<th>Description</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Connector (MAZ, TAZ, TAP, USE)</td>
<td>353,092</td>
</tr>
<tr>
<td>1</td>
<td>Freeway to Freeway</td>
<td>596</td>
</tr>
<tr>
<td>2</td>
<td>Freeway</td>
<td>15,312</td>
</tr>
<tr>
<td>3</td>
<td>Expressway</td>
<td>21,384</td>
</tr>
<tr>
<td>4</td>
<td>Collector</td>
<td>593,450</td>
</tr>
<tr>
<td>5</td>
<td>Ramp</td>
<td>2,582</td>
</tr>
<tr>
<td>7</td>
<td>Major Arterial</td>
<td>161,339</td>
</tr>
</tbody>
</table>
5.1  shortestPath_UseLinks.s

*Purpose:* Generate a shortest path cost file between special user class corridor start and end nodes.

*Inputs:* tana_sp_with_maz_taz_centroids_connectors.net

*Outputs:* use_links_nodes.csv, a CSV file with origin node, destination node, intermediate node, and cumulative cost

5.2  mergeUseLinks.s

*Purpose:* Merge in user class links, user class link connectors, code link facility type, free flow speed, toll booths, and lanes.

*Inputs:* tana_sp_with_maz_taz_centroids_connectors.net

*Outputs:* tana_sp_with_maz_taz_centroids_connectors_uselinks.net
6  Network Connectivity

Two network connectivity tests were performed to check the network. The first was to run the MAZ to MAZ shortest path script (shortestPath.s) to ensure it produced reasonable distances between MAZ pairs. The second test was to run the TAZ skimming script (MTC_NetworkSkim_TAZ.s) to create a TAZ distance skim matrix and a straight line distance matrix. This second script must renumber the TAZ nodes since Cube requires sequential zone numbers for skimming.

The MAZ to MAZ shortest path script produced distances for all MAZ pairs. To ensure the TAZ assignable network does not have any holes, the TAZ distance skim was created. The distance skim had no TAZ pairs (beyond intra-zonals) with an unconnected value, which confirms all TAZ pairs are connected. In addition, the network distances were compared with the straight line distances and no extreme outliers were identified. Many of the zone pairs had reasonable deviations between the network and straight line distances given the topology in the area. This is illustrated for TAZ 800013 to 800099 in Figure 3 below.

Figure 3 - Network Distance Review
6.1 shortestPath.s

Purpose: Generate a shortest path cost file between MAZ pairs within a specified cost threshold. The
MAXPATHCOST is set to maximum network distance to consider destinations MAZs. The ORIGIN and
DESTINATION parameters are currently set to consider only MAZ nodes. The script outputs only the cost
to the final node in the path, so the NODE field is always the same as the J field.

Inputs: tana_sp_with_maz_taz_centroids_connectors.net

Outputs: mazShortestPaths.csv, a CSV file with fields I,J,NODE,COST (FEET)

6.2 MTC_NetworkSkim_TAZ.s

Purpose: Create a TAZ level distance skim matrix and straight line distance matrix for checking the
network.

Inputs: tana_sp_with_maz_taz_centroids_connectors.net

Outputs: taz_dist.csv, a text file with fields I,J,DISTANCE

taz_skim_dist.csv, a text file with fields I,J,DISTANCE