

Working Paper 06-05

Rail Transit Coding Procedures for Arc/INFO

By Craig Heither
Associate for System Evaluation

December, 2006

Chicago Area Transportation Study
233 S. Wacker Dr., Suite 800, Chicago, Illinois, 60606

Working Paper 06-05

Rail Transit Coding Procedures for Arc/INFO

By Craig Heither
Associate for System Evaluation

December, 2006

Chicago Area Transportation Study
233 S. Wacker Drive, Suite 800
Chicago, IL 60606

TABLE OF CONTENTS

1	RAIL NETWORK DESCRIPTION	1
1.1	ARC AND NODE TABLES.....	1
1.2	RAIL TRANSIT TABLES.....	3
1.3	MAINTAINING TOPOLOGY	4
2	CODING RAIL ROUTES.....	5
3	EXPORTING RAIL DATA FOR EMME/2 NETWORKS.....	7

APPENDIX A: COVERAGE MAINTENANCE PROGRAMS

APPENDIX B: PROGRAMS TO IMPORT RAIL CODING

APPENDIX C: PROGRAMS TO EXPORT RAIL CODING

APPENDIX D: RAIL COVERAGE CREATION PROGRAMS

Introduction

This report documents the procedures used to code rail transit routes in an Arc/INFO[®] coverage. It also includes the procedures to export rail transit information from the coverage and use it to develop travel demand model networks for regional analyses. Performing these functions requires the use of ArcInfo[®], SAS[®] and EMME/2[®]. The reader is assumed to have some familiarity with CATS' model networks and the programs used to maintain them.

The rail transit coverage (named “railnet”) is stored on the C:\ drive of my computer in the **build_networks** directory. The ARC Macro Language (AML) programs and SAS programs used for processing and network development are located in the **build_networks\programs\rail** directory. These items are regularly backed-up to the P:\ drive on the CATS network to maintain secure copies. The rail transit coverage was created using all scenario information from the 2006 Conformity analysis. Appendix D briefly describes the steps involved in creating the coverage. The scenario structure for Conformity 2006 is:

Analysis <u>Year</u>	Numeric <u>Designation</u>
2002	100
2007	300
2010	400
2020	500
2030	600

1 Rail Network Description

Related data tables contained within the coverage structure provide all of the information needed to build networks for regional analyses. The four kinds of data tables maintained in railnet are: arcs (links), nodes, routes and sections. The use of route and section tables to store highway project information and bus route coding has been discussed in several previous working papers (for example, Working Paper 06-04: Master Highway Network Coding and Processing Procedures for the 20006 Conformity Analysis). The route system “rail_lines” contains all rail transit information. The following tables highlight the data table variables maintained in railnet. *Note:* Not all of ARC's internally maintained variables are listed; only those relevant to the discussion are included

1.1 Arc and Node tables

Table 1 lists the rail network link variables contained in the arc table. These represent the variables required by EMME/2 in a network batchin file. An additional variable (*aux_scenario*) was added to identify which scenarios include a specific auxiliary link (access, egress or transfer link) that is hard-coded into the coverage.

Table 1. Link Table Variables (railnet.aat)

Variable	Description
FNODE#	ARC generated unique identification number for "From" node.
TNODE#	ARC generated unique identification number for "To" node.
RAILNET#	ARC generated unique identification number for each link.
ANODE	CATS "From" node.
BNODE	CATS "To" node.
MILES	Link length in miles.
MODES1	Modes permitted on anode-bnode direction of link (mode letters).
MODES2	Modes permitted on bnode-anode direction of link (mode letters); blank if link is only 1 direction.
DIRECTIONS	Either one-way or two-way link.
LANES	Number of lanes; this does not have a real-world connection to driving lanes.
TYPE	This value was used to display different features when the coding was all stored in an EMME/2 bank. While not critical, the values were carried over from the databank.
AUX_SCENARIO	Text string comprised of concatenated hundred's place values from C06 scenario numbers, indicating which scenarios include the auxiliary link. This field only applies to access, egress and transfer links hard-coded into coverage and is blank in all other instances. Use CONTAINS (ArcInfo) or LIKE (ArcGIS) to select in a query.

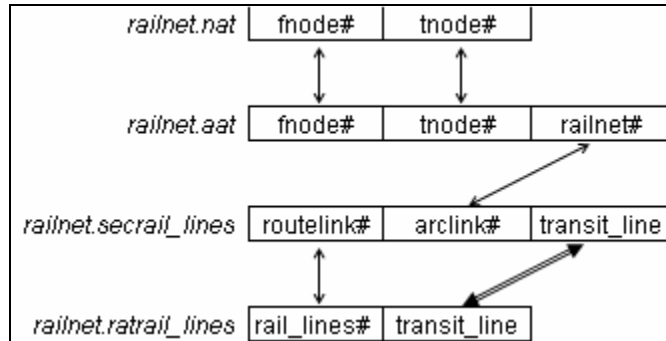
Node variables are listed in Table 2. The values for *pspace* and *pcost* represent conditions in the base scenario, thus these variables should be zero if the nodes are not included in the base scenario. Descriptions of *future_pspace* and *future_pcost* are provided in Table 2 below. A *future_pspace* value of "4:150:6:200" is interpreted as follows: the node will have 150 parking spaces beginning in scenario 400 and 200 parking spaces beginning in scenario 600. The value is assigned through scenarios until a later scenario is specified, so the node will have 150 parking spaces in scenario 500 as well. While it is easier for the analyst to read this value if the scenarios are coded in chronological order, the processing programs do not require this in order to assign the correct value to each scenario.

Table 2. Node Table Variables (railnet.nat)

Variable	Description
RAILNET-ID	CATS node number. CTA rail uses range 30000-39999. Metra uses range 40000-49999.
LABEL	Node label (station name).
PSPACE	Number of parking spaces at node <i>in base scenario</i> ; zero if not used in base scenario.
PCOST	Parking cost at node <i>in base scenario</i> ; zero if not used in base scenario.
FUTURE_PSPACE	Number of parking spaces at node in future scenarios. This text string uses the format "s1:p1:s2:p2: ..." where <i>s</i> =the hundred's place value from C06 scenario numbers and <i>p</i> =the number of parking spaces in the corresponding scenario. Each value must be separated by a colon.
FUTURE_PCOST	Parking cost at node in future scenarios; same format as <i>future_pspace</i> .
X-COORD	ARC generated x coordinate, IL east state plane feet.
Y-COORD	ARC generated y coordinate, IL east state plane feet.
FNODE#	ARC generated unique identification number for "From" node.
TNODE#	ARC generated unique identification number for "To" node.

Figure 1 illustrates how the arc, node and rail transit route and section tables are related to each other. The linkage between *transit_line* in the route and section tables is slightly different than the others. The user (via an AML) explicitly creates this when new route information is added to the table. While not essential, this linkage was added to help simplify understanding the route coding information. All other linkages shown are automatically maintained by ARC. Figure 1 shows the **only** relationships that should be used between these tables. For instance, *rail_lines#* in the route table **does not** represent the same thing it does in the section table.

Figure 1. Data Table Relationships



1.2 Rail transit tables

Variables in the rail transit route table (shown in Table 3) correspond to header information EMME/2 requires when reading in rail itineraries. The use of the *scenario* variable allows all base and future rail coding to be stored in one location and each rail route to be coded only once in the route system. *Scenario* allows the programs to select only those rail routes that are to be included in a specific network scenario.

Table 3. Rail Transit Route Table Variables (railnet.ratrail_lines)

Variable	Description
RAIL_LINES#	ARC generated unique identification number for each rail route.
TRANSIT_LINE	CATS rail route name.
DESCRIPTION	Real-world description of rail route (20 character maximum).
MODE	Rail mode: C=CTA rail, M=Metra rail
VEHICLE_TYPE	Rail vehicle type: 7=mode C, 8=mode M
HEADWAY	AM peak period rail headway in minutes; applies to first scenario where route appears, not base scenario.
SPEED	Rail route speed in mph, not used but required by EMME/2 (cannot be 0).
SCENARIO	Text string comprised of concatenated hundred's place values from C06 scenario numbers, indicating which scenarios include the rail route. Use CONTAINS (ArcInfo) or LIKE (ArcGIS) to select in a query.
FUTURE_HEADWAY	Headway in future scenarios; same format as <i>future_pspace</i> in NAT.

Rail lines were included in the route table based on their unique itineraries. For instance, 'ctr007' (Skokie Swift) has a different itinerary in scenarios 100-400 than it does in scenarios 500-600, so there are two entries in the table: 'ctr007' and 'ctr007_5'. The '_5' portion of the line name indicates that the transit line first appears in scenario 500. This portion of the line name is not included in the EMME/2 batchin file, as EMME/2 only accepts six characters in the name. This naming convention provides an easy way for the analyst to see that while line 'ctr007' is

included in all scenarios, its itinerary changes between scenarios 400 and 500. The headway value for each line applies to the first scenario the line appears in, which is not necessarily the base scenario.

The actual itinerary information for rail routes is contained in the section table. These variables are listed in Table 4. The variable *transit_line* can be used to relate the rail transit route and section tables.

Table 4. Rail Transit Section Table Variables (railnet.secrail_lines)

Variable	Description
ROUTELINK#	ARC generated unique identification number for route.
ARCLINK#	ARC generated unique identification number for link.
TRANSIT_LINE	CATS rail route name.
ITINERARY_A	CATS "From" node of itinerary segment.
ITINERARY_B	CATS "To" node of itinerary segment.
ORDER	Order number of rail segment in itinerary.
LAYOVER	Layover time in minutes.
DWELL_CODE	Code for stops (corresponding EMME/2 code), applies to itinerary_b: 0=stop, default dwell time of 0.01 minutes 1=no stop (#), default dwell time of 0 minutes 2=alighting only (>) 3=boarding only (<) 4=boarding & alighting allowed (+) 5=dwell time factor (*)
ZONE_FARE	Incremental zone fare in cents.
LINE_SERV_TIME	Itinerary segment travel time in minutes.
DWELL_TIME	Dwell time at stop in minutes.

1.3 Maintaining topology

Topology defines the spatial relationships in an ARC coverage. For instance, arcs have a beginning and an ending point, and arcs sharing a common beginning/ending point are connected. It is necessary to update the topology in railnet after the network has been edited because it ensures that all of the relationships are current. The user updates coverage topology by running **update_rail.aml** (listed in Appendix A). Topology must be updated when any of the following occur:

- links are added or deleted from the coverage,
- nodes are added, deleted or moved, or
- the node number is changed.

It is not necessary to update coverage topology when arc or node attributes are edited (except in the case of the node number), or when routes or sections are edited, added or deleted. When **update_rail.aml** is run, it automatically calls **itinerary_update.aml** and **itinerary2.sas** to perform the following functions:

- update x- and y-coordinates in the NAT,
- update anode and bnode values in the AAT,

- update itinerary_a and itinerary_b values in the SEC,
- update coded link length in the AAT for links that are split, and
- update affected itinerary coding in the SEC when links are split.

When a link is split (i.e., a new node is added somewhere along the link) and transit segments are affected, the itinerary coding will be automatically adjusted by the AML. The itinerary order will be adjusted to incorporate the new segments. A layover or a zone fare on the original segment will be applied to the last segment of the string of split links. The line service time of the original segment will be split among the new segments proportionally based on link length. The dwell code and dwell time of the original segment will be applied to all of the new segments. *Note:* the analyst should verify the itinerary coding update correctly reflects the desired conditions and make any manual corrections that are necessary.

2 Coding Rail Routes

Rail itinerary coding must be submitted in the format shown in Table 5. The entire rail itinerary must be submitted each time, regardless of whether the route is being added to the route system for the first time or only a minor correction is being made. There is no limit on the number of rail routes that can be imported at one time.

Table 5. Rail Itinerary Coding Sheet

transit line	itinerary a	itinerary b	order	layover	dwell code	dwell time	zone fare	line serv time
xyz345	32009	32010	1					1
xyz345	32010	32011	2					2
xyz345	32011	32012	3				123	1.1
xyz345	32012	32013	4		1			
xyz345	32013	32014	5	0				6
xyz345	32014	32015	6					1
xyz345	32015	32016	7		2			1
xyz345	32016	32017	8					1
xyz345	32017	32018	9		1			
xyz345	32018	32019	10					3
xyz345	32019	32020	11					1
xyz345	32020	32021	12	5				3

The first four data items in the coding sheet must be filled out for every record. Only actual layover values should be included in the layover column; thus a zero indicates a layover time of zero. A maximum of two cells per rail line should have a layover value (this is a limit set by EMME/2). Blank cells are allowed for the four remaining data items and are interpreted as zeroes. If no dwell time is specified, a default dwell time value of 0.01 minutes is assigned to every segment where dwell code does not equal one and a default value of 0 minutes is assigned to every segment with a dwell code of one. The layover, dwell code and dwell time values apply to the *itinerary b* node of each segment.

The rail route header information required by EMME/2 must always be submitted along with the itinerary coding. This information must be submitted in a file separate from the itinerary coding and be formatted as shown in Table 6.

Table 6. Rail Route Header Information

transit line	description	mode	veh. type	headway	speed
xyz345	HOWARD DAN RYAN	C	7	3.8	25

After the rail itinerary information is imported into ARC, the *scenario* variable in the route table needs to be calculated to identify which scenario networks will include the rail route. This field may be calculated manually, or a spreadsheet like the one shown in Table 7 can be used to track the routes contained in each scenario. The program **rail_scenario_add.aml** can be used to import the scenario data into the route table. *Future_headway* should be calculated manually if it is needed.

Table 7. Rail Scenario Sheet

line	scen_code	Conformity 2006 Scenarios				
		100	300	400	500	600
ctr001	134	1	1	1		
ctr001_5	56				1	1
ctr002	1345	1	1	1	1	
ctr002_6	6					1
ctr003	13456	1	1	1	1	1
ctr004	13456	1	1	1	1	1
ctr005	13456	1	1	1	1	1
ctr006	1345	1	1	1	1	
ctr007	134	1	1	1		
ctr007_5	56				1	1

Importing rail route coding into railnet

The rail itinerary coding information is saved to the comma-separated value file “rail_itin.csv” with no header, and stored in the working directory along with the railnet coverage. The transit line information is saved to “rail_route.csv” in the same location. AMLs are then run to import the data into the coverage. These programs are listed in Appendix B and are briefly described below.

1. **rail_path_edit.aml** - This AML imports the rail route coding into ARC and must be submitted with the coverage and route names. ARC’s path-building algorithm is used to create the rail route based on the itinerary coding and an impedance value. The SAS program **railpath2.sas** is automatically called to perform some quality checks on the itinerary coding and to reformat it for use by the path-building algorithm. This program identifies miscoded links, itinerary gaps, routes coded in the wrong direction on a one-way link and any routes with more than two layovers that must be corrected before data are imported into the coverage.
2. **rail_scenario_add.aml** – This program imports data from “rail_scen.csv” into the rail transit route table to fill the *scenario* field. The name of the coverage and route system are required arguments. This program could be altered to populate the *future_headway* field as well.

3 Exporting Rail Data for EMME/2 Networks

In order to use the rail transit information stored in ARC to develop an EMME/2 transit scenario network, data must be exported from the coverage and formatted into four files: rail.itinerary, rail.network, railnode.extatt and railseg.extatt. This is accomplished by submitting **export_rail_scenario.aml** (listed in Appendix C) along with two arguments: the coverage name and scenario number. This program extracts the appropriate rail lines and their itineraries for the scenario from the coverage, as well as the links, nodes and auxiliary links needed to build the rail network. The SAS program **export_rail2.sas** is automatically called to format the data into the four rail files.

Once the four rail transit files are created, the normal procedures can be used to generate an EMME/2 transit network (refer to section 5 of Working Paper 06-04 for a complete description).

Appendix A: Coverage Maintenance Programs

```
/* UPDATE_RAIL.AML */
/* AML program updates coverage information and topology. */
/* Link Anode-Bnode values and itinerary node values are updated. */
/* Used to update the rail network after editing. */

/* CALL FROM ARC PROMPT WITH "&r update_rail <coverage> <route>" */

/* Craig Heither, last revised 11/20/06 */
/* (based on Updatebase.aml originally developed by Matt Rogus) */

&args cov route

/* ERROR CHECKING */
&if [null %cov%] &then &return YOU MUST ENTER A COVERAGE NAME!
&if [null %route%] &then &return YOU MUST ENTER A ROUTE SYSTEM!
&if not [exists %cov%.aat -info] &then &return COVERAGE DOES NOT EXIST!
&if not [exists %cov%.rat%route% -info] &then &return ROUTE SYSTEM DOES NOT EXIST!
&if [exists len.dat -info] &then killinfo len.dat

build %cov% lines
build %cov% nodes
addxy %cov% node

/* UPDATE CODED LENGTH FOR SPLIT LINKS IF NECESSARY */
frequency %cov%.aat len.dat
%cov%-id; end
length; end
ae
edit len.dat info
sel frequency gt 1
&sv total = [extract 1 [show number selected]]
&if %total% > 0 &then &do
  sel frequency = 1
  delete
  save
  &sv cntr = 1
  &sv var1 = :edit.%cov%-id
  &do &while %cntr% le %total%
    edit len.dat info
    sel all
    cursor open
    cursor %cntr%
    &sv id = [value %var1%]
    &sv tolen = %:edit.length%
    cursor close
    edit %cov% arc
    sel %cov%-id = %id%
    cal miles = length / %tolen% * miles
    save
    &sv cntr = %cntr% + 1
  &end
q; y; y
&type *~~~~~*
&type LENGTH FOR %total% LINKS UPDATED
&type *~~~~~*
&end
&else &do
  q
&end
killinfo len.dat

/* UPDATE ANODE & BNODE IN AAT */
indexitem %cov%.aat fnode#
indexitem %cov%.aat tnode#
indexitem %cov%.nat fnode#
indexitem %cov%.nat tnode#

relate add
```

```

temp1
%cov%.nat
info
fnode#
fnode#
linear
ro
temp2
%cov%.nat
info
tnode#
tnode#
linear
ro
~
ap
&type *** updating anode and bnode fields in AAT file ***
calculate %cov% arc anode = temp1//%cov%-id
calculate %cov% arc bnode = temp2//%cov%-id
calculate %cov% arc %cov%-id = %cov%#
q

/* UPDATE NODES IN ITINERARY */
indexitem %cov%.aat %cov%#
indexitem %cov%.sec%route% arlink#

relate add
temp3
%cov%.aat
info
arlink#
%cov%#
linear
ro
~

ap
&type *** updating %route% itineraries ***
asel %cov% section.%route%
cal %cov% section.%route% itinerary_a = temp3//anode
cal %cov% section.%route% itinerary_b = temp3//bnode
resel %cov% section.%route% f-pos gt t-pos
&sv flag = [extract 1 [show select %cov% section.%route%]]
&if %flag% gt 0 &then &do
  cal %cov% section.%route% itinerary_a = temp3//bnode
  cal %cov% section.%route% itinerary_b = temp3//anode
&end
q
relate drop $ALL

/* UPDATE ARC IDs */
idedit %cov% line

dropindex %cov%.aat
dropindex %cov%.nat
dropindex %cov%.sec%route%

&r programs\rail\itinerary_update %cov% %route%

/* UNLOAD ROUTE AND SECTION FILES */
&sv when = [date -usa]
listoutput rail.section init
listoutput note %when%
listoutput note transit_line,itinerary_a,itinerary_b,order,layover,dwell_code,dwell_time,zone_fare,line_serv_time
listoutput screen
listoutput rail.route init
listoutput note %when%
listoutput note transit_line,description,mode,vehicle_type,headway,speed
listoutput screen
tables
sel %cov%.sec%route%
  unload rail.section transit_line itinerary_a itinerary_b order ~
  layover dwell_code dwell_time zone_fare line_serv_time
sel %cov%.rat%route%

```



```

/* IMPORT CORRECTED DATA AND UPDATE ITINERARY SEGMENTS */
tables
define new.dat
code,8,8,i
ord,3,3,i
lay,8,8,c
dwell,1,1,i
zfare,7,7,n,2
lserv,8,8,n,5
dtime,5,5,n,2
~
add from %dir%new.txt

sel %cov%.sec%route%
additem %cov%.sec%route% code 8 8 i
cal code = routelink# * 100000 + %route%#
sel new.dat
resel lay = 'X'
cal lay = ''
q

indexitem new.dat code
indexitem %cov%.sec%route% code
relate add
temp1
new.dat
info
code
code
linear
ro
~
ap
asel new.dat info
resel %cov% section.%route% keyfile new.dat info code
&type *** UPDATING ITINERARY INFORMATION ***
calculate %cov% section.%route% order = temp1//ord
calculate %cov% section.%route% layover = temp1//lay
calculate %cov% section.%route% dwell_code = temp1//dwell
calculate %cov% section.%route% zone_fare = temp1//zfare
calculate %cov% section.%route% line_serv_time = temp1//lserv
calculate %cov% section.%route% dwell_time = temp1//dtime
q
relate drop $ALL

&type
&type *****
&type ITINERARIES WERE UPDATED FOR THESE LINES:
list freq.dat transit_line
&type *****
&type

dropitem %cov%.sec%route% %cov%.sec%route% code
dropindex %cov%.sec%route%
killinfo freq.dat
killinfo update.out
killinfo new.dat
&if [delete %dir%freq.txt] = 0 &then &type Successful Deletion of FREQ.TXT
&if [delete %dir%update.txt] = 0 &then &type Successful Deletion of UPDATE.TXT
&if [delete %dir%arcs.txt] = 0 &then &type Successful Deletion of ARCS.TXT
&if [delete %dir%new.txt] = 0 &then &type Successful Deletion of NEW.TXT
&return SEGMENT UPDATE FINISHED

```

/* **itinerary2.sas**

Heither, last revised 10/16/06

PROGRAM IS CALLED BY ITINERARY_UPDATE.AML AND
UPDATES RAIL ITINERARIES TO GO INTO ARC.
----- */

%let dir=&sysparm;

/*-----*/

 * INPUT FILES *;
filename in1 "&dir.freq.txt";
filename in2 "&dir.update.txt";
filename in3 "&dir.arcs.txt";

 * OUTPUT FILES *;
filename out1 "&dir.new.txt";

/*-----*/

 ** READ IN FREQUENCY FILE **;

data freq; infile in1 missover dsd;
input line \$ order freq;
proc sort; by line order;

 ** READ IN ITINERARY CODING **;

data itin(drop=r1 r); infile in2 missover dsd;
input r1 link r line \$ a b order layover \$ dwell zfare lserv dtime;
code=r1*100000+r;
if layover=' ' then layover='X';
proc sort; by line order code;

 ** CORRECT SEGMENT CODING **;

data fix; merge itin freq (in=hit); by line order; if hit;
proc sort; by link;

data length; infile in3 missover dsd;
input link miles;
proc sort; by link;

data fix; merge fix (in=hit) length; by link; if hit;
proc sort; by line order code;
proc summary nway; var miles; class line order; output out=junk sum=totalmi;

data fix(drop=_type__freq_); merge fix junk; by line order;
lserv=round(miles/totalmi*lserv,.1);
proc sort; by line order;

data fix(keep=layover dwell zfare lserv code); set fix; by line order;
if not last.order then layover='X';
if not last.order then zfare=0;
proc sort; by code;

 ** UPDATE ITINERARY **;

proc sort data=itin; by code;
data itin(drop=link); merge itin fix; by code;
proc sort; by line code;

data itin; set itin; by line code;
retain ord 1;
ord+1;
if first.line then ord=1;
output;

 ** WRITE FILE TO UPDATE ITINERARIES IN ARC **;

data itin; set itin;
file out1 dlm='';
put code ord layover dwell zfare lserv dtime;

run;

Appendix B: Programs to Import Rail Coding

```
/* RAIL_PATH_EDIT.AML */
/* Heather, last revised 11/20/06 */
/* Call with "&r rail_path_edit <coverage name> <route>" */

/* ***** */
/* THIS PROGRAM IS USED TO IMPORT RAIL ITINERARIES INTO ARC. */
/* THE COVERAGE & ROUTE WHERE THE RAIL CODING WILL BE STORED ARE REQUIRED. */
/* */
/* TWO INPUT FILES ARE NEEDED: */
/* RAIL_ITIN.CSV - FILE OF RAIL ITINERARIES IN A SPECIFIED FORMAT. */
/* RAIL_ROUTE.CSV - RAIL HEADER INFORMATION IN A SPECIFIED FORMAT. */
/* ***** */

&args cov route

/*#####*/
/* CHANGE INFO HERE */
&sv dir = C:\build_networks\ /* storage directory path */
/*#####*/
&sv count = 1

/** HOUSEKEEPING **/
&if [null %cov%] &then &return YOU MUST ENTER A COVERAGE NAME.
&if [null %route%] &then &return YOU MUST ENTER A ROUTE SYSTEM.
&if not [exists %cov%.aat -info] &then &return COVERAGE DOES NOT EXIST!
&if not [exists %cov%.rat%route% -info] &then &return ROUTE SYSTEM DOES NOT EXIST!
&if not [exists rail_itin.csv -file] &then &return ITINERARY FILE DOES NOT EXIST!
&if not [exists rail_route.csv -file] &then &return ROUTE HEADER FILE DOES NOT EXIST!
&if [exists railpath2.lst -file] &then &do
  &if [delete railpath2.lst] = 0 &then &type FILE CLEANUP
&end
&if [exists route.new -info] &then killinfo route.new
&if [exists itinerary.new -info] &then killinfo itinerary.new
&if [exists %cov%.stp -info] &then killinfo %cov%.stp
&if [exists %cov%.sectranpath -info] &then dropfeatures %cov% section.tranpath
&if [iteminfo %cov% -arc abnode -exists] &then dropitem %cov%.aat %cov%.aat abnode ft_imp tf_imp arlink#

/** SELECT DATA & WRITE TO FILES FOR SAS PROCESSING & CODING VERIFICATION **/
tables
sel %cov%.aat
unload %dir%links.txt anode bnode directions init
q

&type *****
&type * RUNNING SAS *
&type *****
"C:\Program Files\SAS\SAS System\9.0\sas.exe" -sysin %dir%programs\rail\railpath2.sas -sysparm %dir%
&if [exists railpath2.lst -file] &then &return FIX RAIL ROUTE CODING.

/** IMPORT DATA & BUILD ROUTES **/
tables
define route.new
transit_line,8,8,c
description,20,20,c
mode,1,1,c
vehicle_type,1,1,i
headway,5,5,n,2
speed,4,4,n,1
~
add from rail_route.csv
define itinerary.new
transit_line,8,8,c
anode,5,5,i
bnode,5,5,i
layover,8,8,c
dwell_code,1,1,i
zone_fare,7,7,n,2
line_serv_time,8,8,n,5
```

```

dwell_time,5,5,n,2
order,3,3,i
route,3,3,i
place,5,5,n,0
abnode,10,10,n,0
~
add from itin.txt
define %cov%.stp
%cov%-id,5,5,n,0
order,5,5,n,0
route,5,5,n,0
~
add from path.txt

sel route.new
additem route.new link 4 4 i
cal link = $recno
sel itinerary.new
additem itinerary.new code 8 8 n 0
cal code = route * 10000 + place
sel itinerary.new
resel layover = 'X'
cal layover = ' '
sel %cov%.aat
additem %cov%.aat abnode 10 10 n 0
cal abnode = anode * 100000 + bnode
/**** ADD IMPEDANCE VARIABLE TO PROHIBIT USE OF UNDESIRABLE LINKS ****/
/**** (NEGATIVE VALUES MEAN LINK CANNOT BE TRAVERSED) ****/
additem %cov%.aat ft_imp 5 5 n 2
additem %cov%.aat tf_imp 5 5 n 2
cal ft_imp = -1
cal tf_imp = -1
additem %cov%.aat arclink# 4 5 b
cal arclink# = %cov%#
q

/**** DETERMINE NUMBER OF ROUTES TO CREATE ****/
ae
edit route.new info
sel all
cursor open
&sv max = %:edit.AML$NSEL%
cursor close
q; n

/**** ITERATE THROUGH ROUTES AND BUILD THEM ****/
/**** This was added to build each individually to eliminate any chance of path building error ****/
&do &while %count% le %max%
ap
asel itinerary.new info
resel itinerary.new info route = %count%
infile itinerary.new info one.new init
resel %cov% arc keyfile one.new info abnode
cal %cov% arc ft_imp = miles
resel %cov% arc directions gt 1
&sv flag = [extract 1 [show select %cov% arc]]
&if %flag% gt 0 &then cal %cov% arc tf_imp = miles
asel %cov% arc ft_imp gt 0

asel %cov%.stp info
resel %cov%.stp info route = %count%
infile %cov%.stp info one.stp init

netcover %cov% tranpath
impedance ft_imp tf_imp
stops one.stp order route
path stops

asel %cov% arc
cal %cov% arc ft_imp = -1
cal %cov% arc tf_imp = -1
q
&sv count = %count% + 1

```

```

&end

/** ADD DATA TO SECTION AND ROUTE TABLES */
tables
&if not [iteminfo %cov% -route.tranpath transit_line -exists] &then &do
  additem %cov%.ratranpath transit_line 8 8 c
  additem %cov%.ratranpath description 20 20 c
  additem %cov%.ratranpath mode 1 1 c
  additem %cov%.ratranpath vehicle_type 1 1 i
  additem %cov%.ratranpath headway 5 5 n 2
  additem %cov%.ratranpath speed 4 4 n 1
  additem %cov%.ratranpath scenario 8 8 c
  additem %cov%.ratranpath future_headway 20 20 c
&end
&if not [iteminfo %cov% -section.tranpath transit_line -exists] &then &do
  additem %cov%.sectranpath transit_line 8 8 c
  additem %cov%.sectranpath itinerary_a 5 5 i
  additem %cov%.sectranpath itinerary_b 5 5 i
  additem %cov%.sectranpath order 3 3 i
  additem %cov%.sectranpath layover 8 8 c
  additem %cov%.sectranpath dwell_code 1 1 i
  additem %cov%.sectranpath zone_fare 7 7 n 2
  additem %cov%.sectranpath line_serv_time 8 8 n 5
  additem %cov%.sectranpath dwell_time 5 5 n 2
&end
&if not [iteminfo %cov% -section.tranpath code -exists] &then additem %cov%.sectranpath code 8 8 n 0
sel %cov%.sectranpath
  cal code = routelink# * 10000 + tranpath-id
q

indexitem %cov%.aat %cov%#
indexitem %cov%.sectranpath arclink#
indexitem %cov%.sectranpath code
indexitem %cov%.sectranpath routelink#
indexitem %cov%.ratranpath tranpath#
indexitem %cov%.ratranpath transit_line
indexitem itinerary.new code
indexitem route.new transit_line

ap
relate add
one
  %cov%.aat
  info
  arclink#
  %cov%#
  linear
  ro
two
  itinerary.new
  info
  code
  code
  linear
  ro
three
  %cov%.sectranpath
  info
  tranpath#
  routelink#
  linear
  ro
four
  route.new
  info
  transit_line
  transit_line
  linear
  ro
~

asel %cov% section.tranpath
cal %cov% section.tranpath itinerary_a = one//anode

```

```

cal %cov% section.tranpath itinerary_b = one//bnode
resel %cov% section.tranpath f-pos gt t-pos
&sv flag = [extract 1 [show select %cov% section.tranpath]]
&if %flag% gt 0 &then &do
    cal %cov% section.tranpath itinerary_a = one//bnode
    cal %cov% section.tranpath itinerary_b = one//anode
&end
asel %cov% section.tranpath
cal %cov% section.tranpath transit_line = two//transit_line
cal %cov% section.tranpath layover = two//layover
cal %cov% section.tranpath order = two//order
cal %cov% section.tranpath dwell_code = two//dwell_code
cal %cov% section.tranpath zone_fare = two//zone_fare
cal %cov% section.tranpath line_serv_time = two//line_serv_time
cal %cov% section.tranpath dwell_time = two//dwell_time
asel %cov% route.tranpath
cal %cov% route.tranpath transit_line = three//transit_line
cal %cov% route.tranpath description = four//description
cal %cov% route.tranpath mode = four//mode
cal %cov% route.tranpath vehicle_type = four//vehicle_type
cal %cov% route.tranpath headway = four//headway
cal %cov% route.tranpath speed = four//speed
relate drop $ALL
q

/** REMOVE TRANSIT LINES BEING UPDATED. **/
ae
&if [exists %cov%.sec%route% -info] &then &do
    edit %cov% route.%route%
    arcplot asel route.new info
    arcplot resel %cov% route.%route% keyfile route.new info transit_line
    selectget
    &sv flag = [extract 1 [show number selected]]
    &if %flag% gt 0 &then &do
        delete
        save
    &end
&end

/** PUT TRANSIT ROUTES IN FINAL LOCATION ***/
edit %cov% route.tranpath
sel all
duplicate %route%
&if [exists %cov%.sec%route% -info] &then y
save
q

/** CLEAN UP **/
tables
dropitem %cov%.aat abnode ft_imp tf_imp arlink#
sel %cov%.rat%route%
cal %route%-id = %route%#
q
&type * * * * *
&type * DELETING TEMPORARY FILES *
&type * * * * *
killinfo itinerary.new
killinfo route.new
killinfo %cov%.stp
&if [delete %dir%itin.txt] = 0 &then &type Successful Deletion of ITIN.TXT
&if [delete %dir%path.txt] = 0 &then &type Successful Deletion of PATH.TXT
&if [delete %dir%section.txt] = 0 &then &type Successful Deletion of SECTION.TXT
&if [delete %dir%links.txt] = 0 &then &type Successful Deletion of LINKS.TXT
dropfeatures %cov% section.tranpath
dropindex %cov%.aat

/** UNLOAD ROUTE AND SECTION FILES **/
&sv when = [date -usa]
listoutput rail.section init
listoutput note %when%
listoutput note transit_line,itinerary_a,itinerary_b,order,layover,dwell_code,dwell_time,zone_fare,line_serv_time
listoutput screen
listoutput rail.route init

```

```
listoutput note %when%
listoutput note transit_line,description,mode,vehicle_type,headway,speed
listoutput screen
tables
sel %cov%.sec%route%
  unload rail.section transit_line itinerary_a itinerary_b order ~
  layover dwell_code dwell_time zone_fare line_serv_time
sel %cov%.rat%route%
  unload rail.route transit_line description mode vehicle_type headway speed
q
&return AML IS FINISHED
```

/* railpath2.sas

Heither, last revised 11/20/06

PROGRAM IS CALLED BY RAIL_PATH_EDIT.AML AND
FORMATS RAIL ITINERARIES TO BUILD IN ARC.

```
----- */
%let dir=&sysparm;
/*-----*/
  * INPUT FILES *;
filename in1 "&dir.rail_itin.csv";
filename in2 "&dir.rail_route.csv";
filename in3 "&dir.links.txt";
  * OUTPUT FILES *;
filename out1 "&dir.itin.txt";
filename out2 "&dir.path.txt";
filename out3 "&dir.rail_route.csv";
/*-----*/

** READ IN CODING FOR RAIL ROUTES **;
data section; infile in1 missover dsd;
input line $ anode bnode ord layover $ dwcode dwtime zfare ltime;
  if layover=' ' then layover='X';
  if dwcode='.' then dwcode=0;
  if dwtime='.' and dwcode ne 1 then dwtime=0.01;
  else if dwtime='.' and dwcode=1 then dwtime=0;
  if zfare='.' then zfare=0;
  if ltime='.' then ltime=0;
group=lag1(line);
proc sort; by line ord;

data verify; set section; proc sort; by anode bnode;

  ** READ IN ROUTE TABLE CODING **;
** This will ensure description is enclosed in single quotes for ARC **;
data rte; infile in2 missover dlm=';';
length desc $22. d $20.;
input line $ desc $ mode $ type hdwy spd;
d=compress(desc,'');
desc="||trim(d)||";
proc sort; by line;

** Replace File for ARC **;
data rte; set rte;
file out3 dlm=';';
put line desc mode type hdwy spd;

  *-----*;
  ** VERIFY CODING **;
  *-----*;

** Read in rail Links **;
data arcs(drop=c); infile in3 missover dlm=';';
input anode bnode dir;
match=1;
output;
if dir>1 then do;
c=anode; anode=bnode; bnode=c;
output;
end;
proc sort; by anode bnode;

** Find Unmatched Links **;
data check; merge verify (in=hit) arcs; by anode bnode;
if hit;
if match=1 then delete;
proc print; var anode bnode line ord;
title 'MIS-CODED ANODE-BNODE OR DIRECTIONAL PROBLEM ON THESE LINKS';

** Ensure Transit Not Coded on Centroid Links **;
data bad; set verify;
if anode le 1891 or bnode le 1891;
proc print; var anode bnode line ord;
title 'TRANSIT CODING ON CENTROID CONNECTORS';
```

```

*-----*;
** FORMAT ITINERARY DATASET **;
*-----*;
data section(drop=ord); set section;
retain order 1;
order+1;
if line ne group then order=1;
output;
data section; set section;
retain route 0;
if line ne group then route+1;
output;

data section; set section;
place=_n_;
proc sort; by anode bnode;

data arcs; infile in3 missover dlm=" ";
input anode bnode;
true=1;
proc sort; by anode bnode;

** Find True Arc Direction in Rail Network **;
data section; merge section (in=hit) arcs; by anode bnode;
if hit;
if true=1 then abnode=anode*10000+bnode;
else abnode=bnode*10000+anode;
proc sort; by line order;

*-----*;
** WRITE ITINERARY FILE **
*-----*;
data writeout; set section;
file out1 dlm=' ';
put line anode bnode layover dwcode zfare ltime dwtime
order route place abnode;

* ----- *;
**REPORT ITINERARY GAPS**;
**THESE ARE MIS-CODED LINKS**;
data check; set section;
z=lag1(bnode);
if anode ne z and order>1 then output;
proc print; var line order anode bnode z;
title 'Itinerary Gaps';
* ----- *;
* ----- *;
**REPORT LAYOVER PROBLEMS**;
**A MAXIMUM OF TWO LAYOVERS ARE ALLOWED PER TRANSIT LINE **;
data check; set section; if layover ne 'X';
proc freq; tables line / noprint out=check;
data check; set check;
if count>2;
proc print; var line count;
title 'Too Many Layovers Coded';
* ----- *;
*-----*;
** FORMAT PATH-BUILDING FILE FOR ARC **
*-----*;
data path(keep=node route); set section; by line order;
node=anode; output;
if last.line then do;
node=bnode; output;
end;

data path; set path;
rank=_n_;

*-----*;
** WRITE PATH FILE **
*-----*;
data write2; set path;

```

```
file out2 dlm='';  
put node rank route;  
run;
```

Appendix C: Programs to Export Rail Coding

```
/* EXPORT_RAIL_SCENARIO.AML */
/* Craig Heither, last rev. 11/20/06 */
/* Call with "&r export_rail_scenario <coverage> <scenario #>" */
/* Ex: "&r export_rail_scenario railnet 500" */

/* ***** */
/* PROGRAM USED TO CREATE 4 EMME/2 BATCHIN FILES WITH */
/* RAIL TRANSIT INFO FOR A SCENARIO: RAIL.ITINERARY, */
/* RAIL.NETWORK, RAILNODE.EXTATT AND RAILSEG.EXTATT. */
/* ***** */

/* C06 SCENARIO NETWORKS */
/* ----- */
/* 2002 base (100) */
/* 2007 action (300) */
/* 2010 action (400) */
/* 2020 action (500) */
/* 2030 action (600) */

&args cov scen

/*#####*/
/*#####*/
/*** CHANGE DATA HERE ***/
&sv putdir = M:\proj1\cmh\test\transit\ /* directory to hold output files */
&sv amldir = C:\build_networks\programs\rail\ /* directory that holds AML & SAS programs */
&sv rte = rail_lines /* ARC route system holding rail coding */
&sv maxz = 1891 /* zn04 highest POE zone number */
/*#####*/
/*#####*/

&if [null %cov%] &then &return YOU MUST ENTER A COVERAGE NAME!!
&if [null %scen%] &then &return YOU MUST ENTER A SCENARIO!!
&if not [exists %cov%.aat -info] &then &return COVERAGE DOES NOT EXIST!!
&if not [exists %cov%.rat%rte% -info] &then &return ROUTE SYSTEM DOES NOT EXIST!!
&sv scn = %scen% / 100
&sv apos = '
&sv d = $

/* FILE CLEANUP, IF NECESSARY */
&if [iteminfo %cov% -arc arclink# -exists] &then dropitem %cov%.aat %cov%.aat arclink#
&if [exists export_rail2.lst -file] &then &do
  &if [delete export_rail2.lst] = 0 &then &type FILE CLEANUP
&end

/* VERIFY SCENARIO VALUE */
&select %scen%
&when 100, 300, 400, 500, 600
  &type PROCESSING
&otherwise
  &return YOU ENTERED AN INVALID SCENARIO!!
&end
&if not [exists %putdir%%scen% -directory] &then mkdir %putdir%%scen%

/*-----*/
/* EXPORT ARC, NODE AND RAIL ITINERARY DATA */
/*-----*/
&if not [iteminfo %cov% -arc arclink# -exists] &then &do
  tables
  additem %cov%.aat arclink# 4 5 b
  sel %cov%.aat
  calc arclink# = %cov%#
  q
&end

ap
/* RAIL HEADER DATA */
resel %cov% route.%rte% scenario cn %apos%%scen%%apos%
```

```

infile %cov% route.%rte% lines.out description transit_line ~
mode vehicle_type headway speed future_headway init

/* RAIL ITINERARY DATA */
resel %cov% section.%rte% keyfile %cov% route.%rte% transit_line
infile %cov% section.%rte% itins.out arlink# f-pos t-pos transit_line layover ~
order itinerary_a itinerary_b dwell_code zone_fare line_serv_time dwell_time init

/* RAIL NETWORK DATA */
resel %cov% arc keyfile %cov% section.%rte% arlink#
asel %cov% arc aux_scenario cn %apos%%scn%%apos%
infile %cov% arc arcs.out %cov%# anode bnode miles modes1 modes2 directions lanes type init
resel %cov% node keyfile %cov% arc fnode#
asel %cov% node keyfile %cov% arc mnode#
infile %cov% node nodes.out %cov%-id label pspace pcost future_pspace future_pcost x-coord y-coord init
q

/* UNLOAD FILES FOR SAS PROCESSING */
tables
sel arcs.out
unload %putdir%%scn%\network.txt init
kill arcs.out
sel nodes.out
unload %putdir%%scn%\nodes.txt init
kill nodes.out
sel lines.out
unload %putdir%%scn%\lines.txt init
kill lines.out
sel itins.out
resel layover = ''
cal layover = 'X'
sel itins.out
unload %putdir%%scn%\itins.txt init
kill itins.out
q

dropitem %cov%.aat %cov%.aat arlink#

&type *****
&type * SAS INITIATED - CREATING RAIL FILES *
&type *****
"C:\Program Files\SAS\SAS System\9.0\sas.exe" -sysin %amldir%export_rail2.sas -sysparm %putdir%%d%%scn%%d%%maxz%
&if [exists export_rail2.lst -file] &then &return FIX CODING ERRORS!

&type *****
&type * DELETING TEMPORARY DATA *
&type *****
&if [delete %putdir%%scn%\network.txt] = 0 &then &type Successful Deletion of NETWORK.TXT
&if [delete %putdir%%scn%\itins.txt] = 0 &then &type Successful Deletion of ITINS.TXT
&if [delete %putdir%%scn%\lines.txt] = 0 &then &type Successful Deletion of LINES.TXT
&if [delete %putdir%%scn%\nodes.txt] = 0 &then &type Successful Deletion of NODES.TXT

&return AML IS FINISHED.

```

/* EXPORT_RAIL2.SAS

Craig Heither, CATS - last rev. 11/20/06

PROGRAM CREATES RAIL TRANSIT NETWORK BATCHIN FILES.

```

                */
%let dirpath=%scan(&sysparm,1,$);
%let scen=%scan(&sysparm,2,$);
%let maxzone=%scan(&sysparm,3,$); *highest zn04 POE zone number;

/* ----- */
    *** INPUT FILES ***;
filename in1 "&dirpath.&scen.00\lines.txt";
filename in2 "&dirpath.&scen.00\itins.txt";
filename in3 "&dirpath.&scen.00\nodes.txt";
filename in4 "&dirpath.&scen.00\network.txt";

    *** OUTPUT FILES ***;
filename out1 "&dirpath.&scen.00\rail.itinerary";
filename out2 "&dirpath.&scen.00\railseg.extatt";
filename out3 "&dirpath.&scen.00\railnode.extatt";
filename out4 "&dirpath.&scen.00\rail.network";
/* ----- */

*** READ IN RAIL HEADER INFORMATION ***;
data routes(drop=futrhwdy c i s1 h1); infile in1 missover dsd;
length descr futrhwdy $20;
input descr linename $ mode $ vehtype headway speed futrhwdy;
linename=substr(linename,1,6); order=0;
c=count(futrhwdy,'); if c>0 then c=c+1; s=0;
if c>0 then do;
do i=1 to c by 2;
s1=scan(futrhwdy,i,');
h1=scan(futrhwdy,i+1,');
if s1<=&scen and s1>s then do; headway=h1; s=s1; end;
end;
end;

*** READ IN RAIL ITINERARY INFORMATION ***;
data itins; infile in2 missover dsd;
input link fpos tpos linename $ layover $ order itina itinb dwcode
zfare ltime dwtime;
linename=substr(linename,1,6);
* ----- *;
    **REPORT LAYOVER PROBLEMS**;
data check; set itins; if layover ne 'X';
proc freq; tables linename / noprint out=check;
data check; set check;
if count>2;
proc print;
title "Too Many Layovers Coded";
* ----- *;

data combine(drop=link fpos tpos); set routes itins;
proc sort; by linename order;
data combine; set combine; by linename;
length desc $22 d $9;
layov=lag(layover); if layov=' ' then layov='X';
name="||compress(linename, " ")||";
desc="||descr||";
if dwcode=1 then d=compress('dwt=#||dwtime,');
else if dwcode=2 then d=compress('dwt=>||dwtime,');
else if dwcode=3 then d=compress('dwt=<||dwtime,');
else if dwcode=4 then d=compress('dwt=+||dwtime,');
else if dwcode=5 then d=compress('dwt=*||dwtime,');
else d=compress('dwt=||dwtime,');

file out1;
if _n_=1 then put "c RAIL TRANSIT BATCHIN FILE FOR SCENARIO &scen.00" /
"c &sysdate" / t lines init;
if first.linename then put 'a' +0 name +2 mode +2 vehtype +2 headway +2 speed
+2 desc / +2 'path=no';
```

```

else if last.linename and layov ne 'X' then put +4 itina d +1 'ttf=1' +2 'lay=' +0 layov / +4 itinb +2 'lay=' +0 layover;
else if last.linename then put +4 itina d +1 'ttf=1' / +4 itinb +2 'lay=' +0 layover;
else if dwcode=1 then put +4 itina +7 d +4 'ttf=1';
else if layov ne 'X' then put +4 itina d +1 'ttf=1' +2 'lay=' +0 layov;
else put +4 itina d +1 'ttf=1';

data combine; set combine;
if order>0;
data combine; set combine;
file out2;
if _n_=1 then put " line inode jnode @ltime @zfare - &sysdate - SEGMENT EXTRA ATTRIBUTES FOR SCENARIO &scen.00";
put +1 linename +1 itina +1 itinb +2 ltime +2 zfare;

*** READ IN NODE INFORMATION ***;
data nodes; infile in3 missover dsd;
length futrspac futrcost $25;
input node label $ pspac pcost futrspac futrcost x y;
if node>&maxzone;
c=count(futrspac,'); if c>0 then c=c+1; s=0;
if c>0 then do;
do i=1 to c by 2;
s1=scan(futrspac,i,');
h1=scan(futrspac,i+1,');
if s1<=&scen and s1>s then do; pspac=h1; s=s1; end;
end;
end;
c=count(futrcost,'); if c>0 then c=c+1; s=0;
if c>0 then do;
do i=1 to c by 2;
s1=scan(futrcost,i,');
h1=scan(futrcost,i+1,');
if s1<=&scen and s1>s then do; pcost=h1; s=s1; end;
end;
end;
proc sort; by node;
* -----*;
**VERIFY THAT EACH NODE HAS A UNIQUE NUMBER**;
proc freq; tables node / noprint out=check;
data check; set check; if count>1;
proc print noobs; var node count;
title "NETWORK &scen.00 NODES WITH DUPLICATE NUMBERS";
* -----*;

data nodes(keep=node label x y); set nodes;
file out3;
if _n_=1 then put " inode @pspac @pcost - &sysdate - NODE EXTRA ATTRIBUTES FOR SCENARIO &scen.00";
put +2 node +2 pspac +2 pcost;

*** READ IN LINK INFORMATION ***;
data network(drop=link modes2 c); infile in4 missover dsd;
input link anode bnode miles modes1 $ modes2 $ dir lanes type;
output;
if dir=2 then do;
c=anode; anode=bnode; bnode=c;
modes1=modes2;
output;
end;
data network; set network;
m=upcase(modes1); if modes1=m then main=1; **identify links transit runs on;
proc sort; by anode bnode;
* -----*;
**VERIFY THAT EACH LINK HAS A LENGTH**;
data check; set network; if miles=0;
proc print;
title "NETWORK &scen LINKS WITHOUT A CODED LENGTH";
* -----*;

*** LIMIT LINKS TO THOSE IN ITINERARIES ***;
data used(keep=anode bnode); set itins;
anode=itina; bnode=itinb;
proc sort; by anode bnode;
data used; set used; by anode bnode;
if first.bnode;

```

```

data network; merge network used (in=hit); by anode bnode;
if main=1 and not hit then delete; ** keep aux. links but remove main links with no service;

data netnodes; set network;
node=anode; output;
node=bnode; output;
proc freq; tables node / noprint out=netnodes;
data netnodes(keep=node); set netnodes;
if node>&maxzone;

data nodes; merge nodes netnodes (in=hit); by node;
if hit;
    * -----*;
    **VERIFY THAT EACH NODE HAS COORDINATES**;
    data check; set nodes; if x='.' or y='.';
    proc print;
    title "NETWORK &scen.00 NODES WITH NO COORDINATES";
    * -----*;

    *** WRITE OUT NETWORK BATCHIN FILE ***;
data nodes; set nodes;
file out4;
if _n_ = 1 then put "c RAIL NETWORK BATCHIN FILE FOR TRANSIT SCENARIO &scen.00" /
    "c &sysdate" / 'c a node x y ui1 ui2 ui3 label' / 't nodes';
put 'a' +2 node +1 x y +2 '0' +2 '0' +2 '0' +1 label;

data network; set network;
file out4 mod;
if _n_ = 1 then put / 't links';
put 'a' +3 anode +2 bnode +2 miles +2 modes1 +2 type +2 lanes +2 '1';

run;

```

Appendix D: Rail Coverage Creation Programs

This appendix briefly describes the steps used in creating the railnet coverage from the Conformity 2006 scenario networks and lists the programs that were used. The programs were used in the following order:

1. **file_check.sas** – Analyzed link and node attributes to verify they are the same between scenarios and created a file containing the *aux_scenario* value for auxiliary links.
2. **create_coverage_data.sas** – Created the link and node attribute files ARC needs to generate a coverage of the rail network.
3. **generate_rail.aml** – Created the railnet coverage with arc and node variables.
4. **add_auxlink_scen.aml** – Added *aux_scenario* to the AAT.
5. **rail_check.sas** – Analyzed itineraries between scenarios and created files listing rail lines that differed between scenarios.
6. **format_itineraries.sas** – Identified unique itineraries among scenarios and created route and itinerary files to import into ARC.
7. **import_rail.aml** – Imported route and itinerary information into ARC to create rail transit coding after calling **import_rail2.sas** to verify coding and perform final formatting.

/* file_check.sas

Craig Heither, rev. 10/26/06

Read in rail files for C06 scenarios 100, 300, 400, 500, 600
and verify node & link attributes do not vary between scenarios.
Run prior to creating rail network coverage. */

```
%let path=M:\proj1\cmh\create_railnet;
%let count=1;

%macro readin;
%do %while (&count le 6);
  data net&count(drop=recid section); infile "&path.transit\&count.00\rail.network" missover;
  length recid $1;
  retain section 0;
  input recid @;
  if recid='c' then delete;
  if recid=' ' then delete;
  if recid='t' then do; section+1; delete; end;
  if recid='a' and section=2 then do;
  input i j miles modes&count $ type&count lanes&count vdf&count;
  in&count=1;

  output net&count;
  end;
proc sort data=net&count; by i j;

data nd&count; infile "&path.transit\&count.00\railnode.extatt" missover firstobs=2;
input node pspac&count pcost&count;
proc sort; by node;

data nt&count(drop=recid section); infile "&path.transit\&count.00\rail.network" missover;
length recid $1;
retain section 0;
input recid @;
if recid='c' then delete;
if recid=' ' then delete;
if recid='t' then do; section+1; delete; end;
if recid='a' and section=2 then do;
input i j miles modes $ type lanes vdf;
output nt&count;
end;
proc sort data=nt&count; by i j;

%if &count=1 %then %let count=%eval(&count+2);
%else %let count=%eval(&count+1);
%end;
%mend readin;
%readin

*** Verify Link attributes match ***;
data net; merge net1 net3 net4 net5 net6; by i j;
mintype=min(type1,type3,type4,type5,type6);
maxtype=max(type1,type3,type4,type5,type6);
minlanes=min(lanes1,lanes3,lanes4,lanes5,lanes6);
maxlanes=max(lanes1,lanes3,lanes4,lanes5,lanes6);
minvdf=min(vdf1,vdf3,vdf4,vdf5,vdf6);
maxvdf=max(vdf1,vdf3,vdf4,vdf5,vdf6);
in=in1+in3+in4+in5+in6;

data test; set net;
if modes1=modes3 and modes3=modes4 and modes4=modes5 and modes5=modes6 then delete;
/* proc print; title 'Modes do not match'; */
data test; set net;
if mintype=maxtype then delete; /* proc print; title 'Types do not match'; */
data test; set net;
if minlanes=maxlanes then delete; /* proc print; title 'Lanes do not match'; */
data test; set net;
if minvdf=maxvdf then delete; /* proc print; title 'VDFs do not match'; */

data net; set net;
length mode $10;
mode=compress(modes1||modes3||modes4||modes5||modes6,');
*** remove duplicate modes;
```

```

recnum = _n_;
do c = 1 to length(mode);
  x1 = substr(mode,c,1);
  output;
end;
proc sort out=temp nodupkey; by recnum x1;

data temp2 (drop=recnum x1 pos); set temp; by recnum;
length newmodes $10.;
retain newmodes; retain pos 1;
substr(newmodes,pos,1) = x1;
pos = pos + 1 ;
if last.recnum then do;
  output;
  pos=1;
  newmodes="";
end;
proc freq; tables newmodes / noprint out=check0;
data check0; set check0; /* proc print; title 'Combined modes'; */

data nt; set nt1 nt3 nt4 nt5 nt6;
/* if vdf=0; proc print; title 'VDF = 0'; */

proc freq data=nt; tables type / noprint out=check1;
proc freq data=nt; tables vdf / noprint out=check2;
proc freq data=nt; tables lanes / noprint out=check3;
proc print data=check1; title 'Type frequency';
proc print data=check2; title 'VDF frequency';
proc print data=check3; title 'Lane frequency';

*** Verify Node attributes match ***;
data nd; merge nd1 nd3 nd4 nd5 nd6; by node;
minspac=min(pspac1,pspac3,pspac4,pspac5,pspac6);
maxspac=max(pspac1,pspac3,pspac4,pspac5,pspac6);
mincost=min(pcost1,pcost3,pcost4,pcost5,pcost6);
maxcost=max(pcost1,pcost3,pcost4,pcost5,pcost6);

proc freq data=nd; tables node / noprint out=check4;
data check4; set check4; if count>1; /* proc print; title 'node number used more than once'; */

data test; set nd;
if minspac=maxspac then delete; /* proc print; title 'pspac changes thru scenarios'; */
data test; set nd;
if mincost=maxcost then delete; /* proc print; title 'pcost changes thru scenarios'; */

*** Auxiliary Links ***;
data all some; set temp2(where=(newmodes ? 'w' or newmodes ? 'v' or newmodes ? 't' or newmodes ? 'r' or
newmodes ? 'd' or newmodes ? 'z' or newmodes ? 'y'));
if in=5 then output all; else output some;
  keep i j newmodes in1 in3 in4 in5 in6 in;

data some1 some2; set some;
if i<j then output some1; else output some2;

data some1; set some1;
length scen $15;
if in1=1 then scen='1';
if in3=1 then scen=compress(scen||'3',' ');
if in4=1 then scen=compress(scen||'4',' ');
if in5=1 then scen=compress(scen||'5',' ');
if in6=1 then scen=compress(scen||'6',' ');

data some2; set some2;
length scen2 $15;
c=i; i=j; j=c;
if in1=1 then scen2='1';
if in3=1 then scen2=compress(scen2||'3',' ');
if in4=1 then scen2=compress(scen2||'4',' ');
if in5=1 then scen2=compress(scen2||'5',' ');
if in6=1 then scen2=compress(scen2||'6',' ');
proc sort; by i j;
data some; merge some1 some2; by i j;

```

```

/* proc print; title 'Aux Links in Some Scenarios'; */

data diff; set some;
if scen ne scen2; /* proc print; title 'Directional difference'; */

data all1 all2; set all;
if i<j then output all1; else output all2;

data all1; set all1;
length scen $15;
if in1=1 then scen='1';
if in3=1 then scen=compress(scen||'3',' ');
if in4=1 then scen=compress(scen||'4',' ');
if in5=1 then scen=compress(scen||'5',' ');
if in6=1 then scen=compress(scen||'6',' ');

data all2; set all2;
length scen2 $15;
c=i; i=j; j=c;
if in1=1 then scen2='1';
if in3=1 then scen2=compress(scen2||'3',' ');
if in4=1 then scen2=compress(scen2||'4',' ');
if in5=1 then scen2=compress(scen2||'5',' ');
if in6=1 then scen2=compress(scen2||'6',' ');
proc sort; by i j;
data all; merge all1 all2; by i j;
/* proc print; title 'Aux Links in all Scenarios'; */

data diff; set all;
if scen ne scen2; /* proc print; title 'Directional difference'; */

data last(keep=i j scen); set all some;
proc sort; by i j;

data links; infile "&path.data\cov_links.txt" missover dsd;
input i j;
match=1;

*** Aux Link Scenario file to import into ARC ***;
data last; merge last (in=hit) links; by i j;
if hit;
file "&path.data\aux_links.txt" dsd;
if match=1 then put i j scen;
else put j i scen;
proc print;

run;

```

/* create_coverage_data.sas

Craig Heither, rev. 10/30/06

Read in rail files for C06 scenarios 100, 300, 400, 500, 600
and format data to generate an ARC coverage. Creates three files:
link attributes, node attributes and link generate file. */

```
%let path=M:\proj1\cmh\create_railnet;
%let count=1;

* ----- *;
*** Output Files ***;
filename out1 "&path.data\nodes.txt";
filename out2 "&path.data\links.txt";
filename out3 "&path.data\arc.gen";
* ----- *;

*** Read in data from all network and node extra attribute files ***;
%macro readin;
%do %while (&count le 6);
data node&count(keep=node x y label) net&count(drop=recid section node x y label j1-j3);
infile "&path.transit\&count.00\rail.network" missover;
length recid $1;
retain section 0;
input recid @;
if recid='c' then delete;
if recid=' ' then delete;
if recid='t' then do; section+1; delete; end;
if recid='a' and section=1 then do;
input node x y j1-j3 label $;
output node&count;
end;
if recid='a' and section=2 then do;
input i j miles modes&count $ type lanes;
output net&count;
end;
proc sort data=net&count; by i j;

data nd&count; infile "&path.transit\&count.00\railnode.extatt" missover firstobs=2;
input node pspac&count pcost&count;
proc sort; by node;

%if &count=1 %then %let count=%eval(&count+2);
%else %let count=%eval(&count+1);
%end;
%mend readin;
%readin

*** Combine all Node data ***;
data nodes; set node1 node3 node4 node5 node6;
proc sort; by node;
data nodes; set nodes; by node;
if first.node;

data ndatt(drop=i); merge nd1 nd3 nd4 nd5 nd6; by node;
array fixmiss{10} pspac1 pcost1 pspac3 pcost3 pspac4 pcost4 pspac5 pcost5 pspac6 pcost6;
do i=1 to 10;
if fixmiss{i}='.' then fixmiss{i}=0;
end;

data nodes; merge nodes ndatt; by node;

data test; set nodes;
length futrspac futrcost $30.;

if pspac3 ne pspac1 then futrspac=compress('3:||pspac3||:');
if pspac4 ne pspac3 then futrspac=compress(futrspac||'4:||pspac4||:');
if pspac5 ne pspac4 then futrspac=compress(futrspac||'5:||pspac5||:');
if pspac6 ne pspac5 then futrspac=compress(futrspac||'6:||pspac6||:');
len=length(futrspac);
if len>1 then substr(futrspac,len)=";
if pcost3 ne pcost1 then futrcost=compress('3:||pcost3||:');
if pcost4 ne pcost3 then futrcost=compress(futrcost||'4:||pcost4||:');
```

```

if pcost5 ne pcost4 then futrcost=compress(futrcost||'5:'||pcost5||':',' ');
if pcost6 ne pcost5 then futrcost=compress(futrcost||'6:'||pcost6||':',' ');
len2=length(futrcost);
if len2>1 then substr(futrcost,len2)="";
if futrspac='' then futrspac='X';
if futrcost='' then futrcost='X';

file out1 dlm=';';
put node label pspac1 pcost1 futrspac futrcost;
proc print;

data icoord(keep=i ix iy); set nodes;
i=node; ix=x; iy=y;

data jcoord(keep=j jx jy); set nodes;
j=node; jx=x; jy=y;

*** Combine all link data ***;
data net; merge net1 net3 net4 net5 net6; by i j;
length mode $10;
mode=compress(modes1||modes3||modes4||modes5||modes6,' ');
** remove duplicate modes;
recnum = _n_;
do c = 1 to length(mode);
  x1 = substr(mode,c,1);
  output;
end;
proc sort out=temp nodupkey; by recnum x1;

data temp2 (drop=recnum x1 pos c modes1 modes3-modes6 mode); set temp; by recnum;
length newmode $10.;
retain newmode; retain pos 1;
substr(newmode,pos,1) = x1;
pos = pos + 1 ;
if last.recnum then do;
  output;
  pos=1;
  newmode="";
end;

*** Create bi-directional links ***;
data link1 link2; set temp2;
if i<j then output link1; else output link2;

data link2(drop=k type lanes newmode); set link2;
k=i; i=j; j=k; type2=type; lanes2=lanes; newmode2=newmode;
output;
proc sort; by i j;

data links(drop=k); merge link1 (in=hit1) link2 (in=hit2); by i j;
if hit1 and hit2 then dir=2;
else if hit1 and not hit2 then dir=1;
else do;
  k=i; i=j; j=k;
  dir=1; type=type2; lanes=lanes2; newmode=newmode2;
  type2='.'; lanes2='.'; newmode2=' ';
end;
proc sort; by i j;

/* data check; set links; if dir=2;
if type ne type2 or lanes ne lanes2 or newmode ne newmode2;
proc print; */

*** Attach coordinates ***;
data centrd; infile "&path.data\centroids.txt" missover dlm=';';
input i ix iy;
proc sort; by i;

data links(drop=type2 lanes2); merge links (in=hit) icoord centrd; by i; if hit;
proc sort; by j;
data links; merge links (in=hit) jcoord; by j; if hit;
proc sort; by i j;

```

```
data links; set links;
id=_n_;
file out2 dlm=';';
put id i j miles newmode newmode2 dir lanes type;

data links; set links end=eof;
file out3;
put @10 id 5. /
    @5 ix iy /
    @5 jx jy /
    @1 'END';
if eof=1 then put @1 'END';

*proc print;
run;
```

```
/* GENERATE_RAIL.AML */
```

```
/* Heither, last revised 10/31/06 */
```

```
/* This AML is used to generate the rail transit coverage with C06 data. */
```

```
/* Call with "&r generate_rail <coverage name>" */
```

```
&args cov
```

```
/******
```

```
&sv dir = M:\proj1\cmh\create_railnet\data\
```

```
/******
```

```
&if [null %cov%] &then &return YOU MUST ENTER A COVERAGE!!
```

```
&if [exists %cov% -cover] &then &return COVERAGE EXISTS!!
```

```
&if not [exists %dir%arc.gen -file] &then &return CREATE GENERATE FILE!!
```

```
&if not [exists %dir%links.txt -file] &then &return CREATE LINK FILE!!
```

```
&if not [exists %dir%nodes.txt -file] &then &return CREATE NODE FILE!!
```

```
&if [exists link.dat -info] &then killinfo link.dat
```

```
&if [exists node.dat -info] &then killinfo node.dat
```

```
/*** GENERATE COVERAGE ***/
```

```
generate %cov%
```

```
input %dir%arc.gen
```

```
lines
```

```
q
```

```
build %cov% line
```

```
build %cov% node
```

```
/*** ADD LINK ATTRIBUTES TO COVERAGE ***/
```

```
tables
```

```
define link.dat
```

```
%cov%-id,5,5,i
```

```
anode,5,5,i
```

```
bnode,5,5,i
```

```
miles,5,5,n,2
```

```
modes1,8,8,c
```

```
modes2,8,8,c
```

```
directions,1,1,i
```

```
lanes,1,1,i
```

```
type,3,3,i
```

```
~
```

```
add from %dir%links.txt
```

```
q
```

```
indexitem %cov%.aat %cov%-id
```

```
indexitem link.dat %cov%-id
```

```
joinitem %cov%.aat link.dat %cov%.aat %cov%-id %cov%-id
```

```
/*** UPDATE NODE NUMBERS AND ADD ATTRIBUTES ***/
```

```
tables
```

```
sel %cov%.nat
```

```
redefine
```

```
5,fnode#,4,5,b
```

```
5,tnode#,4,5,b
```

```
~
```

```
q
```

```
indexitem %cov%.aat fnode#
```

```
indexitem %cov%.aat tnode#
```

```
indexitem %cov%.nat fnode#
```

```
indexitem %cov%.nat tnode#
```

```
relate add
```

```
one
```

```
%cov%.aat
```

```
info
```

```
fnode#
```

```
fnode#
```

```
linear
```

```
ro
```

```
two
```

```
%cov%.aat
```

```

info
tnode#
tnode#
linear
ro
~
ap
asel %cov% arc
resel %cov% node keyfile %cov% arc fnode#
cal %cov% node %cov%-id = one//anode
clearsel
asel %cov% arc
resel %cov% node keyfile %cov% arc tnode#
cal %cov% node %cov%-id = two//bnode
q
relate drop $ALL

tables
define node.dat
%cov%-id,5,5,i
label,4,4,c
pspace,4,4,i
pcost,3,3,i
future_pspace,25,25,c
future_pcost,25,25,c
~
add from %dir%nodes2.txt
sel node.dat
resel future_pspace = 'X'
cal future_pspace = ''
sel node.dat
resel future_pcost = 'X'
cal future_pcost = ''
q

indexitem %cov%.nat %cov%-id
indexitem node.dat %cov%-id
joinitem %cov%.nat node.dat %cov%.nat %cov%-id %cov%-id
addxy %cov% node
projectcopy cover mhn cover %cov%

/** CLEANUP **/
dropindex %cov%.aat
dropindex %cov%.nat
killinfo link.dat
killinfo node.dat
&return AML IS FINISHED.

```

```

/* ADD_AUXLINK_SCEN.AML */
/* Heither, last revised 11/13/06 */

/* This AML is used to import auxiliary link scenario information into coverage. */
/* Call with "&r add_auxlink_scen <coverage name>" */

&args cov

/*****/
&sv dir = M:\proj1\cmh\create_railnet\data\
/*****/

&if [null %cov%] &then &return YOU MUST ENTER A COVERAGE!!
&if not [exists %cov% -cover] &then &return COVERAGE DOES NOT EXIST!!
&if not [exists %dir%aux_links.txt -file] &then &return CREATE AUX LINK FILE!!
&if [exists aux.dat -info] &then killinfo aux.dat
&if [iteminfo %cov% -arc abnode -exists] &then dropitem %cov%.aat %cov%.aat abnode

/**** ADD AUX LINK ATTRIBUTES TO COVERAGE ****/
tables
define aux.dat
  anode,5,5,i
  bnode,5,5,i
  aux_scenario,8,8,c
  ~
add from %dir%aux_links.txt

additem aux.dat abnode 10 10 i
sel aux.dat
cal abnode = anode * 100000 + bnode
additem %cov%.aat abnode 10 10 i
sel %cov%.aat
cal abnode = anode * 100000 + bnode
q

dropitem aux.dat aux.dat anode bnode
indexitem %cov%.aat abnode
indexitem aux.dat abnode
joinitem %cov%.aat aux.dat %cov%.aat abnode abnode

/**** CLEANUP ****/
dropitem %cov%.aat %cov%.aat abnode
dropindex %cov%.aat
killinfo aux.dat
&return AML IS FINISHED.

```

/* rail_check.sas

Craig Heither, rev. 11/1/06

Read in rail files for C06 scenarios 100, 300, 400, 500, 600.

Identify the scenarios each route is in, verify route has same coding between scenarios. */

```
%let path=M:\proj1\cmh\create_railnet;
%let count=1;
%let totobs=0;
*-----;
%let scen1=5; *compare itineraries from this scenario to scen2;
%let scen2=6;

%macro readin;
%do %while (&count le 6);
  *** READ IN & FORMAT ITINERARIES ***;
  data lines&count itins&count; infile "&path.transit\&count.00\rail.itinerary" missover dlm=" ' # + < > *";
  retain name;
  input @1 check $1. @3 check2 $4. @13 check3 $1. @17 check5 $1. @22 check4 $1. @;
  select;
  when (check in ('c','t')) delete;
  when (check='a') do;
    input @1 check $ name $6. mode $ vehtype hdwy&count speed descr $21.;
    descr=compress(descr,"");
    output lines&count; end;
  when (check2='path') delete;
  when (check=' ' and check3=' ' and (check4 ne '#' and check4 ne '+' and check4 ne '<' and
    check4 ne '>' and check4 ne '*') and check5=' ') delete;
  when (check4 in ('#','>','<','+','*')) do;
    input @1 anode&count text1 $ dwt&count text2 $ ttf&count text3 $ us1 text4 $ us2 text5 $ us3;
    output itins&count; end;
  when (check3='T') do;
    input @1 anode&count text1 $ layov&count; output itins&count; end;
  otherwise do;
    input @1 anode&count text1 $ dwt&count text2 $ ttf&count text3 $ us1 text4 $ us2 text5 $ us3
    text6 $ layov&count;
    output itins&count; end;
end;

data lines&count; set lines&count;
  in&count=1;
keep name mode vehtype hdwy&count speed descr in&count;
  proc sort; by name;

data itins&count; set itins&count;
drop mode vehtype hdwy&count speed descr text1-text6 us1-us3 check check2-check5;
  proc sort; by name;

  data itins&count; set itins&count; by name;
  retain order 1;
  order+1;
  if first.name then order=1;
  output;
  proc sort; by name descending order;

data itins&count(drop=layov&count); set itins&count;
  bnode&count=lag(anode&count); layover&count=lag(layov&count);
  proc sort; by name order anode&count bnode&count;
data itins&count; set itins&count; by name;
if last.name then delete;

  *** Read in segment extra attributes ***;
data seg&count(drop=line); infile "&path.transit\&count.00\railseg.extatt" missover firstobs=2;
  length name $6;
  input line $ anode&count bnode&count ltime&count zfare&count;
  name=line;
  proc sort; by name anode&count bnode&count;
proc sort data=itins&count; by name anode&count bnode&count;
  data itins&count; merge itins&count (in=hit1) seg&count (in=hit2); by name anode&count bnode&count;
  proc sort; by name order;

%if &count=1 %then %let count=%eval(&count+2);
```

```

%else %let count=%eval(&count+1);
%end;
%mend readin;
%readin

*** COMBINE ROUTES & SEE WHICH SCENARIOS THEY BELONG TO ***;
data lines; merge lines1 lines3 lines4 lines5 lines6; by name;
file "&path.data\all_railines.txt" dlm=';';
put name;
/* proc print; var name in1 in3 in4 in5 in6; title 'Scenario chart'; */

data check; set lines;
maxh=max(hdwy1,hdwy3,hdwy4,hdwy5,hdwy6);
minh=min(hdwy1,hdwy3,hdwy4,hdwy5,hdwy6);
if maxh=minh then delete;
/* proc print; var name descr hdwy1 hdwy3 hdwy4 hdwy5 hdwy6;
title 'Headways differ between scenarios'; */

*-----;

*** COMPARE 2 SCENARIOS AND IDENTIFY ITINERARY DIFFERENCES ***;
data inboth(keep=name); set lines;
if in&scen1=1 and in&scen2=1;

*** Limit data to only routes in both scenarios ***;
data itins&scen1; merge itins&scen1 (in=hit1) inboth (in=hit2); by name;
if hit1 and hit2;
data itins&scen2; merge itins&scen2 (in=hit1) inboth (in=hit2); by name;
if hit1 and hit2;

data verify; merge itins&scen1 itins&scen2; by name order;
if anode&scen1=anode&scen2 and bnode&scen1=bnode&scen2 and dwt&scen1=dwt&scen2 and ttf&scen1=ttf&scen2 and
layover&scen1=layover&scen2 and ltime&scen1=ltime&scen2 and zfare&scen1=zfare&scen2 then delete;
proc freq; tables name / noprint out=check1;

%macro newfile;
/* data temp; set check1; */
data check1; set check1 nobs=tot; call symput('totobs',left(put(tot,8))); run;

*write file;
%if &totobs gt 0 %then %do;
data check1; set check1;
file "&path.\data\new_in&scen2.00.txt" dlm=';';
put name;
%end;
%mend newfile;
%newfile
*** end of macro ***;

proc print data=check1; var name;
title "Itinerary Differences between scen &scen1.00 and scen &scen2.00";
run;

```

/* format_itineraries.sas

Craig Heither, rev. 11/20/06

Read in rail files for C06 scenarios 100, 300, 400, 500, 600.

Format rail route and itinerary information to import into ARC coverage. */

```
%let path=M:\proj1\cmh\create_railnet\;
%let count=1;
%let cntr=1;

%macro readin;
%do %while (&count le 6);
  *** READ IN & FORMAT ITINERARIES ***;
  data lines&count itins&count; infile "&path.transit\&count.00\rail.itinerary" missover dlm="= '# + < > *";
  retain name;
  input @1 check $1. @3 check2 $4. @13 check3 $1. @17 check5 $1. @22 check4 $1. @;
  select;
  when (check in ('c','t')) delete;
  when (check='a') do;
    input @1 check $ name $6. mode $ vehtype hdwy&count speed descr $21.;
    descr=compress(descr,"");
    output lines&count; end;
  when (check2='path') delete;
  when (check=' ' and check3=' ' and (check4 ne '#' and check4 ne '+' and check4 ne '<' and
    check4 ne '>' and check4 ne '*') and check5=' ') delete;
  when (check4 in ('#','>','<','+', '*')) do;
    input @1 anode&count text1 $ dwt&count text2 $ ttf&count text3 $ us1 text4 $ us2 text5 $ us3;
    output itins&count; end;
  when (check3='1') do;
    input @1 anode&count text1 $ layover&count; output itins&count; end;
  otherwise do;
    input @1 anode&count text1 $ dwt&count text2 $ ttf&count text3 $ us1 text4 $ us2 text5 $ us3
    text6 $ layover&count;
    output itins&count; end;
  end;

  data lines&count; set lines&count;
    length truename $8.;
    in&count=1; truename=name;
  keep truename mode vehtype hdwy&count speed descr in&count check4;
  proc sort; by truename;

  data itins&count; set itins&count;
    length truename $8.;
    truename=name;
  drop name mode vehtype hdwy&count ttf&count speed descr text1-text6 us1-us3 check check2-check3 check5;
  proc sort; by truename;

  data itins&count; set itins&count; by truename;
  retain order 1;
    order+1;
    if first.truename then order=1;
  output;
  proc sort; by truename order;

  *** Read in segment extra attributes ***;
  data seg&count; infile "&path.transit\&count.00\railseg.extatt" missover firstobs=2;
  input truename $ anode bnode ltime zfare;
  proc sort; by truename;
  data seg&count; set seg&count; by truename;
  retain order 1;
    order+1;
    if first.truename then order=1;
  output;

  *** Rename lines ***;
  %if &count>3 %then %do;
    data rename&count; infile "&path.data\new_in&count.00.txt" missover;
  input truename $; newname=compress(truename||"&count",' ');
    %if &count=5 %then %do;
    data rename&count; merge rename4 rename&count; by truename;
  %end;
  %else %if &count=6 %then %do;
```

```

data rename&count; merge rename4 rename5 rename&count; by truename;
%end;

data lines&count(drop=newname); merge lines&count rename&count (in=hit); by truename;
  if hit then truename=newname;
  data itins&count(drop=newname); merge itins&count rename&count (in=hit); by truename;
  if hit then truename=newname;
  data seg&count(drop=newname); merge seg&count rename&count (in=hit); by truename;
  if hit then truename=newname;
%end;

  %if &count=1 %then %let count=%eval(&count+2);
%else %let count=%eval(&count+1);
%end;
%mend readin;
%readin

  *** COMBINE ROUTES & SEE WHICH SCENARIOS THEY BELONG TO ***;
data lines; merge lines1 lines3 lines4 lines5 lines6; by truename;
file "&path.data\all_raillines.txt" dlm=';';
if _n_=1 then put 'name,in100,in300,in400,in500,in600';
put truename in1 in3 in4 in5 in6;
/* proc print; var truename in1 in3 in4 in5 in6; title 'Scenario chart'; */

  *** FORMAT ROUTE HEADERS FOR ARC ***;
data lines(drop=descr); set lines;
length futrhdwy $20 des $22 scen $8;
*** Create headway variable ***;
if hdwy1>0 then hdwy=hdwy1;
else if hdwy3>0 then hdwy=hdwy3;
else if hdwy4>0 then hdwy=hdwy4;
else if hdwy5>0 then hdwy=hdwy5;
else if hdwy6>0 then hdwy=hdwy6;

if in1=1 then scen='1';
if in3=1 then scen=compress(scen||'3',' ');
if in4=1 then scen=compress(scen||'4',' ');
if in5=1 then scen=compress(scen||'5',' ');
if in6=1 then scen=compress(scen||'6',' ');

if hdwy3>0 and hdwy3 ne hdwy then futrhdwy=compress('3'||hdwy3||':',' ');
if hdwy4>0 and hdwy4 ne hdwy and hdwy4 ne hdwy3 then futrhdwy=compress(futrhdwy||'4'||hdwy4||':',' ');
if hdwy5>0 and hdwy5 ne hdwy and hdwy5 ne hdwy4 then futrhdwy=compress(futrhdwy||'5'||hdwy5||':',' ');
if hdwy6>0 and hdwy6 ne hdwy and hdwy6 ne hdwy5 then futrhdwy=compress(futrhdwy||'6'||hdwy6||':',' ');
len=length(futrhdwy);
if len>1 then substr(futrhdwy,len)='';
if futrhdwy=' ' then futrhdwy='X';
ord=_n_;
  descr=left(descr); des=""||trim(descr)||"";
/* proc print; var truename hdwy1 hdwy3 hdwy4 hdwy5 hdwy6 hdwy futrhdwy scen; */
*-----*;

  *** ANALYZE SCENARIO ITINERARIES ***;
** build from scen 1 and add new lines from each successive scenario;
data it1; merge itins1 itins3 (in=hit); by truename order;
if hit;
if anode1=anode3 and dwt1=dwt3 and layover1=layover3 then delete;
proc freq; tables truename / noprint out=keep3;

data it1; merge itins3 itins4 (in=hit); by truename order;
if hit;
if anode3=anode4 and dwt3=dwt4 and layover3=layover4 then delete;
proc freq; tables truename / noprint out=keep4;

data it1; merge itins4 itins5 (in=hit); by truename order;
if hit;
if anode4=anode5 and dwt4=dwt5 and layover4=layover5 then delete;
proc freq; tables truename / noprint out=keep5;

data it1; merge itins5 itins6 (in=hit); by truename order;
if hit;
if anode5=anode6 and dwt5=dwt6 and layover5=layover6 then delete;
proc freq; tables truename / noprint out=keep6;

```

```

** combine itineraries into final dataset **;
data three; merge itins3 keep3 (in=hit); by truename; if hit;
data four; merge itins4 keep4 (in=hit); by truename; if hit;
data five; merge itins5 keep5 (in=hit); by truename; if hit;
data six; merge itins6 keep6 (in=hit); by truename; if hit;

data all(keep=truename order anode dwt layover check4); set itins1 three four five six;
anode=max(anode1,anode3,anode4,anode5,anode6);
dwt=max(dwt1,dwt3,dwt4,dwt5,dwt6);
layover=max(layover1,layover3,layover4,layover5,layover6);
proc sort; by truename descending order;

data all(drop=layover); set all;
  bnode=lag(anode); lay=lag(layover);
proc sort; by truename order anode bnode;
data all; set all; by truename;
if last.truename then delete;

** combine segment extra attributes into final dataset **;
data s3; merge seg3 keep3 (in=hit); by truename; if hit;
data s4; merge seg4 keep4 (in=hit); by truename; if hit;
data s5; merge seg5 keep5 (in=hit); by truename; if hit;
data s6; merge seg6 keep6 (in=hit); by truename; if hit;
data allseg(drop=count percent); set seg1 s3 s4 s5 s6;
proc sort; by truename order anode bnode;

data all; merge all (in=hit1) allseg (in=hit2); by truename order anode bnode;
length layt $10;
if hit1 and hit2;
  layt=compress(lay||' ');
if layt='.' then layt='X';
  if check4='#' then dcode=1;
  else if check4='>' then dcode=2;
  else if check4='<' then dcode=3;
  else if check4='+' then dcode=4;
  else if check4='*' then dcode=5;
  else dcode=0;
proc print;

proc summary nway; var dwt ltime zfare; output out=check max=;
proc print;
  proc freq data=all; tables dwt / noprint out=junk; proc print;
  proc freq data=all; tables lay / noprint out=junk; proc print;
  proc freq data=all; tables ltime / noprint out=junk; proc print;
  proc freq data=all; tables zfare / noprint out=junk; proc print;

** write out files to import into ARC **;
data header; set lines;
file "&path.data\rail_route1.csv" dlm=';';
put truename des mode vehtype hdwy speed scen futrhdwy;

data path; set all;
file "&path.data\rail_itin1.csv" dlm=';';
put truename anode bnode order layt dcode zfare ltime dwt;

run;

```

```

/* IMPORT_RAIL.AML */
/* Heither, last revised 11/8/06 */
/* Call with "&r import_rail <cov> <route> <rte file> <itin file>" */

/* ***** */
/* THIS PROGRAM IS USED TO IMPORT RAIL ITINERARIES INTO ARC. */
/* THE COVERAGE & ROUTE WHERE THE RAIL CODING WILL BE STORED ARE REQUIRED */
/* ALONG WITH THE HEADER AND ITINERARY FILES. */
/* ***** */

&args cov route file1 file2

/*#####*/
/* CHANGE INFO HERE */
&sv dir = C:\build_networks\programs\create_railcov\ /* program path */
&sv datadir = M:\proj1\cmh\create_railnet\data\ /* data file path */
/*#####*/
&sv d = $
&sv count = 1

/** HOUSEKEEPING **/
&if [null %cov%] &then &return YOU MUST ENTER A COVERAGE NAME.
&if [null %route%] &then &return YOU MUST ENTER A ROUTE SYSTEM.
&if not [exists %cov%.aat -info] &then &return COVERAGE DOES NOT EXIST!
&if not [exists %datadir%%file1% -file] &then &return ROUTE HEADER FILE DOES NOT EXIST!
&if not [exists %datadir%%file2% -file] &then &return ITINERARY FILE DOES NOT EXIST!
&if [exists import_rail2.lst -file] &then &do
&if [delete import_rail2.lst] = 0 &then &type FILE CLEANUP
&end
&if [exists route.new -info] &then killinfo route.new
&if [exists one.new -info] &then killinfo one.new
&if [exists itinerary.new -info] &then killinfo itinerary.new
&if [exists %cov%.stp -info] &then killinfo %cov%.stp
&if [exists one.stp -info] &then killinfo one.stp
&if [exists %cov%.sectranpath -info] &then dropfeatures %cov% section.tranpath
&if [iteminfo %cov% -arc abnode -exists] &then dropitem %cov%.aat %cov%.aat abnode ft_imp tf_imp arclink#

/** SELECT DATA & WRITE TO FILES FOR SAS PROCESSING & CODING VERIFICATION **/
tables
sel %cov%.aat
unload %datadir%links.txt anode bnode directions modes1 modes2 init
q

&type * * * * *
&type * RUNNING SAS *
&type * * * * *
"C:\Program Files\SAS\SAS System\9.0\sas.exe" -sysin %dir%import_rail2.sas -sysparm %datadir%%d%%file1%%d%%file2%
&if [exists import_rail2.lst -file] &then &return FIX RAIL ROUTE CODING.

/** IMPORT DATA & BUILD ROUTES **/
tables
define route.new
transit_line,8,8,c
description,20,20,c
mode,1,1,c
vehicle_type,1,1,i
headway,5,5,n,2
speed,4,4,n,1
scenario,8,8,c
futrhdwy,20,20,c
~
add from %datadir%%file1%
define itinerary.new
transit_line,8,8,c
anode,5,5,i
bnode,5,5,i
layover,8,8,c
dwell_code,1,1,i
zone_fare,7,7,n,2
line_serv_time,8,8,n,5
dwell,5,5,n,2
order,3,3,i
route,3,3,i

```

```

place,5,5,n,0
abnode,10,10,n,0
~
add from %datadir%itin.txt
define %cov%.stp
%cov%-id,5,5,n,0
order,5,5,n,0
route,5,5,n,0
~
add from %datadir%path.txt

sel route.new
additem route.new link 4 4 i
  cal link = $recno
sel itinerary.new
resel layover = 'X'
  cal layover = ''
sel route.new
resel futrhdwy = 'X'
  cal futrhdwy = ''
sel itinerary.new
additem itinerary.new code 8 8 n 0
  cal code = route * 10000 + place
sel %cov%.aat
additem %cov%.aat abnode 10 10 n 0
  cal abnode = anode * 100000 + bnode
/**** ADD IMPEDANCE VARIABLE TO PROHIBIT USE OF UNDESIRABLE LINKS ****/
/**** (NEGATIVE VALUES MEAN LINK CANNOT BE TRAVERSED) ****/
additem %cov%.aat ft_imp 5 5 n 2
additem %cov%.aat tf_imp 5 5 n 2
  cal ft_imp = -1
  cal tf_imp = -1
additem %cov%.aat arclink# 4 5 b
  cal arclink# = %cov%#
q

/** DETERMINE NUMBER OF ROUTES TO CREATE **/
ae
edit route.new info
sel all
cursor open
&sv max = %:edit.AML$NSEL%
cursor close
q; n

/** ITERATE THROUGH ROUTES AND BUILD THEM **/
/** This was added to build each individually to eliminate any chance of path building error **/
&do &while %count% le %max%
ap
  asel itinerary.new info
  resel itinerary.new info route = %count%
  infofile itinerary.new info one.new init
  resel %cov% arc keyfile one.new info abnode
  cal %cov% arc ft_imp = miles
  resel %cov% arc directions gt 1
  &sv flag = [extract 1 [show select %cov% arc]]
  &if %flag% gt 0 &then cal %cov% arc tf_imp = miles
  asel %cov% arc ft_imp gt 0

  asel %cov%.stp info
  resel %cov%.stp info route = %count%
  infofile %cov%.stp info one.stp init

  netcover %cov% tranpath
  impedance ft_imp tf_imp
  stops one.stp order route
  path stops

  asel %cov% arc
  cal %cov% arc ft_imp = -1
  cal %cov% arc tf_imp = -1
  q
&sv count = %count% + 1

```

```

&end

/** ADD DATA TO SECTION AND ROUTE TABLES **/
tables
&if not [iteminfo %cov% -route.tranpath transit_line -exists] &then &do
  additem %cov%.ratranpath transit_line 8 8 c
  additem %cov%.ratranpath description 20 20 c
  additem %cov%.ratranpath mode 1 1 c
  additem %cov%.ratranpath vehicle_type 1 1 i
  additem %cov%.ratranpath headway 5 5 n 2
  additem %cov%.ratranpath speed 4 4 n 1
  additem %cov%.ratranpath scenario 8 8 c
  additem %cov%.ratranpath future_headway 20 20 c
&end
&if not [iteminfo %cov% -section.tranpath transit_line -exists] &then &do
  additem %cov%.sectranpath transit_line 8 8 c
  additem %cov%.sectranpath itinerary_a 5 5 i
  additem %cov%.sectranpath itinerary_b 5 5 i
  additem %cov%.sectranpath order 3 3 i
  additem %cov%.sectranpath layover 8 8 c
  additem %cov%.sectranpath dwell_code 1 1 i
  additem %cov%.sectranpath zone_fare 7 7 n 2
  additem %cov%.sectranpath line_serv_time 8 8 n 5
  additem %cov%.sectranpath dwell_time 5 5 n 2
&end
&if not [iteminfo %cov% -section.tranpath code -exists] &then additem %cov%.sectranpath code 8 8 n 0
sel %cov%.sectranpath
  cal code = routelink# * 10000 + tranpath#
q

indexitem %cov%.aat %cov%#
indexitem %cov%.sectranpath arclink#
indexitem %cov%.sectranpath code
indexitem %cov%.sectranpath routelink#
indexitem %cov%.ratranpath tranpath#
indexitem %cov%.ratranpath transit_line
indexitem itinerary.new code
indexitem route.new transit_line

ap
relate add
one
  %cov%.aat
  info
  arclink#
  %cov%#
  linear
  ro
two
  itinerary.new
  info
  code
  code
  linear
  ro
three
  %cov%.sectranpath
  info
  tranpath#
  routelink#
  linear
  ro
four
  route.new
  info
  transit_line
  transit_line
  linear
  ro
~

asel %cov% section.tranpath
cal %cov% section.tranpath itinerary_a = one//anode

```

```

cal %cov% section.tranpath itinerary_b = one//bnode
resel %cov% section.tranpath f-pos gt t-pos
&sv flag = [extract 1 [show select %cov% section.tranpath]]
&if %flag% gt 0 &then &do
  cal %cov% section.tranpath itinerary_a = one//bnode
  cal %cov% section.tranpath itinerary_b = one//anode
&end
asel %cov% section.tranpath
cal %cov% section.tranpath transit_line = two//transit_line
cal %cov% section.tranpath layover = two//layover
cal %cov% section.tranpath order = two//order
cal %cov% section.tranpath dwell_code = two//dwell_code
cal %cov% section.tranpath zone_fare = two//zone_fare
cal %cov% section.tranpath line_serv_time = two//line_serv_time
cal %cov% section.tranpath dwell_time = two//dwell
asel %cov% route.tranpath
cal %cov% route.tranpath transit_line = three//transit_line
cal %cov% route.tranpath description = four//description
cal %cov% route.tranpath mode = four//mode
cal %cov% route.tranpath vehicle_type = four//vehicle_type
cal %cov% route.tranpath headway = four//headway
cal %cov% route.tranpath speed = four//speed
cal %cov% route.tranpath scenario = four//scenario
cal %cov% route.tranpath future_headway = four//futrhdwy

relate drop $ALL
q
dropitem %cov%.sectranpath %cov%.sectranpath code

/** REMOVE TRANSIT LINES BEING UPDATED. **/
ae
&if [exists %cov%.sec%route% -info] &then &do
  edit %cov% route.%route%
  arcplot asel route.new info
  arcplot resel %cov% route.%route% keyfile route.new info transit_line
  selectget
  &sv flag = [extract 1 [show number selected]]
  &if %flag% gt 0 &then &do
    delete
    save
  &end
&end

/** PUT TRANSIT ROUTES IN FINAL LOCATION ***/
edit %cov% route.tranpath
sel all
duplicate %route%
&if [exists %cov%.sec%route% -info] &then y
save
q

/** CLEAN UP **/
tables
dropitem %cov%.aat abnode ft_imp tf_imp arclink#
sel %cov%.rat%route%
cal %route%-id = %route%#
q
&type * * * * *
&type * DELETING TEMPORARY FILES *
&type * * * * *
killinfo itinerary.new
killinfo route.new
killinfo one.new
killinfo %cov%.stp
killinfo one.stp
&if [delete %dir%itin.txt] = 0 &then &type Successful Deletion of ITIN.TXT
&if [delete %dir%path.txt] = 0 &then &type Successful Deletion of PATH.TXT
&if [delete %dir%section.txt] = 0 &then &type Successful Deletion of SECTION.TXT
&if [delete %dir%links.txt] = 0 &then &type Successful Deletion of LINKS.TXT
dropfeatures %cov% section.tranpath
dropindex %cov%.aat

&return AML IS FINISHED

```

/* import_rail2.sas

Heither, last revised 11/20/06

PROGRAM IS CALLED BY IMPORT_RAIL.AML TO
IMPORT RAIL ITINERARIES INTO ARC.

```
----- */
%let dir=%scan(&sysparm,1,$);
%let rte=%scan(&sysparm,2,$);
%let itin=%scan(&sysparm,3,$);
/*-----*/
  * INPUT FILES *;
filename in1 "&dir.&itin";
filename in2 "&dir.&rte";
filename in3 "&dir.links.txt";
  * OUTPUT FILES *;
filename out1 "&dir.itin.txt";
filename out2 "&dir.path.txt";
filename out3 "&dir.&rte";
/*-----*/

** READ IN CODING FOR RAIL ROUTES **;
data section; infile in1 missover dsd;
input line $ anode bnode ord layover $ dwcode zfare ltime dwt;
proc sort; by line ord;

data section; set section;
if ltime='.' then ltime=0;
group=lag1(line);

data verify; set section; proc sort; by anode bnode;

  ** READ IN ROUTE TABLE CODING **;
** This will ensure description is enclosed in single quotes for ARC **;
data rte; infile in2 missover dlm=';';
length desc $22. d futrhdwy $20.;
input line $ desc $ mode $ type hdwy spd scen $ futrhdwy $;
d=compress(desc,"");
desc="||trim(d)||";
proc sort; by line;

** Replace File for ARC **;
data rte; set rte;
file out3 dlm=';';
put line desc mode type hdwy spd scen futrhdwy;

  *-----*;
  ** VERIFY CODING **;
  *-----*;
** Read in coverage Links **;
data mhn; infile in3 missover dlm=';';
input anode bnode dir modes1 $ modes2 $;

data mhn(drop=c); set mhn(where=(modes1 ? 'C' or modes1 ? 'M' or modes2 ? 'C' or modes2 ? 'M'));
match=1; output;
if dir>1 then do;
c=anode; anode=bnode; bnode=c;
output;
end;
proc sort; by anode bnode;

** Find Unmatched Links **;
data check; merge verify (in=hit) mhn; by anode bnode;
if hit;
if match=1 then delete;
proc print; var anode bnode line ord;
title 'MIS-CODED ANODE-BNODE OR DIRECTIONAL PROBLEM ON THESE LINKS';

** Ensure Transit Not Coded on Centroid Links **;
data bad; set verify;
if anode le 1891 or bnode le 1891;
proc print; var anode bnode line ord;
title 'TRANSIT CODING ON CENTROID CONNECTORS';
```

```

*-----*;
** FORMAT ITINERARY DATASET **;
*-----*;
data section(drop=ord); set section;
retain order 1;
order+1;
if line ne group then order=1;
output;

data section; set section;
retain route 0;
if line ne group then route+1;
output;

data section; set section;
place=_n_;
proc sort; by anode bnode;

data arcs; infile in3 missover dlm=" ";
input anode bnode;
true=1;
proc sort; by anode bnode;

** Find True Arc Direction in MHN **;
data section; merge section (in=hit) arcs; by anode bnode;
if hit;
if true=1 then abnode=anode*100000+bnode;
else abnode=bnode*100000+anode;
proc sort; by line order;

*-----*;
** WRITE ITINERARY FILE **
*-----*;
data writeout; set section;
file out1 dlm=';';
put line anode bnode layover dwcode zfare ltime dwt order route place abnode;

* _ _ _ _ _ _ _ _ _ _ *;
**REPORT ITINERARY GAPS**;
**THESE ARE MIS-CODED LINKS **;
data check; set section;
z=lag1(bnode);
if anode ne z and order>1 then output;
proc print; var line order anode bnode z;
title 'Itinerary Gaps';

**REPORT LAYOVER PROBLEMS**;
**A MAXIMUM OF TWO LAYOVERS ARE ALLOWED PER TRANSIT LINE **;
data check; set section; if layover ne 'X';
proc freq; tables line / noprint out=check;
data check; set check;
if count>2;
proc print; var line count;
title 'Too Many Layovers Coded';
* _ _ _ _ _ _ _ _ _ _ *;
*-----*;
** FORMAT PATH-BUILDING FILE FOR ARC **
*-----*;
data path(keep=node route); set section; by line order;
node=anode; output;
if last.line then do;
node=bnode; output;
end;
data path; set path;
rank=_n_;

*-----*;
** WRITE PATH FILE **
*-----*;
data write2; set path;
file out2 dlm=';';
put node rank route;
run;

```