

***Wasatch Front Travel Demand Model***

**Version 7.0.2a**

**‘What’s New’ Documentation**

**2009 Base Year Travel Demand Model**

**Updated:**

**June 2012**

# Version 7.0.2a

Changes in this version of the model affect model run time as explained below. Model results remain unchanged.

## Cube Cluster

Cube Cluster was added to improve model run times. Cluster allows model processes to be distributed over multiple computer processors, thus dividing the amount of work done on any one processor and reducing run time. Depending on the number of processing nodes over which the model will be distributed, the run time can be greatly reduced.

### Implementation of Cluster & Model Run Time Reduction

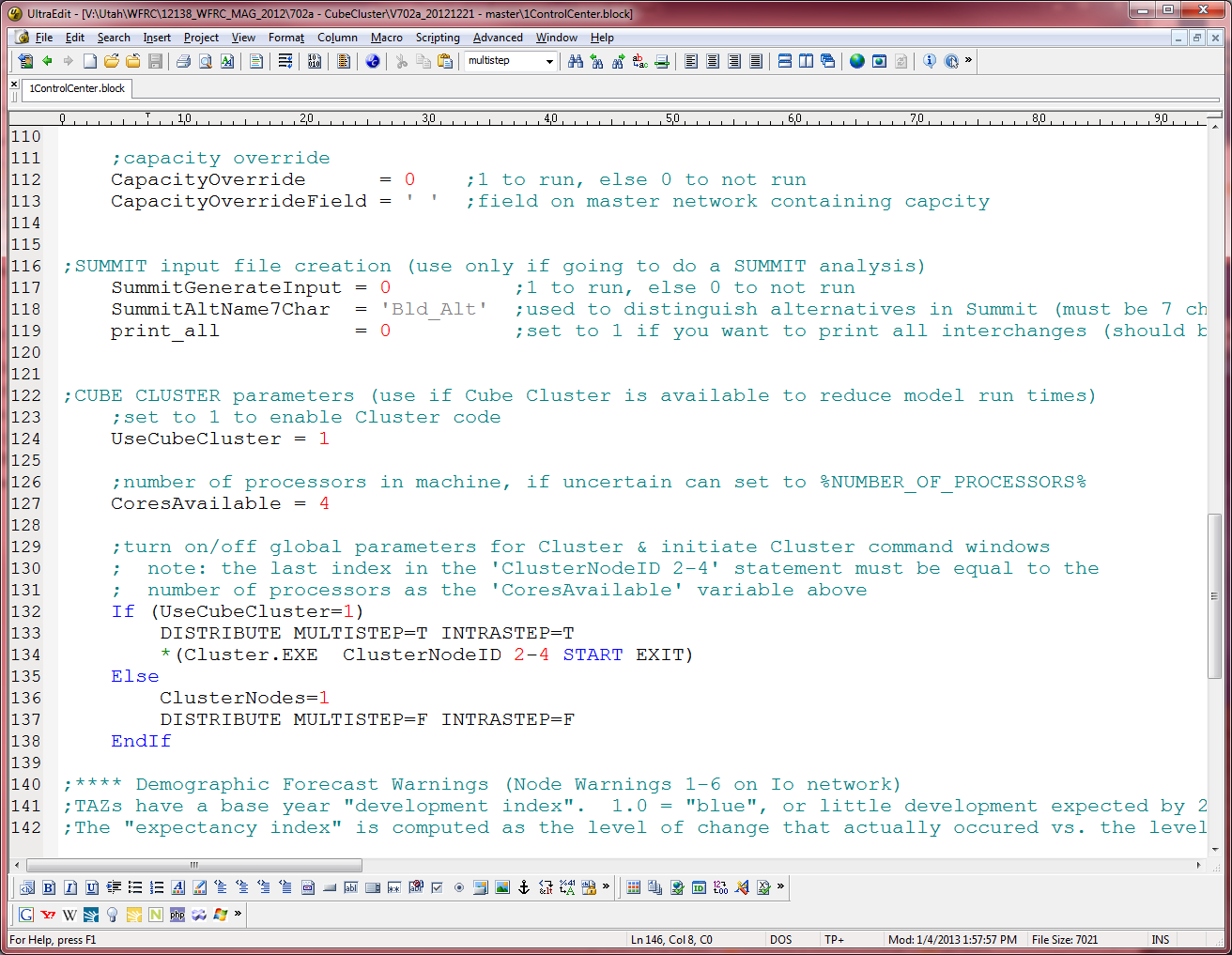
Not all parts of the model are able to use Cluster. The following graphic illustrates which scripts (shown in green) have Cluster coded into the model.



Because some parts of the model are not distributed over multiple processors, there is a minimum time the model will take to run and at which point adding more processors does not significantly improve model run times. A test of v7.0.2a on a computer with 32 processors showed the following model run time improvements when adding subsequent processing nodes:

### Setting Cluster Controls

The following controls were added to the ‘1ControlCenter.block’ file to allow the user the ability to turn on/off Cluster and to set the Cluster parameters (note, Citilabs charges an additional fee for Cluster, so the model was made to run with or without Cluster):



To turn on Cube Cluster, set the ‘UseCubeCluster’ variable to 1. If the variable is set to any other number the Cluster code in the model is ignored. The default is set to not run Cluster.

The ‘CoresAvailable’ variable identifies the number of extra processing nodes over which the model processing will be distributed. Typically this is set to the number of processors, or processing cores, available in a given machine. If the user wishes to not use all the available processors available in a computer, then this variable would be set to the number of processing cores the user wants Cluster to use. The user must also change the second index in the ‘ClusterNodeID 2-4’ statement (in this example the index is 4), to be the number of cores specified in the ‘CoresAvailable’ variable. If only 1 processing node is to be used, then Cluster should be turned off.

The ‘If’ statement is the last control and directs Cube to either turn Cluster on/off. If Cluster is to be turned on, then the ‘Cluster Node Management‘ window is initiated which sets number of Voyager processing nodes in wait mode. A control was also added to the ‘HailMary.s’ script which closes these additional Voyager processing node windows when the model has completed running.

## Elapsed Time Scripts

To be able to better track Cluster results script by script, an elapsed time text file is printed for each script in the model. The code that prints the elapsed time text file was updated was also updated so time stamps are printed in a similar fashion in each script.

# Version 7.0.2

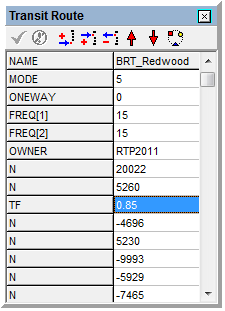
Changes in this version of the model primarily affect the modeling of modes 5 and 9, which have been modified to reflect BRT-Light (e.g. BRT I) and BRT-Heavy (e.g. BRT III), respectively.

## Modes 5 and 9

For the UTA Network Study there was a desire by the study team to better differentiate between classes of BRT using the travel model. The decision was made to revise the model so that mode 5 would represent a lower class of BRT, similar to BRT I, which is generally characterized by improved passenger stations, off-board fare payment, and transit signal priority. Mode 9 was modified to represent a higher class of BRT, similar to BRT III, which includes all of the features of the lower class of BRT, but also includes substantial route segments with exclusive bus lanes.

### BRT Speeds

The biggest difference from prior model versions is how the speed of BRT routes is coded. Previously, the model assumed that BRT would operate at 80% of the vehicle speed. Now it is up to the user to specify the desired operating speed. If the user neglects to specify the speed of the BRT route it will operate at the same speed as the local bus (approximately 60% of the vehicle speed, depending on the facility type of the road).

The user specifies the speed of the BRT route in the transit line file and should typically be via using the ‘TF’ function (see figure). The ‘TF’ function is a time factor that adjusts the transit travel time to the nodes following the ‘TF.’ A time factor less than 1.0 will increase the transit speed, while a factor greater than 1.0 will decrease the speed. As a general rule, it is recommended that the following time factors be used for modes 5 and 9.

|  |  |
| --- | --- |
| **Mode** | **Time Factor** |
| BRT Light (Mode 5) | 0.85 |
| BRT Heavy (Mode 9) | 0.75 |

These factors will result in BRT-Light speeds that are approximately 70% of the vehicle speed and BRT-Heavy speeds that are approximately 80% of the vehicle speed.

This approach to BRT speeds allows maximum flexibility in setting the speeds, which can vary by segment. For example, if a route is coded as BRT Heavy because it is expected to be predominantly exclusive bus lanes, but with short segment of shared lanes with no other improvements, it is possible to vary the speeds by segment by coding the exclusive lane segments with a ‘TF’ of 0.75 and the shared lane segment with a 1.0.

The user may also use the ‘SPEED’ control in the line file to set the speed for the exclusive lane segments. ‘SPEED’ sets the link speed to a fixed value allowing the BRT line to run independent of adjacent roadway congestion. When the exclusive lane segment ends, a ‘TF’ control can be used to reset the BRT speed to once again be subject to roadway congestion.

### Mode Specific Constants

The mode specific constant refers to the general attractiveness of each mode. Rail modes have the highest values while local bus has the lowest. Previously modes 5 and 9 had a constant that was halfway between local bus and light rail. These constants were modified so that mode 5 now has a constant of 33% of the difference between local bus and light rail and mode 9 has a constant of 67% of the difference.

## Hand Coded Access Links

The model was modified so that new hand coded walk and drive access links can be added to transit scenarios. This was done by separating the walk and drive access links that remain unchanged regardless of transit from those that are specific to a given scenario. General walk and drive access link files were placed in the ‘\_Inputs\4\_Transit\\_General Hand-Coded Support Links’ directory. Scenario specific walk and drive access links can be added to the ‘scenario\_hand\_coded\_walk\_links.sup’ and ‘scenario\_hand\_coded\_drive\_links.sup’ files found in the scenario line folders.

## Other Minor Modifications

Two other minor modifications were made to the model. The ‘1DemographicsAnalysis.s’ script was modified to improve the reporting of the city-level demographics. The ‘1NetProcessor.s’ script was modified so that the name and values of the various districts are added to the network rather than the generic ‘DISTLRG,’ ‘DISTMED,’ and ‘DISTSML’ labels.

# Version 7.0.1

Changes in this version of the model affect usability and performance but do not affect model results. Below is a description of the changes made in this model version.

## HailMary.s

With the release of 64 bit operating systems, such as in Windows 7, Cube software is installed in the program files x86 directory. This change broke the paths in the ‘HailMary.bat’ file to the Voyager executable. Rather than maintain two hailmary batch files (one for 32 bit operating systems and one for 64 bit systems), the model run launch point will now be done using a Cube script (HailMary.s).

The ‘HailMary.s’ launch point offers a few advantages:

* Running the ‘HailMary.s’ script should work regardless of where Cube software is installed.
* Error handling for the entire model stream is handled in Voyager. Previously, the ‘HailMary.bat’ included error handling between model scripts while Voyager handled errors inside the scripts.

A text file indicating the step of the model is still generated as the model runs. These files are now found in the root directory which helps the user see the progress of the model in one location. If the model were to crash, this file also informs the user the script where the crash occurred.

The model run print file (\*.PRN) is also found in the root directory and contains information from the entire model run. This has the advantage of not having to hunt for this file in the scripts folder to identify model fatal errors and other information. However, this file can be large, which is a drawback. The aggregate runtime reported in the Voyager run window also represents the entire model runtime and not simply the runtime for an individual script.

* Viewing and modifying the ‘HailMary.s’ script is easier and uses the same logic/rules the user is familiar with in running Voyager scripts. The user does not need to be familiar with DOS to manipulate scripts in the model run.
* The model now calls the ‘0GeneralParameters.block’ and ‘1ControlCenter.block’ files only once at the beginning of the model run. If a user still would like to run individual scripts, the user will simply need to uncomment the READ statements at the beginning of the individual script to read in the general and scenario specific parameters. When finished, the user will then re-comment out these READ statements.

## Elapsed Time

The ‘ElapsedTime.exe’ file no longer worked with Windows 7. The elapsed time code was removed from the model and replaced with a new method for monitoring model step runtimes. Code was added inside the scripts to print a text file at the beginning of various model steps. The model runtime can then be inferred by looking at the text file time stamps. Just one file is created at the beginning of the Input Processing, Auto Ownership and Trip Generation modeling steps. Beginning with Distribution, a text file is create for each script in the modeling step, with the exception of Mode Choice where scripts 2-3 do not generate a time stamp text file because the model runs fairly quickly through these scripts. The time stamp files are found in the ‘\_ElapsedTime’ folder in the root directory.

## Model Notification (Play Sound & Send Mail/Text)

### Send Mail / Text

The ‘HailMary.s’ script will now send the user and email alert when the model finishes or if the model crashes. To receive emails alerts, the user needs to specify the SMTPSERVER, USERNAME, PASSWORD, FROM and TO sections in the SENDMAIL command (note there are two places in ‘HailMary.s’ that would need to be changed). For a user not familiar with defining email parameters, most of this information may be obtained from looking at the email settings in an email enabled smart phone or by contacting the agency’s IT department.

SENDMAIL SMTPSERVER = 'webmail.rsginc.com',

USERNAME = 'i-rsg\chad',

PASSWORD = '\*\*XYyRODsGKVEGOpPVlHOPDVVsPElStCOuLIMDIqBatCQTU\*\*',

FROM = 'chad.worthen@rsginc.com',

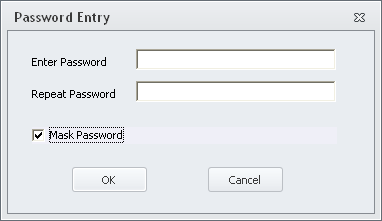
TO = 'chad.worthen@rsginc.com',

Subject = 'Model Finished',

Message = 'Congratulations!'

The user may also use the Cube Base text editor to encrypt the user’s PASSWORD (highlighted in red above).

* Open any text file in Cube Base.
* In the ‘Insert’ menu option or ribbon command, select ‘Password for email’.
* A dialog box opens which allows the user to enter a password (make sure ‘Mask Password’ is checked). The encrypted text will be entered into the text document in Cube.



* Copy the encrypted password text and paste into the PASSWORD variable (text between the quote marks) in ‘HailMary.s’.

According to Cube help, the user can also send mail as a text message to a cell phone account that supports text messaging. Examples of cell phone provider messaging e-mail addresses are listed below (check with cell phone service provider to verify the appropriate address):

* Phone#@tmomail.net
* Phone#@ messaging.sprintpcs.com
* Phone#@ vtext.com
* Phone#@ txt.att.net

For instance, a text message address might be added to the TO command by typing the following text:

TO = 'chad.worthen@rsginc.com', '5106635200@messaging.sprintpcs.com',

To turn email alerts off set the ‘SendEmailAlerts’ variable to 0 in ‘1ControlCenter.txt’.

## Input/Output File Path Names in Quotes

The file path names in the model are now encapsulated in quote marks. This allows the path to include special characters, such as spaces, dashes and andpersands without crashing the model. This allows the user greater flexibility in file and folder name construction as well as greater freedom in placing the model root directory.

## Update to Voyager Module References

Previous versions of the model continued to use the following legacy TP+ module calls: HWYNET, HWYLOAD and TRIPDIST. These have been updated to the following Voyager module calls: NETWORK, HIGHWAY and DISTRIBUTION. This should allow the model to take advantage of module updates as Citilabs is maintaining the Voyager modules but has not guaranteed to update the legacy TP+ modules. The only remaining TP+ module in the WF model is the TRNBUILD module which should be replaced in V71 of the model.

## General Updates

### Network Processing Bug Fixes

A bug was fixed that was causing the model to freeze in the ‘1NetProcessor.s’ script. The bug only affected land use reporting and did not affect model results.

There was an inconsistency in how the hand-coded drive and walk support links were being called in the model. In the ‘4TranRouteTest\_0Xfer\_Emp30.s’ script, the support links were being read in a ‘Sup’ folder in the scenario transit line directory. All other references to hand coded access links were using the ‘Sup’ folder in the general inputs transit folder. Both locations had copies of the same hand-coded access link data. The ‘4TranRouteTest\_0Xfer\_Emp30.s’ script now references the same support link directory as the rest of the model. The duplicate ‘Sup’ folder in the line folder can be deleted. Scenario specific hand-coded support links will be included in a later version of the WF model.

### General Parameters Edits

The ‘runselectlink’ variable was added as a toggle to turn on select link analysis in Final Assignment. The corresponding application of the toggle variable is found in ‘4AssignHwy\_ManagedLanes.s’.

The post processing sub folders were removed as tokens. These folder tokens were not being used and were labeled incorrectly. The sub folders are referenced with a static path name after the @ParentDir@@PostProcDir@ tokens in the post processing scripts called by ‘HailMary.s’