Advanced Aviation Simulation Techniques

by AirportTools, Inc.

Document Contents

This document presents the following topics:

- Metering utilizing SIMMOD.
- Metering utilizing Aeroscript.

SIMMOD Metering

SIMMOD metering provides two unique capabilities to the aviation analyst:

- The first ability allows the analyst to separate multiple aircraft prior to their reaching a common airspace node.

  For each metered route the analyst supplies a meter node.

  Downstream from these metered nodes should be a common meter post node.

  For each metered aircraft SIMMOD computes the nominal travel time from the meter node to the meter post node.

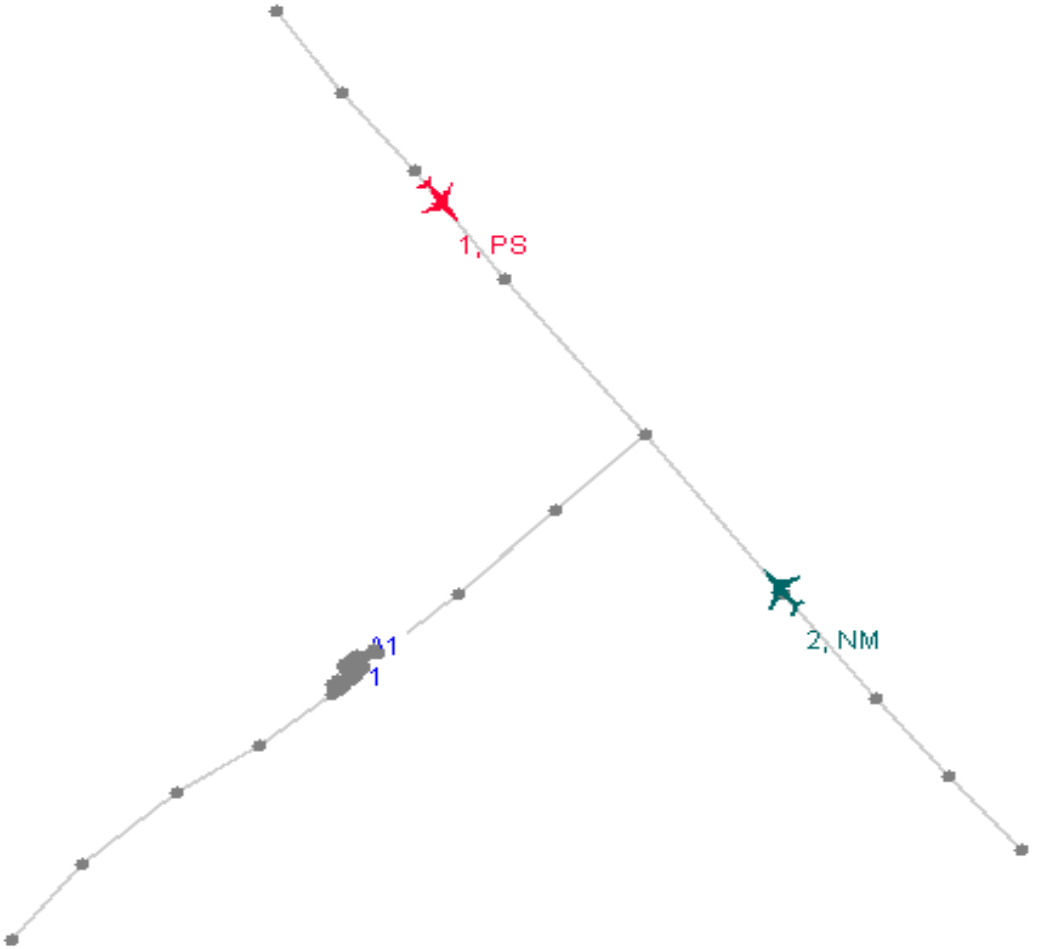
  If the supplied intrail metering separation will be violated by a later arriving aircraft the same later arriving aircraft will be delayed.

  The delay will be an appropriate amount necessary to prevent violation of the intrail metering separations.

  In the example on the next page please note the aircraft which is experiencing a PS (path stretching or vectoring) operation. The PS delay is taken prior to the common meter post node where the separation is required.

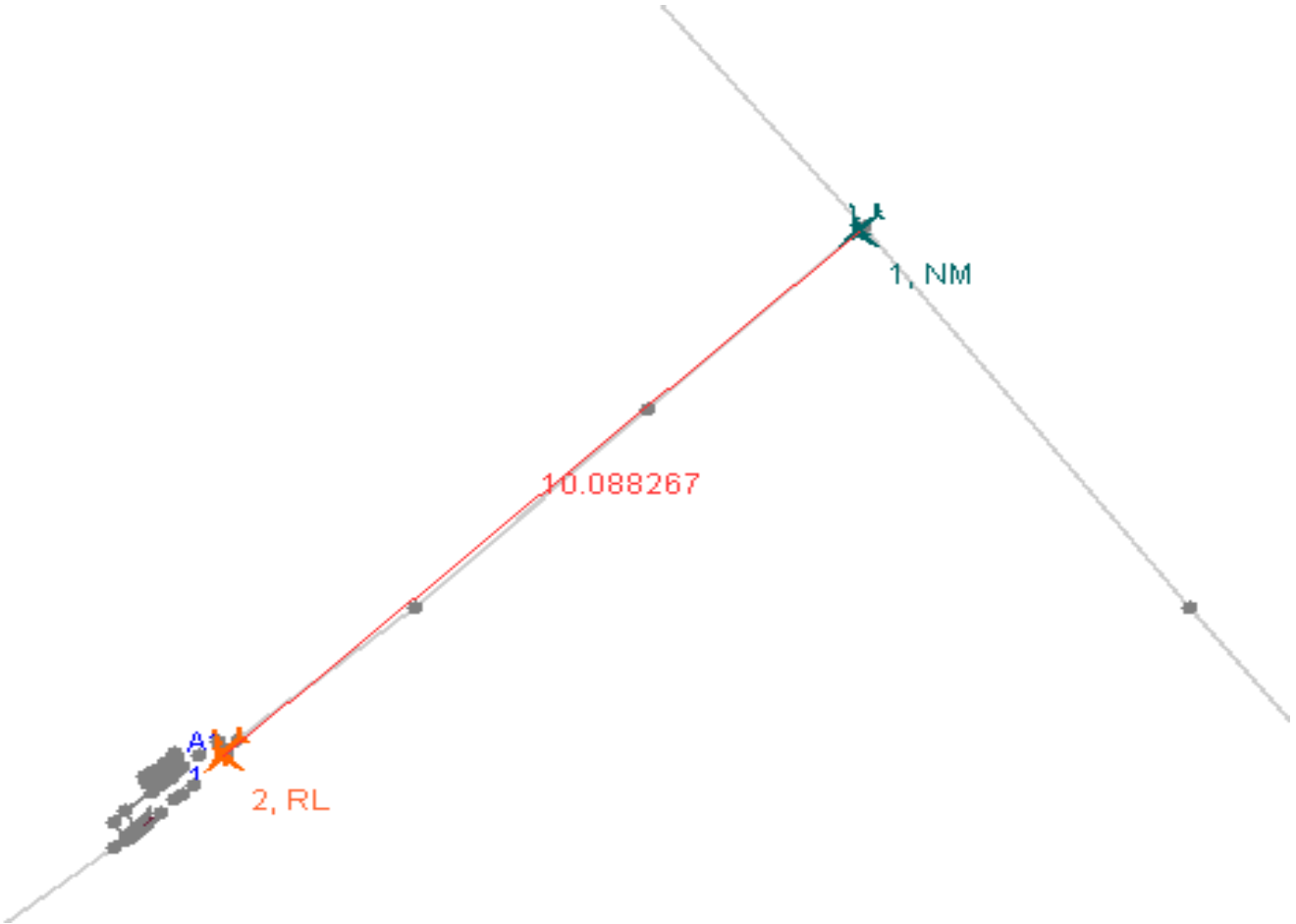
SIMMOD Metering (continued)

In this particular case the metering inputs request that the later aircraft arriving at the meter post node be separated from the first aircraft by 10 nautical miles.



SIMMOD Metering (continued)

When the second aircraft eventually reaches the meter post node it is separated from the first aircraft by the requested 10 nautical miles.



SIMMOD Metering (continued)

The following data was used to specify this metering scenario:

METER_POST_NODE

File Edit Data Options

ASN_ID	MIN_SEP	MAX_QUEUE...	MIN_QUEUE...
ASN_0009	10.00000000	3	1

METER_NODE

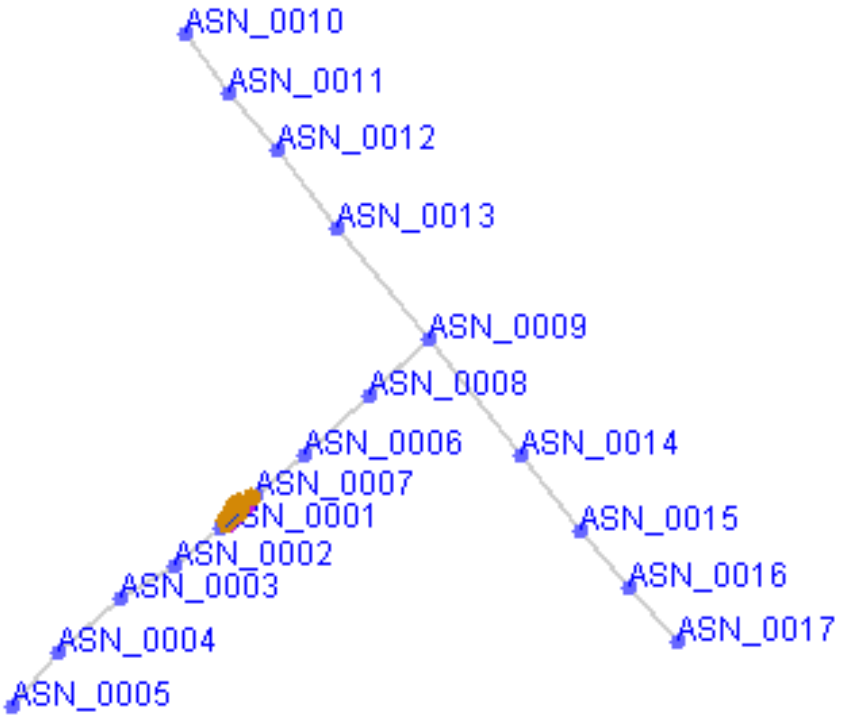
File Edit Data Options

MPN_ID	ASN_ID
ASN_0009	ASN_0011
ASN_0009	ASN_0016

ROUTE_METER_NODE

File Edit Data Options

MPN_ID	ASN_ID	RTE_ID
ASN_0009	ASN_0011	RTE_01
ASN_0009	ASN_0016	RTE_02



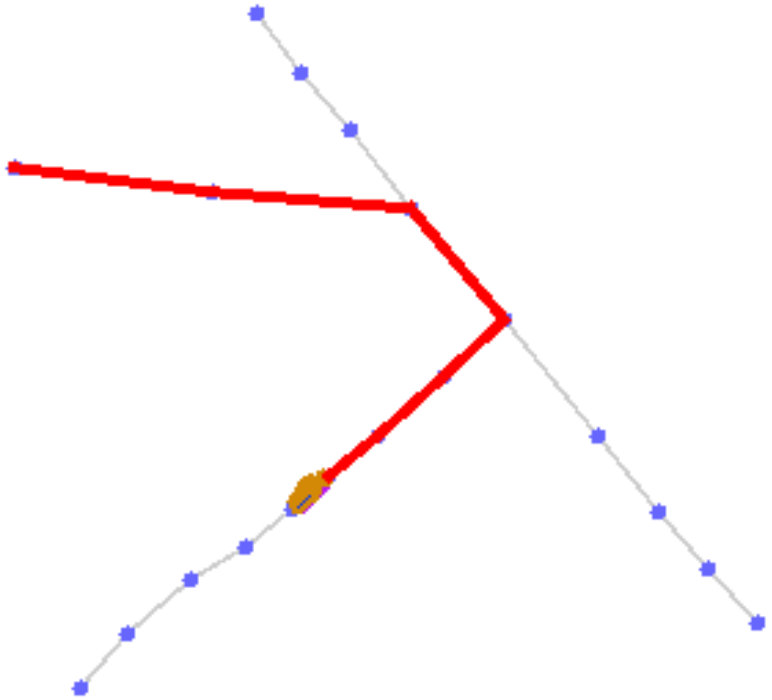
SIMMOD Metering (continued)

Warning:

Since SIMMOD uses the nominal travel time between the meter node and the meter post node as it's prediction of when each aircraft will arrive at the meter post node any extraneous delays encountered between a meter node and a meter post node will cause the metering separation to be incorrect.

The example shown to the right depicts a third route, which, if aircraft traversing it are left un-metered might cause unforeseen delays in the metered traffic and thus cause the metering results to become invalid.

The correct this situation it is recommended that the analyst meter the highlighted route shown.



SIMMOD Metering (continued)

- The second metering ability SIMMOD provides is the capability to divert aircraft from a congested final approach route to an alternative final approach route which is not congested.

  The net effect of this metering capability is to allow the model to spread arrivals out over multiple final approaches while giving initial preference to one or more final approaches.

  To determine whether an aircraft should be diverted a count of the number of flights between the meter node and the meter post node on the desired final approach is taken.

  If the count exceeds a user specified minimum (i.e. the final approach is congested) then the user supplied alternate route is checked for congestion.

  If the alternate route isn't congested then the aircraft is diverted.

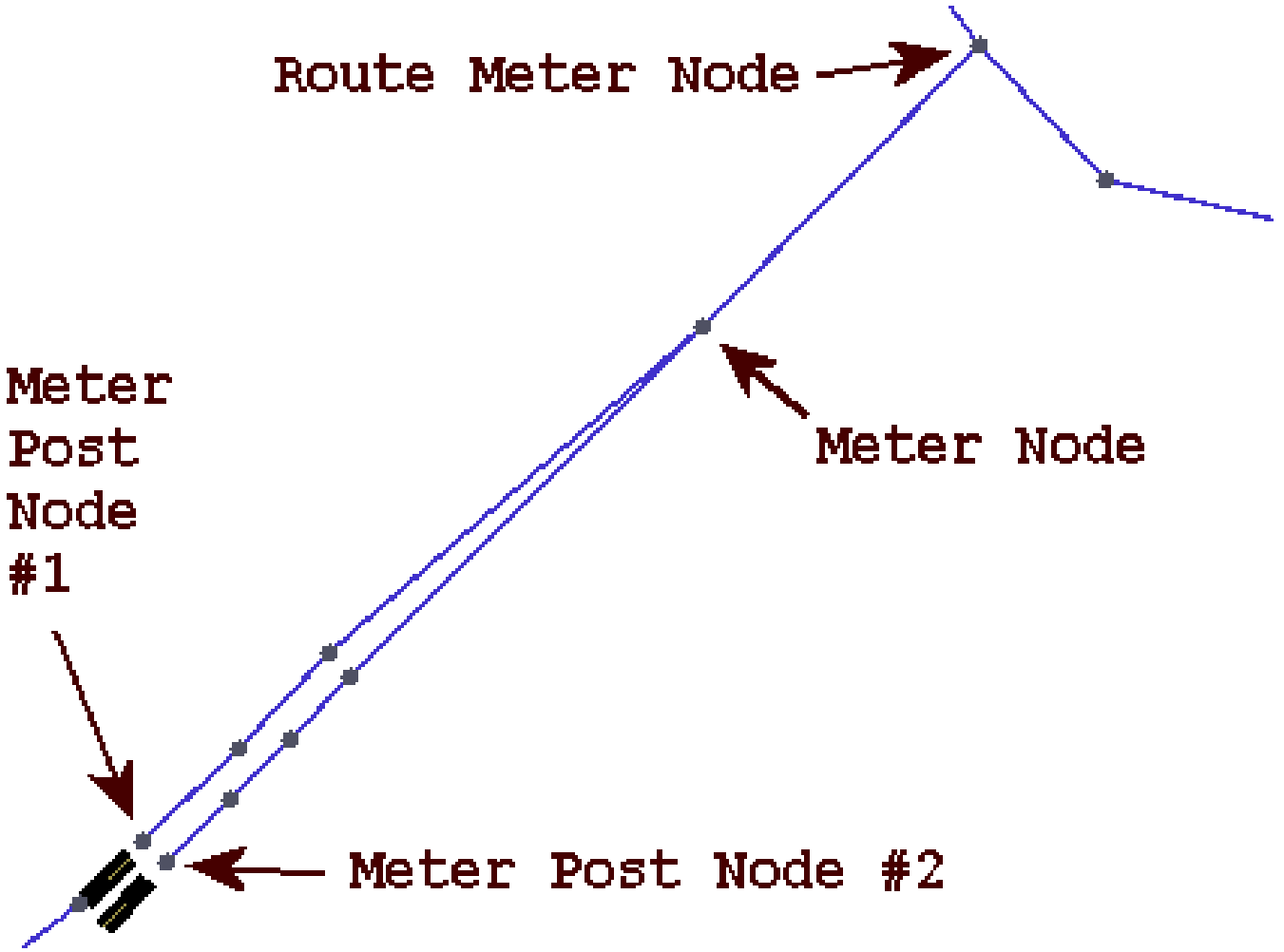
SIMMOD Metering (continued)

Final approach route switching via metering assumes:

- The desired final approach shares an airspace link with any alternate final approach routes.
- Each metered final approach route has a meter post node which is co-located at an arrival interface node.
- Each metered final approach route has a threshold level of traffic which will cause flights to seek a less congested alternate final approach route.
- Each alternate final approach route has a maximum level of traffic which will cause flights to avoid it.

SIMMOD Metering (continued)

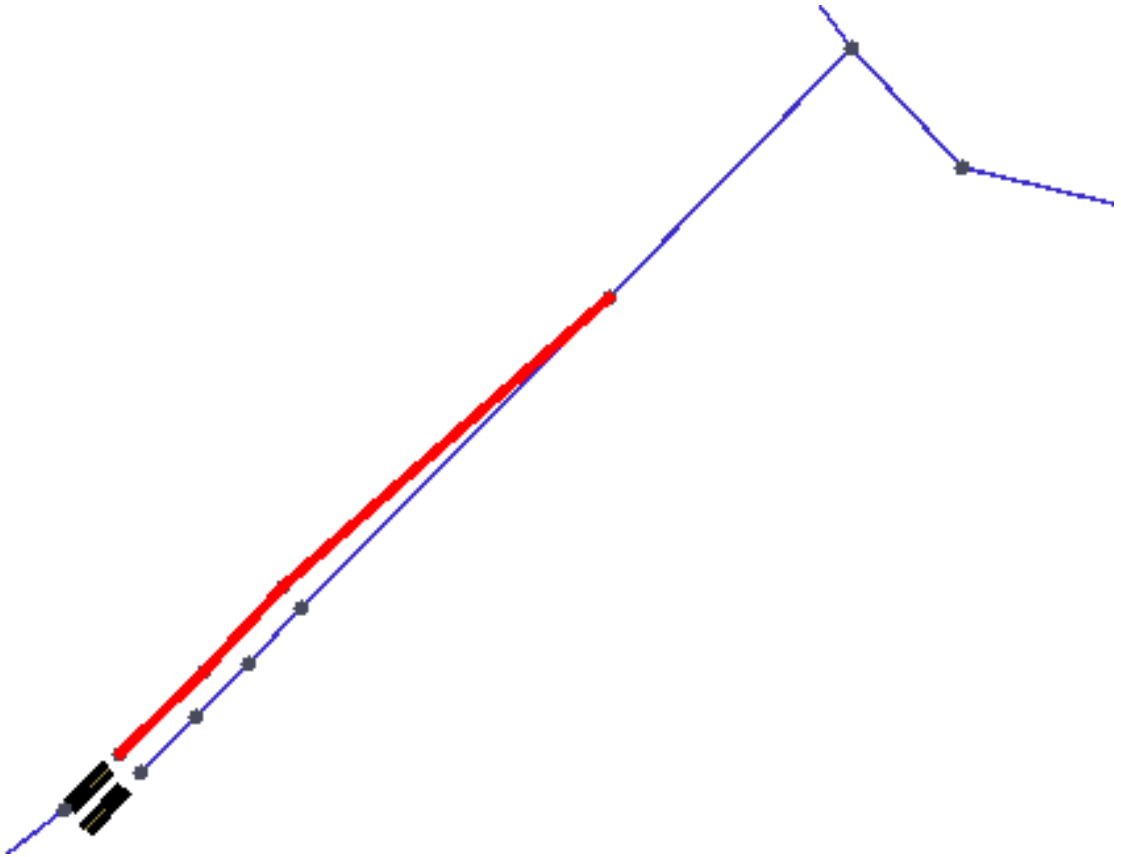
In the following case two final approach arrival routes are shown:



SIMMOD Metering (continued)

The preferred final approach is shown highlighted in red below.

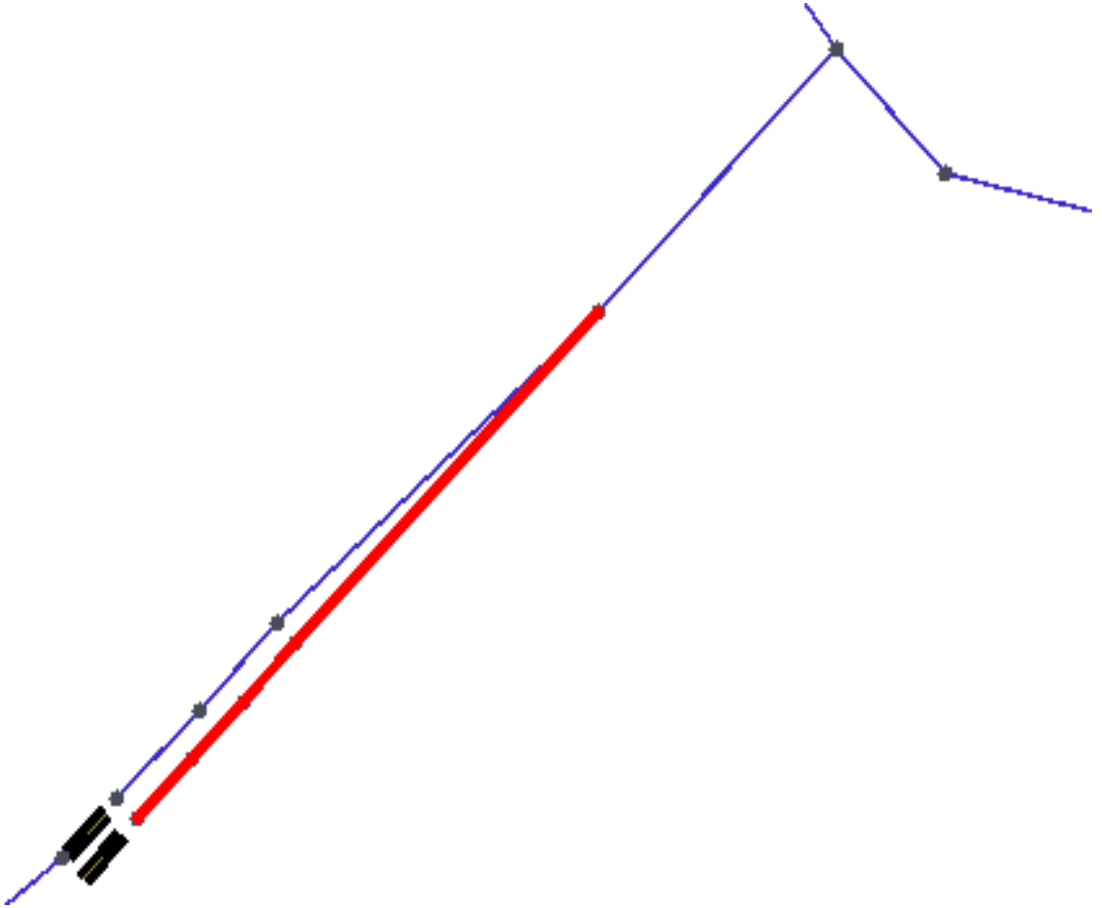
As noted earlier, if the number of aircraft traversing the preferred arrival route between the meter node and the meter post node exceeds a minimum threshold value provided by the user then aircraft are diverted at the meter node to the provided alternate route.



SIMMOD Metering (continued)

The alternate route is shown highlighted in red.

Aircraft may be diverted to the alternate route until the count of the number of aircraft on this route between the meter node and the meter post node exceeds a user specified maximum value.



SIMMOD Metering (continued)

The required inputs to SIMMOD are as follows:

METER_POST_NODE

File Edit Data Options

ASN_ID	MIN_SEP	MAX_QUEUE...	MIN_QUEUE...
ASN_0007	0.00000000	1	1
ASN_0022	0.00000000	2	1

METER_NODE

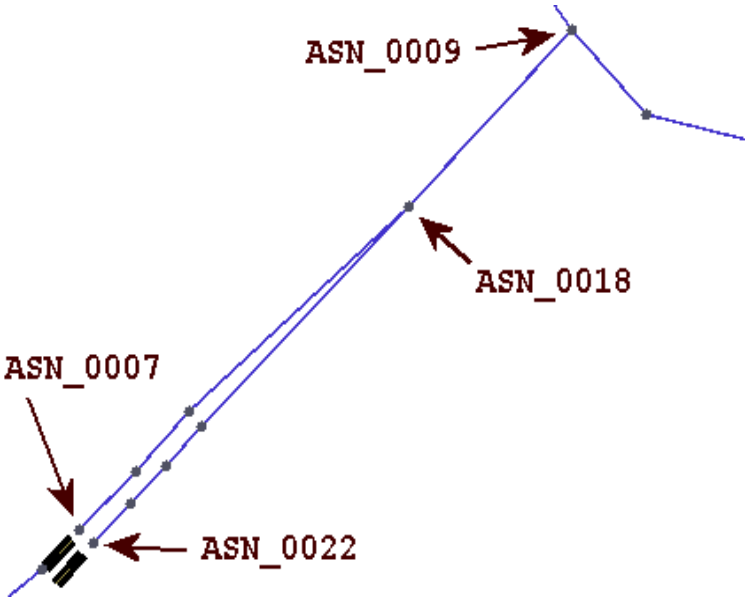
File Edit Data Options

MPN_ID	ASN_ID
ASN_0007	ASN_0018
ASN_0022	ASN_0018

ROUTE_METER_NODE

File Edit Data Options

MPN_ID	ASN_ID	RTE_ID
ASN_0007	ASN_0009	RTE_01
ASN_0007	ASN_0009	RTE_02
ASN_0022	ASN_0009	ARR_23L

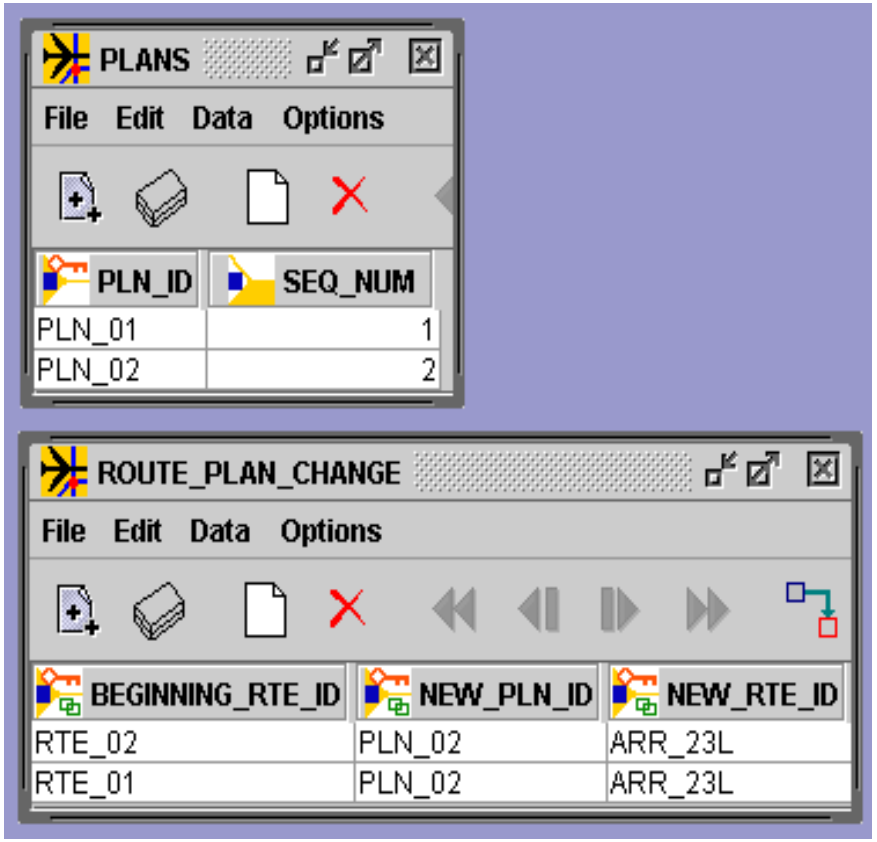


Please note that the system could have been defined such that the meter node (ASN_0018) and the route meter node (ASN_0009) were located on the same physical node.

But, testing has shown that metering tends to work better when configured as shown.

SIMMOD Metering (continued)

Additional required inputs to SIMMOD are as follows:



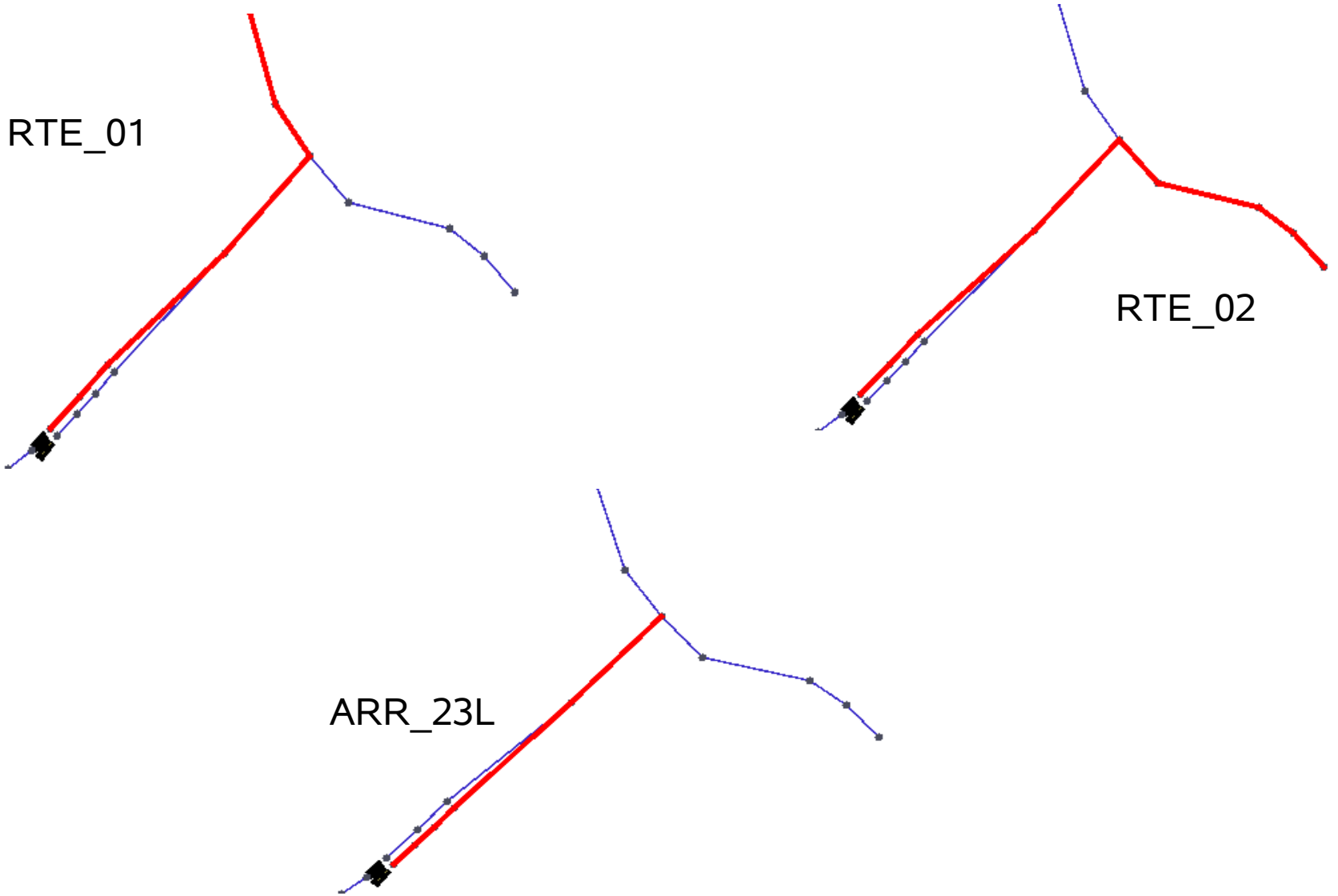
A plan to accommodate route switching is required. Note plans “PLN_01” and “PLN_02”.

PLN_02 is the plan which is utilized for route switching.

The ROUTE_PLAN_CHANGE data indicates that if the original route is RTE_01 or RTE_02 then upon the requested metering route switch utilize the ARR_23L route instead.

SIMMOD Metering (continued)

The described routes are as follows:



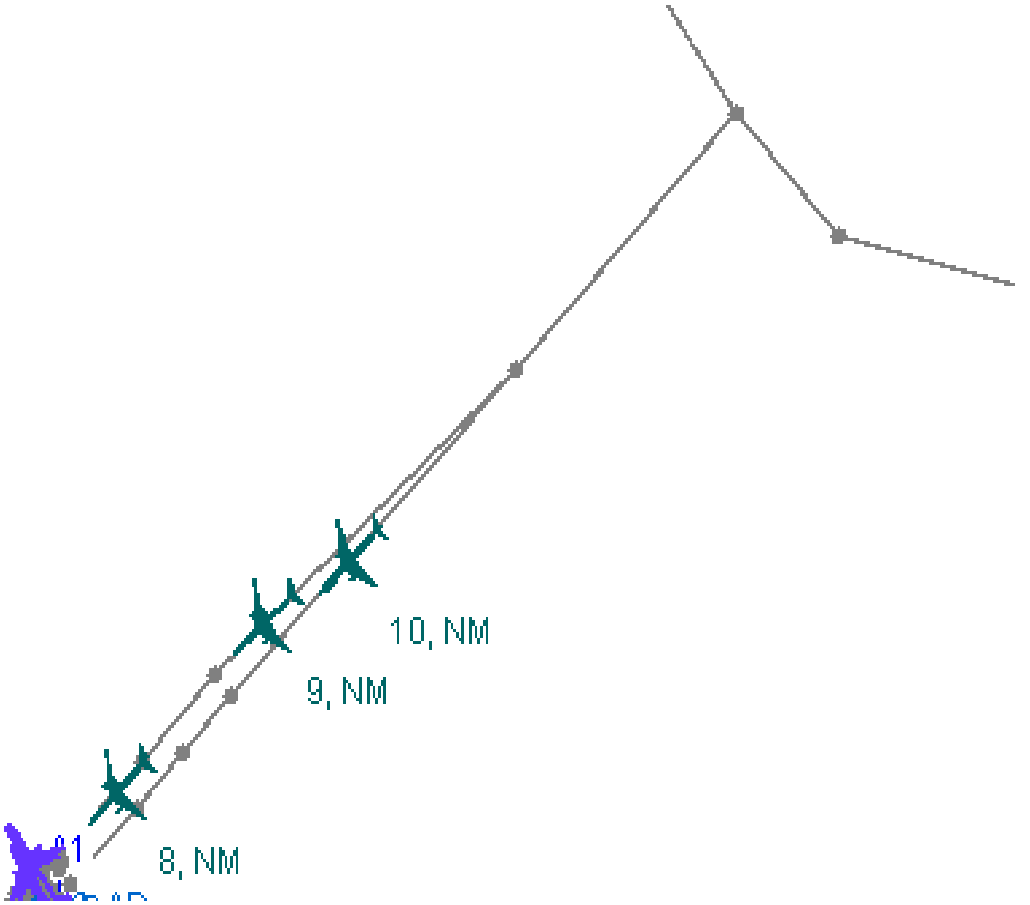
SIMMOD Metering (continued)

After running SIMMOD an example of the results:

As can be seen two aircraft (8 and 9) are making an approach upon the desired final.

But, due to there being two aircraft on the final approach aircraft 10 has been re-directed to land upon the alternate final approach route.

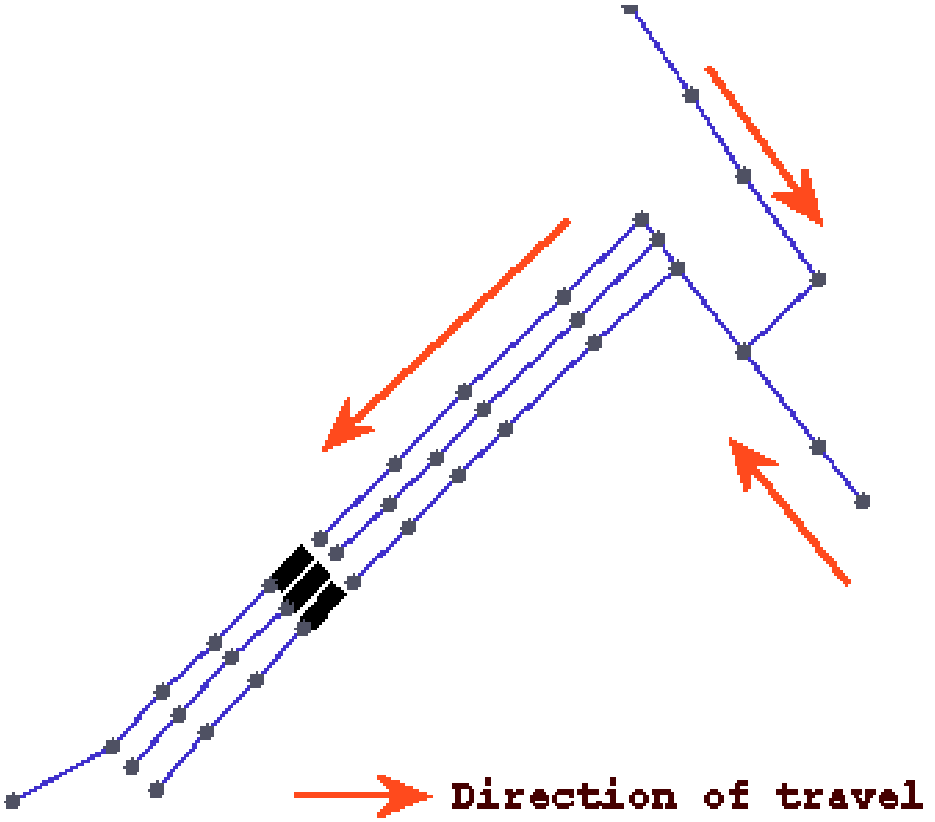
If more than two aircraft had previously been assigned to the alternate final approach then no further re-assignments would take place until the alternate is no longer congested.



Aeroscript Final Approach Metering

Aeroscript is a new powerful aviation modeling capability. With Aeroscript an analyst can create a metering scenario that far exceeds that available with SIMMOD.

Aeroscript provides the means to check multiple alternative final approach routes. The following scenario can be easily modeled using Aeroscript:

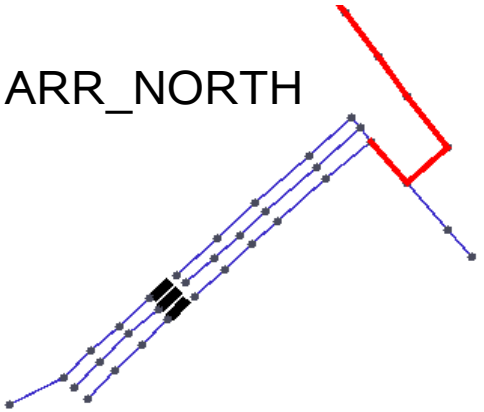


Aeroscript Final Approach Metering (continued)

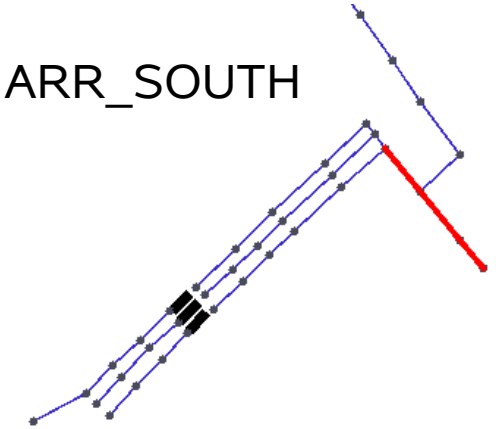
To begin using Aeroscript you need to perform the following basic steps:

- Create a standard SIMMOD application with three runways. Then create the following routes:

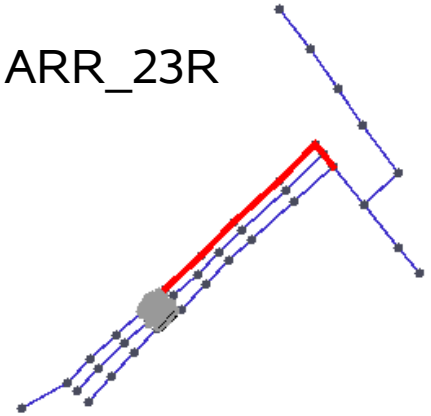
ARR_NORTH



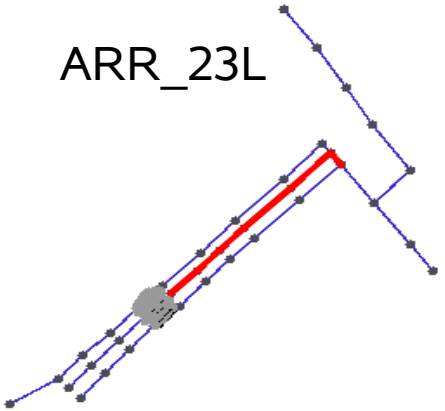
ARR_SOUTH



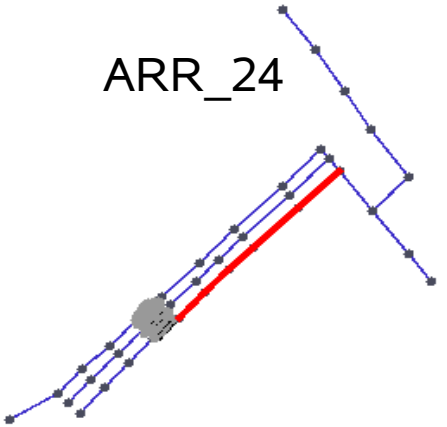
ARR_23R



ARR_23L

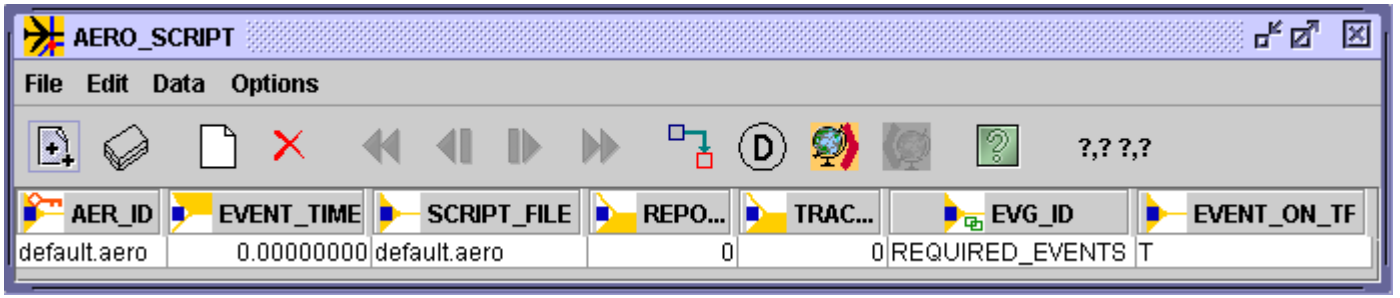


ARR_24



Aeroscript Final Approach Metering (continued)

- Create an entry in the JSIMMOD... AERO_SCRIPT table in the following manner:



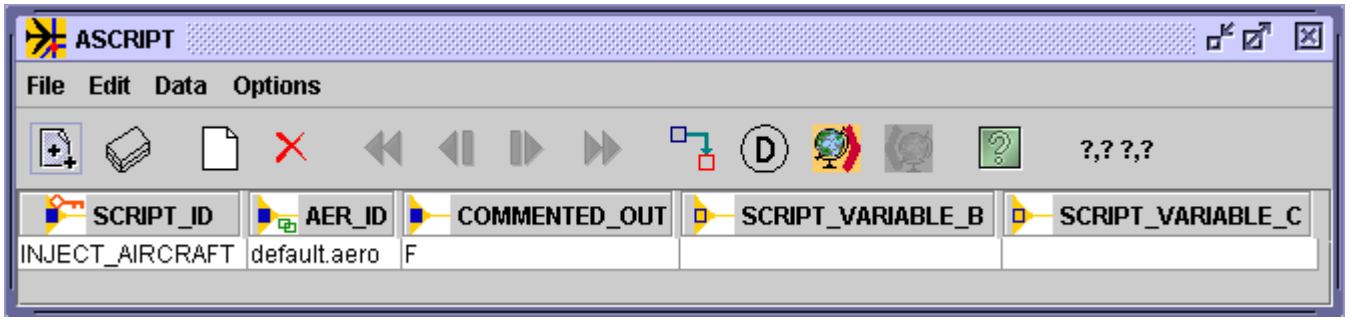
The above entry states that the entire default.aero SCRIPT_FILE will be read at time 0.0 during the simulation run.

Default.aero should contain one or more Aeroscript routines.

The contents of default.aero will be read into memory and retained for further usage by MISSIONs which desire to execute any of the scripts contained there-in.

Aeroscript Final Approach Metering (continued)

- Create an entry in the JSIMMOD... ASCRIPT table in the following manner:



The above entry indicates that SCRIPT_ID “INJECT_AIRCRAFT” will be located in the file (default.aero) referenced by the default.aero AER_ID.

As noted previously, default.aero should contain one or more Aeroscript routines. While the name of an Aeroscript routine is user-defined, in this example the name shall be: INJECT_AIRCRAFT.

INJECT_AIRCRAFT will be a routine which performs some basic initialization routines that all missions will utilize.

The actual contents of INJECT_AIRCRAFT will be described further at a later time in this document.

Aeroscript Final Approach Metering (continued)

- Create any number of arrival SIMMOD FLIGHTs in the standard manner with the following exceptions:

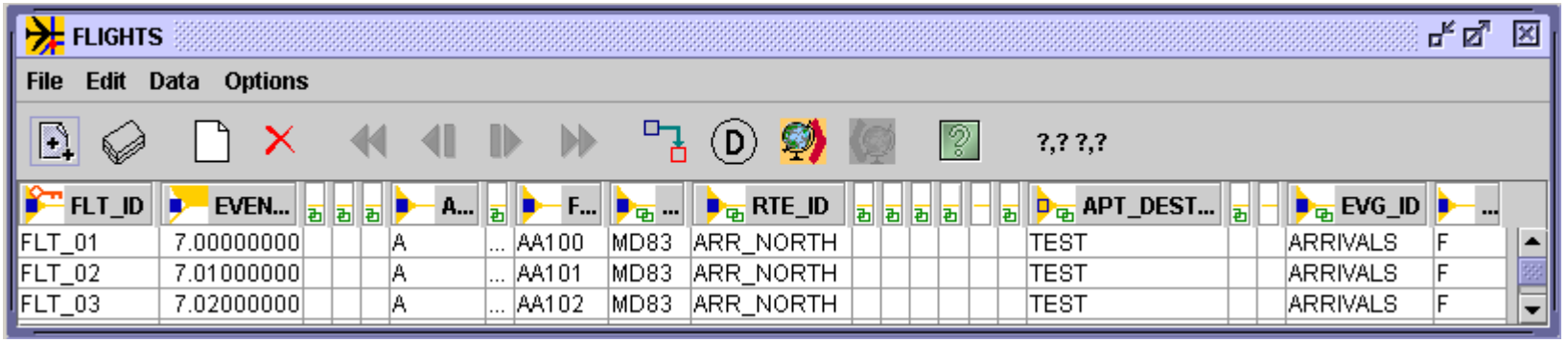
  For the purposes of this tutorial please create these flights with an EVENT_TIME of 7.00000.

  Additionally, all of the flights should either utilize route ARR_NORTH or ARR_SOUTH in their RTE_ID field.

  Fifteen flights utilizing ARR_NORTH and fifteen flights utilizing ARR_SOUTH should be sufficient.

  And lastly, the EVENT_ON_TF field should be set to F.

  By setting the EVENT_ON_TF field to F you have created a placeholder which will allow Aeroscript to "lookup" the data needed from the FLIGHTS table without having the flights actually flown in the normal SIMMOD manner.



FLT_ID	EVEN...	A...	F...	...	RTE_ID	APT_DEST...	EVG_ID	...
FLT_01	7.00000000	A	...	AA100	MD83	ARR_NORTH	TEST	ARRIVALS	F
FLT_02	7.01000000	A	...	AA101	MD83	ARR_NORTH	TEST	ARRIVALS	F
FLT_03	7.02000000	A	...	AA102	MD83	ARR_NORTH	TEST	ARRIVALS	F

Aeroscript Final Approach Metering (continued)

- Create the exact same number of Aeroscript Events...JSIMMOD MISSIONs (in this case 30) as the number of FLIGHTs you created.

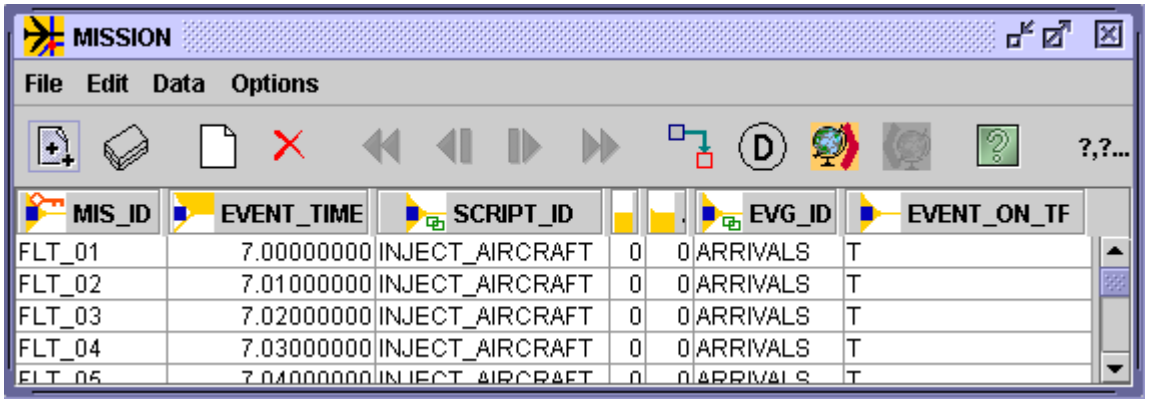
  In Aeroscript, a mission represents a flight. In advanced cases, a mission can represent an entity other than a flight, such as an air traffic controller.

  Because we desire to link our missions data to our flights data each MISSION's MIS_ID field should be equivalent to one FLT_ID in the FLIGHTS table.

  Pay close attention to the EVENT_TIME for each mission. The EVENT_TIME is when each mission will be executed by the Aeroscript model.

  You should supply the previously defined “INJECT_AIRCRAFT” as the SCRIPT_ID this mission will initially execute.

  You may place 0 in the REPORTING_LEVEL and TRACE_LEVEL fields.



MIS_ID	EVENT_TIME	SCRIPT_ID			EVG_ID	EVENT_ON_TF
FLT_01	7.00000000	INJECT_AIRCRAFT	0	0	ARRIVALS	T
FLT_02	7.01000000	INJECT_AIRCRAFT	0	0	ARRIVALS	T
FLT_03	7.02000000	INJECT_AIRCRAFT	0	0	ARRIVALS	T
FLT_04	7.03000000	INJECT_AIRCRAFT	0	0	ARRIVALS	T
FLT_05	7.04000000	INJECT_AIRCRAFT	0	0	ARRIVALS	T

Aeroscript Final Approach Metering (continued)

- The next step in the creation of an Aeroscript model is to create the INJECT_AIRCRAFT routine in the default.aero script.

  Open the Aeroscript Editor from the tools menu of the Visual SIMMOD Editor and click on the INJECT_AIRCRAFT routine in the left column.

  Click on the Misc tab.

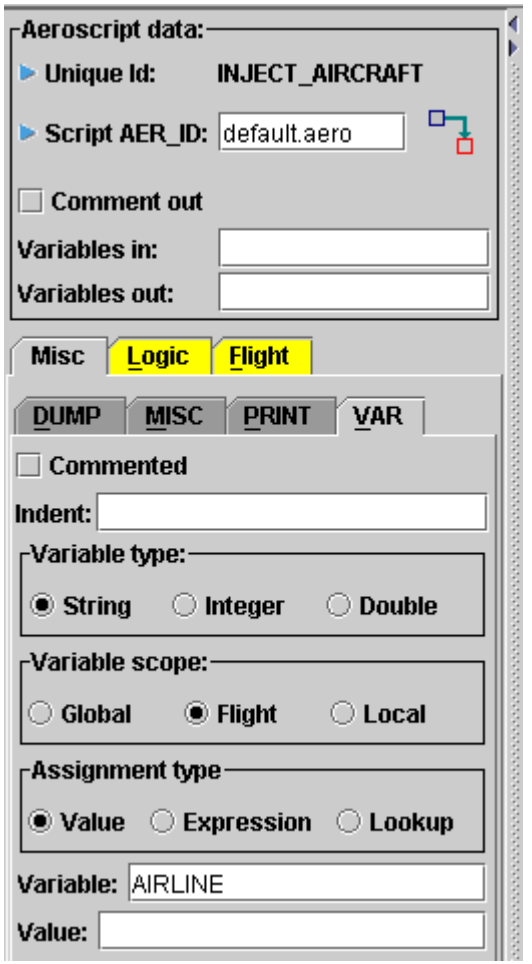
  Click on the Var sub-tab.

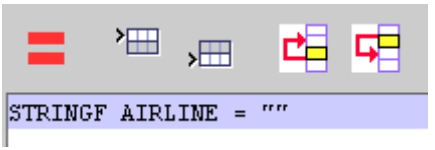
  Change the variable scope to Flight. This means that the variable to be defined is local to the particular mission and cannot be used or seen by other missions.

  Type the word AIRLINE in the variable field.

  Press the  button to insert the line into the routine as shown to the right.

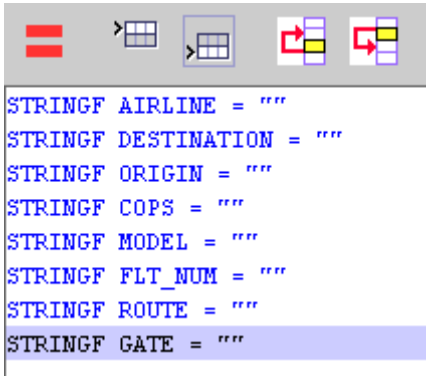
  You have now specified a variable called AIRLINE which will be associated with any missions executing this routine.





Aeroscript Final Approach Metering (continued)

- Create the variables as shown to the right exactly as you created the AIRLINE variable.



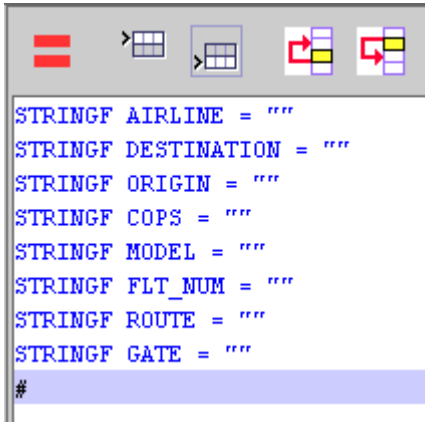
- To create a more readable script insert a comment line next.

  Click on the Misc tab.

  Click on the Misc sub-tab.

  Choose the Commented checkbox

  Add the line in the same manner as those prior.



Aeroscript Final Approach Metering (continued)

- Next assign values via the Aeroscript Lookup function to the variables just created.

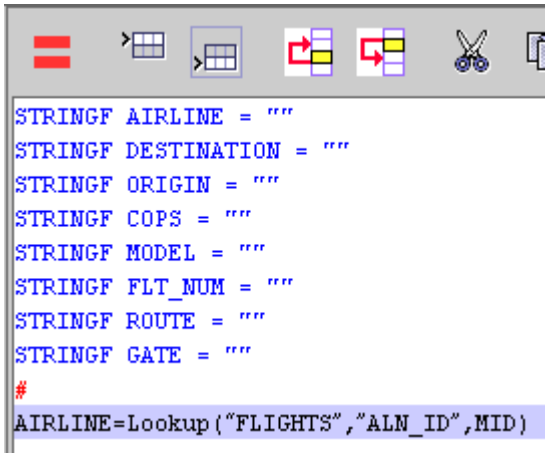
  Click on the Misc tab.

  Click on the Misc sub-tab.

  Make sure the Commented checkbox is not chosen.

  Type the following into the Value field:

  AIRLINE=Lookup(“FLIGHTS”,”ALN_ID”,MID)



  Use the  button to add the line to the routine.

  In the example shown, the variable AIRLINE is being assigned the ALN_ID value “looked up” in the FLIGHTS table for the particular MID.

  MID refers to the current mission's identifier or rather it's MIS_ID from the MISSION table. The MID is matched to the key field of the FLIGHTS table, which is the FLT_ID field.

  At this time, the airline has not yet been assigned to the mission.

Aeroscript Final Approach Metering (continued)

- Insert the following lines in the same manner as was just used to assign the airline.

DESTINATION=Lookup ("FLIGHTS","APT_DEST_ID",MID)

ORIGIN=Lookup ("FLIGHTS","APT_ORIG_ID",MID)

COPS=Lookup ("FLIGHTS","A_D_FLAG",MID)

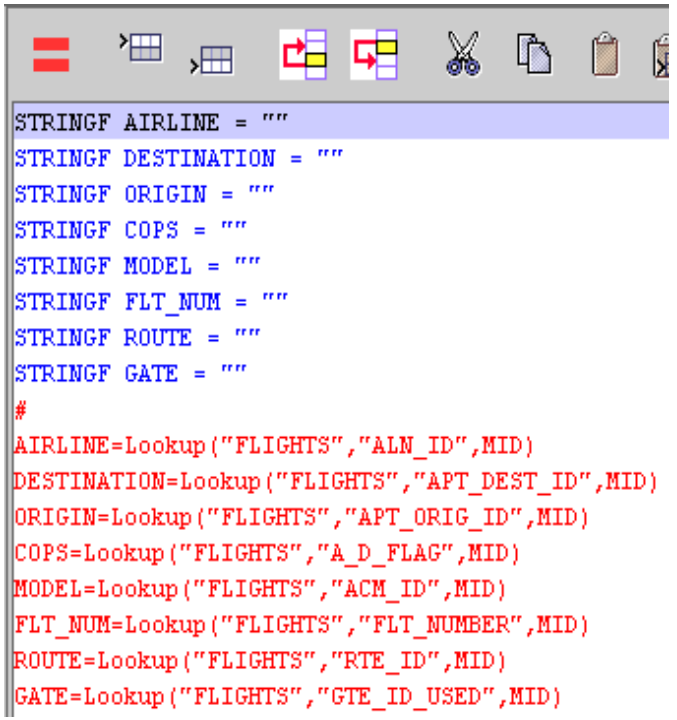
MODEL=Lookup ("FLIGHTS","ACM_ID",MID)

FLT_NUM=Lookup ("FLIGHTS","FLT_NUMBER",MID)

ROUTE=Lookup ("FLIGHTS","RTE_ID",MID)

GATE=Lookup ("FLIGHTS","GTE_ID_USED",MID)

The newly inserted lines are shown in red because any line which is a miscellaneous line is coded to be red.



Insert a comment line next by clicking on the Misc tab and then the Misc sub-tab, then click on the Commented checkbox, then erase any material in the Edit field.

Then press the  button.

Aeroscript Final Approach Metering (continued)

- Next assign the variable values to the mission itself:

  Assign the flight number via the following steps:

  Click on the Flight tab.

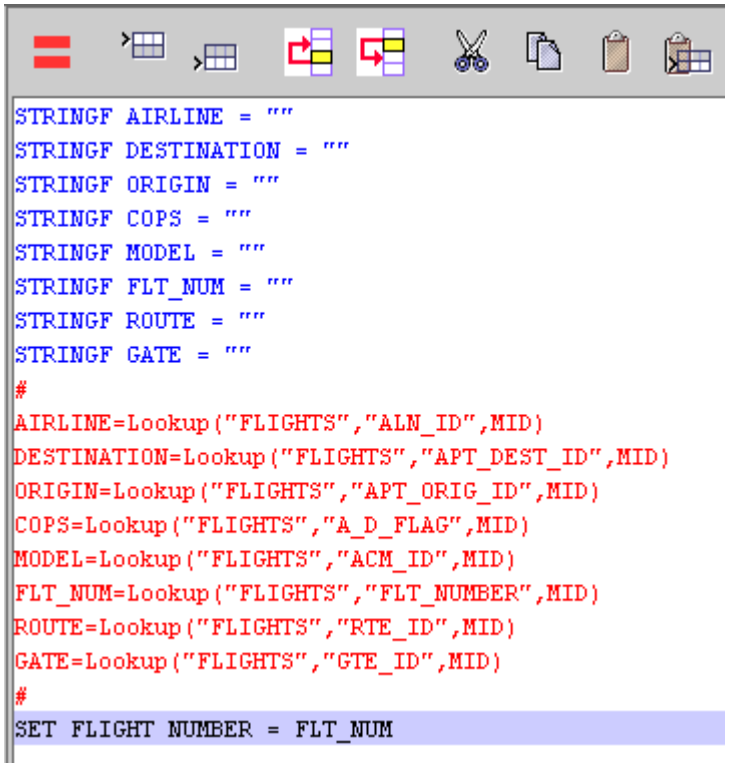
  Click on the FLN sub-tab.

  Click on the Expression assignment type.

  Type FLT_NUM into the Expression field provided.

  Click on the  button to add the line to the routine.

  This line actually assigns the flight number which was stored in the FLT_NUM variable to the MISSION.



```
STRINGF AIRLINE = ""
STRINGF DESTINATION = ""
STRINGF ORIGIN = ""
STRINGF COPS = ""
STRINGF MODEL = ""
STRINGF FLT_NUM = ""
STRINGF ROUTE = ""
STRINGF GATE = ""
#
AIRLINE=Lookup("FLIGHTS","ALN_ID",MID)
DESTINATION=Lookup("FLIGHTS","APT_DEST_ID",MID)
ORIGIN=Lookup("FLIGHTS","APT_ORIG_ID",MID)
COPS=Lookup("FLIGHTS","A_D_FLAG",MID)
MODEL=Lookup("FLIGHTS","ACM_ID",MID)
FLT_NUM=Lookup("FLIGHTS","FLT_NUMBER",MID)
ROUTE=Lookup("FLIGHTS","RTE_ID",MID)
GATE=Lookup("FLIGHTS","GTE_ID",MID)
#
SET FLIGHT NUMBER = FLT_NUM
```

Aeroscript Final Approach Metering (continued)

- Assign the COPS (category of operations, A-arrival, D-departure, etc.) via the following steps:

  Click on the Flights tab.

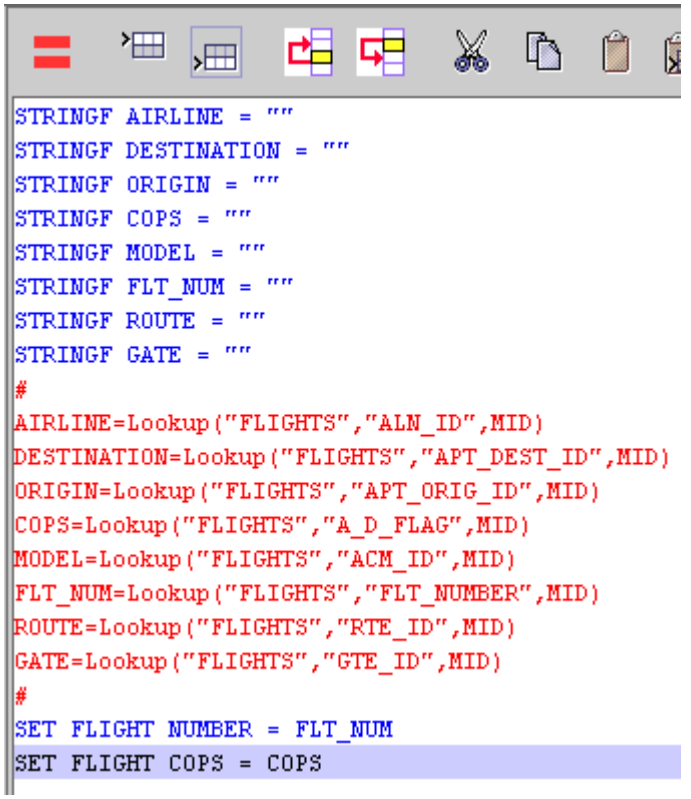
  Click on the COPS sub-tab.

  Click on the Expression assignment type.

  Type COPS into the Expression field provided.

  Click on the  button to add the line to the routine.

  This line actually assigns the category of operations which was stored in the COPS variable to the MISSION.



```
STRINGF AIRLINE = ""
STRINGF DESTINATION = ""
STRINGF ORIGIN = ""
STRINGF COPS = ""
STRINGF MODEL = ""
STRINGF FLT_NUM = ""
STRINGF ROUTE = ""
STRINGF GATE = ""
#
AIRLINE=Lookup("FLIGHTS","ALN_ID",MID)
DESTINATION=Lookup("FLIGHTS","APT_DEST_ID",MID)
ORIGIN=Lookup("FLIGHTS","APT_ORIG_ID",MID)
COPS=Lookup("FLIGHTS","A_D_FLAG",MID)
MODEL=Lookup("FLIGHTS","ACM_ID",MID)
FLT_NUM=Lookup("FLIGHTS","FLT_NUMBER",MID)
ROUTE=Lookup("FLIGHTS","RTE_ID",MID)
GATE=Lookup("FLIGHTS","GTE_ID",MID)
#
SET FLIGHT NUMBER = FLT_NUM
SET FLIGHT COPS = COPS
```

Aeroscript Final Approach Metering (continued)

- Assign the airline via the following steps:

  Click on the Flights tab.

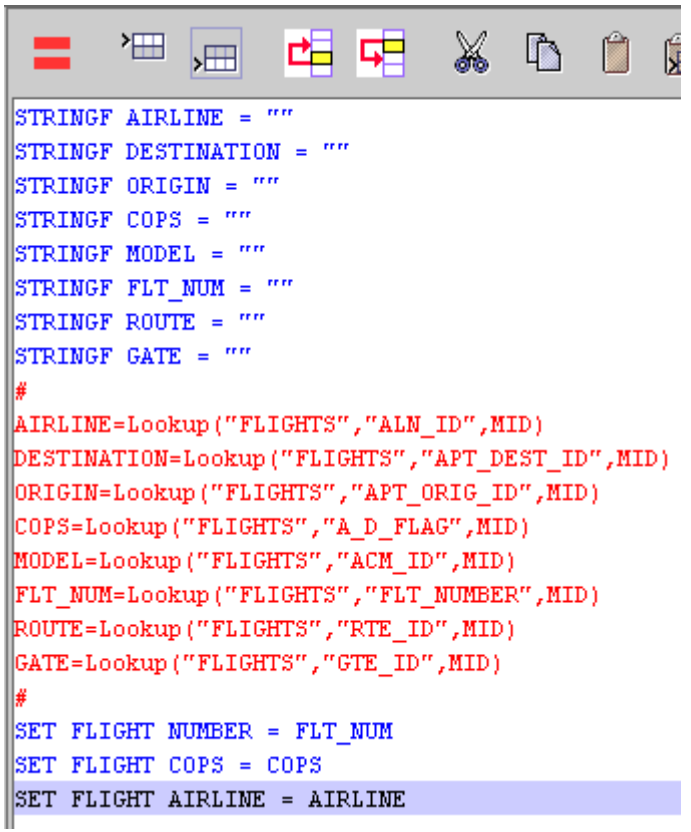
  Click on the ALN sub-tab.

  Click on the Expression assignment type.

  Type AIRLINE into the Expression field provided.

  Click on the  button to add the line to the routine.

  This line assigns the airline identifier which was stored in the AIRLINE variable to the MISSION.



```
STRINGF AIRLINE = ""
STRINGF DESTINATION = ""
STRINGF ORIGIN = ""
STRINGF COPS = ""
STRINGF MODEL = ""
STRINGF FLT_NUM = ""
STRINGF ROUTE = ""
STRINGF GATE = ""
#
AIRLINE=Lookup("FLIGHTS","ALN_ID",MID)
DESTINATION=Lookup("FLIGHTS","APT_DEST_ID",MID)
ORIGIN=Lookup("FLIGHTS","APT_ORIG_ID",MID)
COPS=Lookup("FLIGHTS","A_D_FLAG",MID)
MODEL=Lookup("FLIGHTS","ACM_ID",MID)
FLT_NUM=Lookup("FLIGHTS","FLT_NUMBER",MID)
ROUTE=Lookup("FLIGHTS","RTE_ID",MID)
GATE=Lookup("FLIGHTS","GTE_ID",MID)
#
SET FLIGHT NUMBER = FLT_NUM
SET FLIGHT COPS = COPS
SET FLIGHT AIRLINE = AIRLINE
```

Aeroscript Final Approach Metering (continued)

- Assign the aircraft model via the following steps:

  Click on the Flights tab.

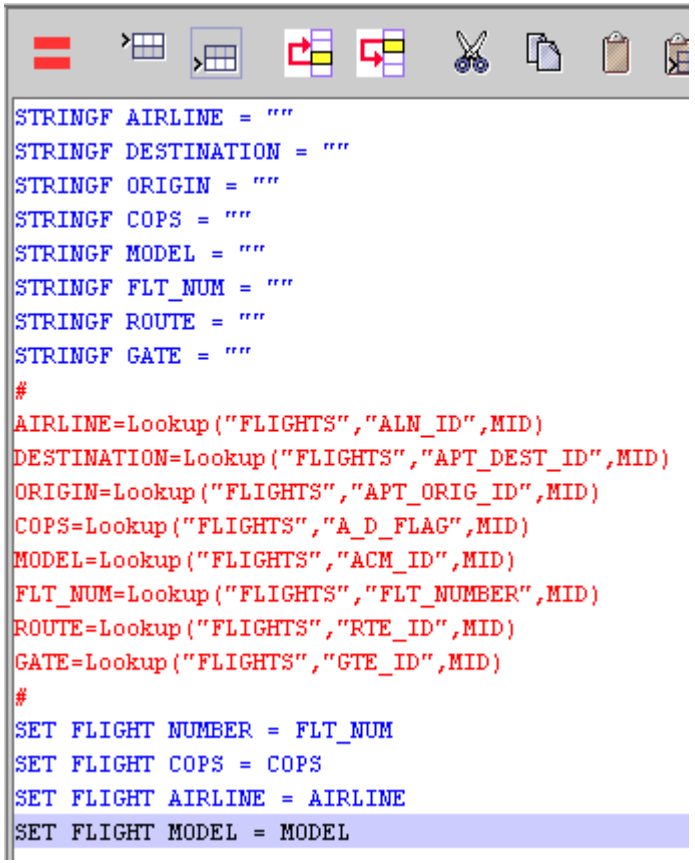
  Click on the ACM sub-tab.

  Click on the Expression assignment type.

  Type MODEL into the Expression field provided.

  Click on the  button to add the line to the routine.

  This line assigns the aircraft model identifier which was stored in the MODEL variable to the MISSION.



```
STRINGF AIRLINE = ""
STRINGF DESTINATION = ""
STRINGF ORIGIN = ""
STRINGF COPS = ""
STRINGF MODEL = ""
STRINGF FLT_NUM = ""
STRINGF ROUTE = ""
STRINGF GATE = ""
#
AIRLINE=Lookup("FLIGHTS","ALN_ID",MID)
DESTINATION=Lookup("FLIGHTS","APT_DEST_ID",MID)
ORIGIN=Lookup("FLIGHTS","APT_ORIG_ID",MID)
COPS=Lookup("FLIGHTS","A_D_FLAG",MID)
MODEL=Lookup("FLIGHTS","ACM_ID",MID)
FLT_NUM=Lookup("FLIGHTS","FLT_NUMBER",MID)
ROUTE=Lookup("FLIGHTS","RTE_ID",MID)
GATE=Lookup("FLIGHTS","GTE_ID",MID)
#
SET FLIGHT NUMBER = FLT_NUM
SET FLIGHT COPS = COPS
SET FLIGHT AIRLINE = AIRLINE
SET FLIGHT MODEL = MODEL
```

Aeroscript Final Approach Metering (continued)

- Assign the destination via the following steps:

  Click on the Flights tab.

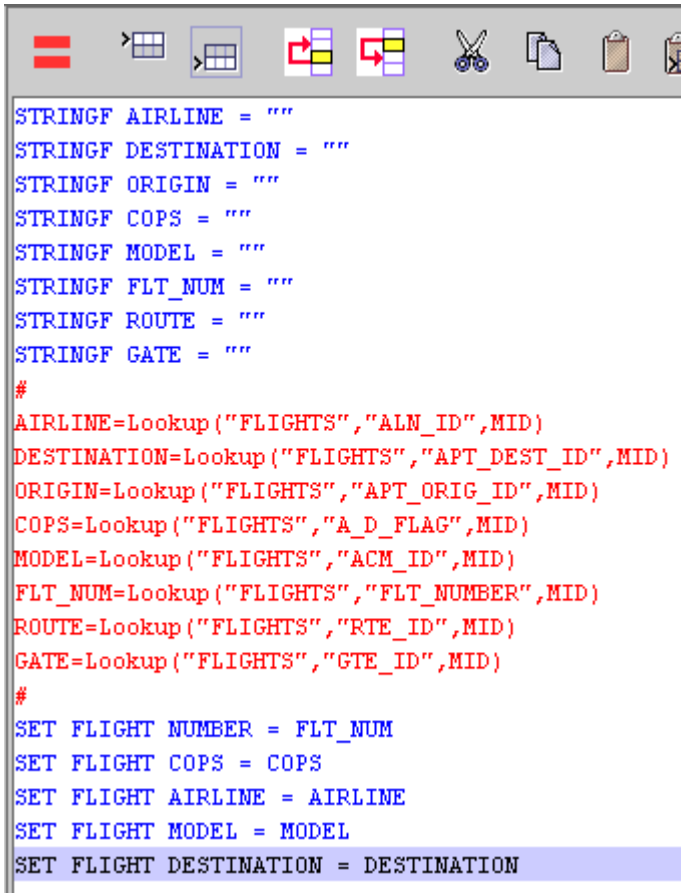
  Click on the DEST sub-tab.

  Click on the Expression assignment type.

  Type DESTINATION into the Expression field provided.

  Click on the  button to add the line to the routine.

  This line assigns the destination airport identifier which was stored in the DESTINATION variable to the MISSION.



```
STRINGF AIRLINE = ""
STRINGF DESTINATION = ""
STRINGF ORIGIN = ""
STRINGF COPS = ""
STRINGF MODEL = ""
STRINGF FLT_NUM = ""
STRINGF ROUTE = ""
STRINGF GATE = ""
#
AIRLINE=Lookup("FLIGHTS","ALN_ID",MID)
DESTINATION=Lookup("FLIGHTS","APT_DEST_ID",MID)
ORIGIN=Lookup("FLIGHTS","APT_ORIG_ID",MID)
COPS=Lookup("FLIGHTS","A_D_FLAG",MID)
MODEL=Lookup("FLIGHTS","ACM_ID",MID)
FLT_NUM=Lookup("FLIGHTS","FLT_NUMBER",MID)
ROUTE=Lookup("FLIGHTS","RTE_ID",MID)
GATE=Lookup("FLIGHTS","GTE_ID",MID)
#
SET FLIGHT NUMBER = FLT_NUM
SET FLIGHT COPS = COPS
SET FLIGHT AIRLINE = AIRLINE
SET FLIGHT MODEL = MODEL
SET FLIGHT DESTINATION = DESTINATION
```

Aeroscript Final Approach Metering (continued)

- Assign the origin via the following steps:

  Click on the Flights tab.

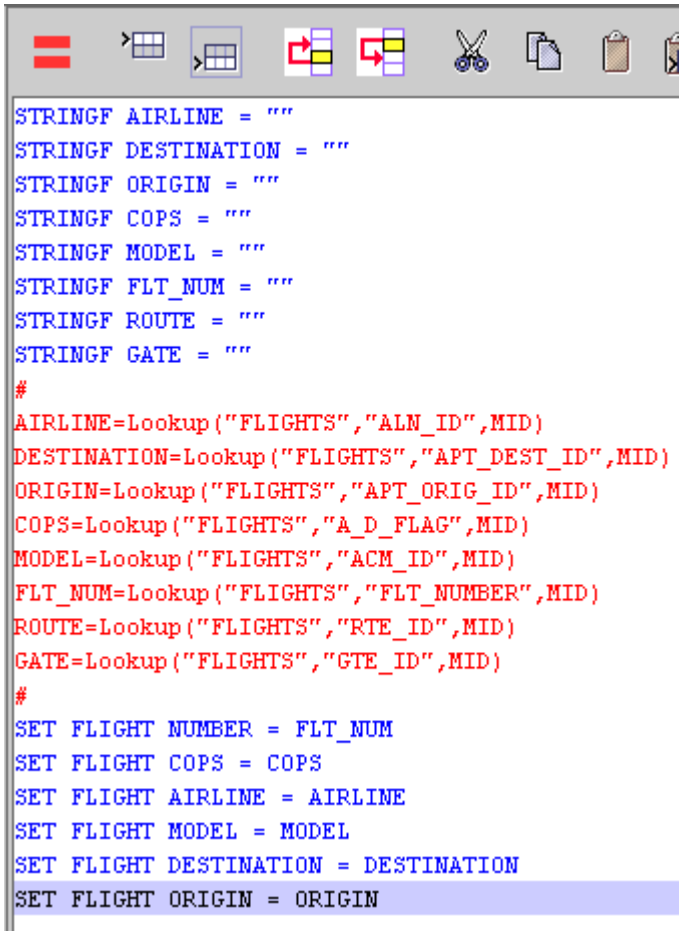
  Click on the ORIG sub-tab.

  Click on the Expression assignment type.

  Type ORIGIN into the Expression field provided.

  Click on the  button to add the line to the routine.

  This line assigns the origin airport identifier which was stored in the ORIGIN variable to the MISSION.



```
STRINGF AIRLINE = ""
STRINGF DESTINATION = ""
STRINGF ORIGIN = ""
STRINGF COPS = ""
STRINGF MODEL = ""
STRINGF FLT_NUM = ""
STRINGF ROUTE = ""
STRINGF GATE = ""
#
AIRLINE=Lookup("FLIGHTS","ALN_ID",MID)
DESTINATION=Lookup("FLIGHTS","APT_DEST_ID",MID)
ORIGIN=Lookup("FLIGHTS","APT_ORIG_ID",MID)
COPS=Lookup("FLIGHTS","A_D_FLAG",MID)
MODEL=Lookup("FLIGHTS","ACM_ID",MID)
FLT_NUM=Lookup("FLIGHTS","FLT_NUMBER",MID)
ROUTE=Lookup("FLIGHTS","RTE_ID",MID)
GATE=Lookup("FLIGHTS","GTE_ID",MID)
#
SET FLIGHT NUMBER = FLT_NUM
SET FLIGHT COPS = COPS
SET FLIGHT AIRLINE = AIRLINE
SET FLIGHT MODEL = MODEL
SET FLIGHT DESTINATION = DESTINATION
SET FLIGHT ORIGIN = ORIGIN
```

Aeroscript Final Approach Metering (continued)

- The next step is to have the mission actually fly the initial route indicated by the RTE_ID you assigned earlier in the FLIGHTS table.

  Assign and fly the initial route via the following steps:

  Click on the Flights tab.

  Click on the RTE sub-tab.

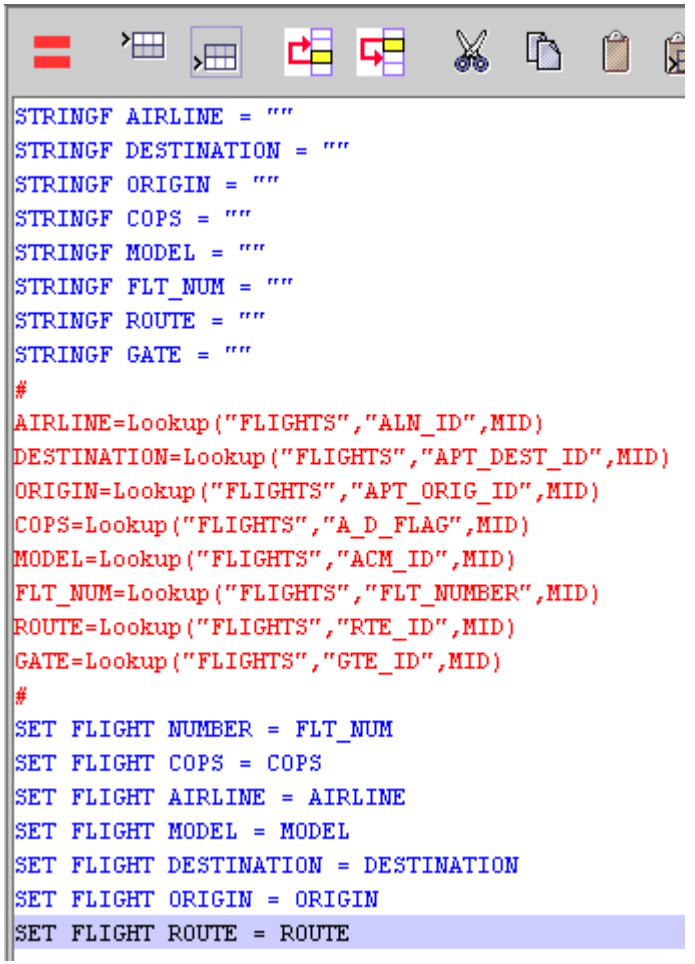
  Click on the Expression assignment type.

  Type ROUTE into the Expression field provided.

  Click on the  button to add the line to the routine.

  This line assigns the route identifier which was stored in the ROUTE variable to the MISSION.

  The mission immediately flies the route. After flying the route the mission will be ready to accept more guidance from the script, or it will exit the model.



```
STRINGF AIRLINE = ""
STRINGF DESTINATION = ""
STRINGF ORIGIN = ""
STRINGF COPS = ""
STRINGF MODEL = ""
STRINGF FLT_NUM = ""
STRINGF ROUTE = ""
STRINGF GATE = ""
#
AIRLINE=Lookup("FLIGHTS","ALN_ID",MID)
DESTINATION=Lookup("FLIGHTS","APT_DEST_ID",MID)
ORIGIN=Lookup("FLIGHTS","APT_ORIG_ID",MID)
COPS=Lookup("FLIGHTS","A_D_FLAG",MID)
MODEL=Lookup("FLIGHTS","ACM_ID",MID)
FLT_NUM=Lookup("FLIGHTS","FLT_NUMBER",MID)
ROUTE=Lookup("FLIGHTS","RTE_ID",MID)
GATE=Lookup("FLIGHTS","GTE_ID",MID)
#
SET FLIGHT NUMBER = FLT_NUM
SET FLIGHT COPS = COPS
SET FLIGHT AIRLINE = AIRLINE
SET FLIGHT MODEL = MODEL
SET FLIGHT DESTINATION = DESTINATION
SET FLIGHT ORIGIN = ORIGIN
SET FLIGHT ROUTE = ROUTE
```

Aeroscript Final Approach Metering (continued)

- In the Aeroscript Editor write your data to the database, then click on the Save button on the main Network Builder toolbar.

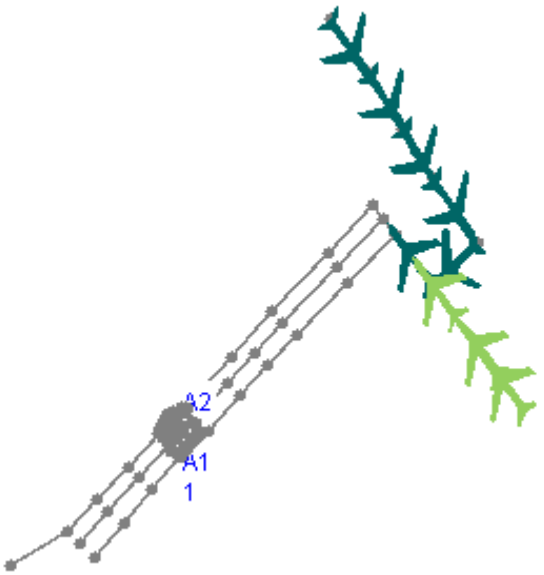
At this time you may test the model by running it.

To run the model go to the Network Builder, select the Simulation -> Run Simulation menu item.

Then select the JSIMMOD tab.

Run JSIMMOD just as you would run SIMMOD.

Assuming no errors have occurred, after you run JSIMMOD you should see something in the 2d Animator similar to the screen capture to the right.



You may notice that none of the flights are traversing any of the the final approach routes. This is because that functionality has not been created yet in the Aeroscript routine.

Aeroscript Final Approach Metering (continued)

- The next step in the routine is to determine which final approach route is least congested. When that route is properly identified then the mission will be directed to traverse that route.

  Because it is possible that all final approach routes are congested it may be necessary that the mission perform an airspace hold temporarily while the congestion clears.

  And because the congested situation may persist it is useful to create a sub-routine which can be called upon repeatedly.

  This subroutine will be created later. Temporarily we will assume it already exists.

  Thus, the next line in our current routine should be the following:

  ```
  SUB getNextRoute RETURN NEXT_RTE_ID
  ```

  When encountered in the script a call will be made to the getNextRoute subroutine. The routine will determine the final arrival route. If all three final arrival routes are congested then getNextRoute will return the phrase ALL_RTES_CONGESTED. This will signal the calling routine that the mission should be delayed and that the final approach route should be computed again.

Aeroscript Final Approach Metering (continued)

- Prior to creating the sub-routine call we must define a variable to hold the hold the route value returned:

  Click on the Misc tab.

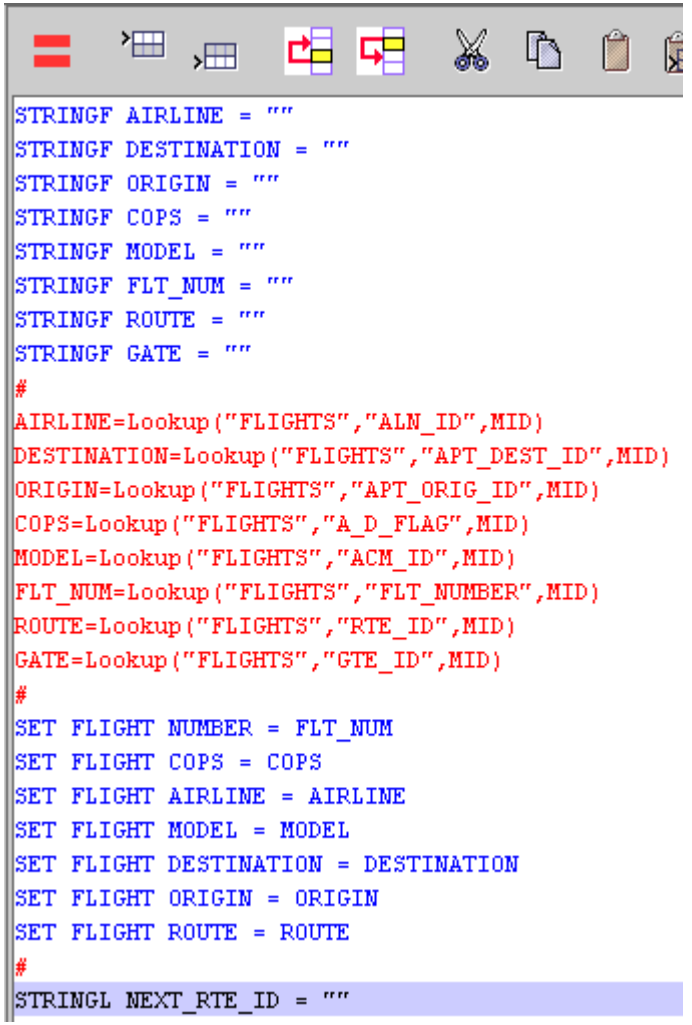
  Click on the Var sub-tab.

  Change the variable scope to Local. This means that the variable to be defined is local to this routine only and cannot be used or seen by other routines, including sub-routines.

  Type the word NEXT_RTE_ID in the variable field.

  Press the  button to insert the line into the routine as shown to the right.

  You have now specified a variable called NEXT_RTE_ID which will be associated with any missions which execute this routine.



```
STRINGF AIRLINE = ""
STRINGF DESTINATION = ""
STRINGF ORIGIN = ""
STRINGF COPS = ""
STRINGF MODEL = ""
STRINGF FLT_NUM = ""
STRINGF ROUTE = ""
STRINGF GATE = ""
#
AIRLINE=Lookup("FLIGHTS","ALN_ID",MID)
DESTINATION=Lookup("FLIGHTS","APT_DEST_ID",MID)
ORIGIN=Lookup("FLIGHTS","APT_ORIG_ID",MID)
COPS=Lookup("FLIGHTS","A_D_FLAG",MID)
MODEL=Lookup("FLIGHTS","ACM_ID",MID)
FLT_NUM=Lookup("FLIGHTS","FLT_NUMBER",MID)
ROUTE=Lookup("FLIGHTS","RTE_ID",MID)
GATE=Lookup("FLIGHTS","GTE_ID",MID)
#
SET FLIGHT NUMBER = FLT_NUM
SET FLIGHT COPS = COPS
SET FLIGHT AIRLINE = AIRLINE
SET FLIGHT MODEL = MODEL
SET FLIGHT DESTINATION = DESTINATION
SET FLIGHT ORIGIN = ORIGIN
SET FLIGHT ROUTE = ROUTE
#
STRINGL NEXT_RTE_ID = ""
```

Aeroscript Final Approach Metering (continued)

To create the call to the subroutine:

Click on the Misc tab.

Click on the Misc sub-tab.

Type

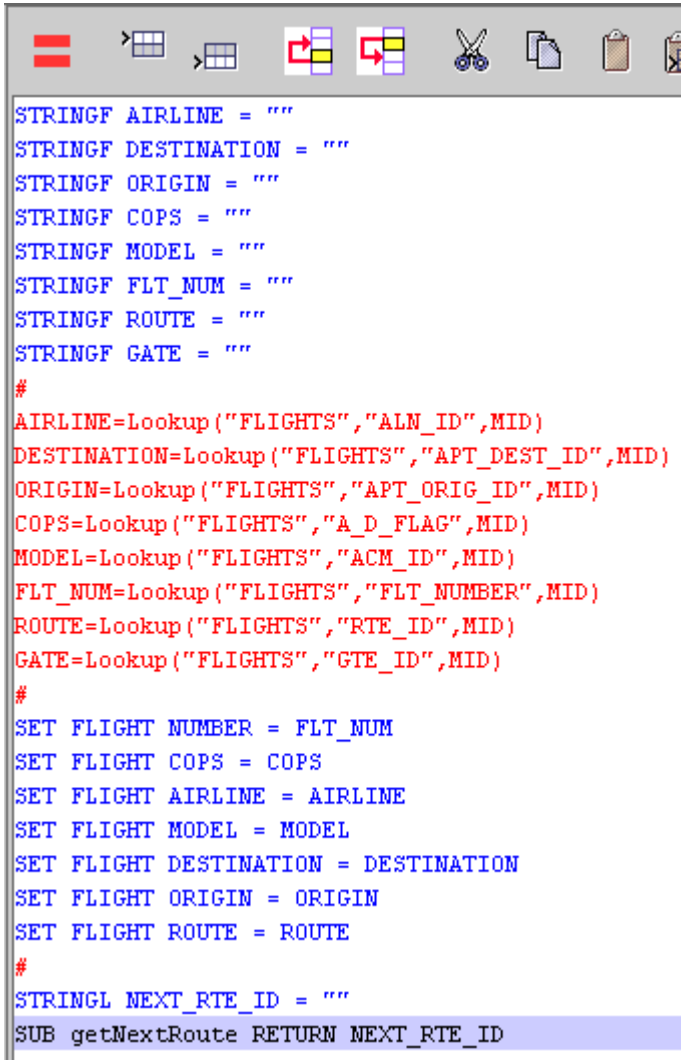
SUB getNextRoute RETURN NEXT_RTE_ID

into the Edit field provided.

Click on the  button to add the line to the routine.

This line calls the getNextRoute sub-routine.

Via the getNextRoute routine the mission immediately checks for a non-congested route. If none is found the phrase ALL_RTES_CONGESTED is returned. If a suitable route is found then that route identifier is returned.



```
STRINGF AIRLINE = ""
STRINGF DESTINATION = ""
STRINGF ORIGIN = ""
STRINGF COPS = ""
STRINGF MODEL = ""
STRINGF FLT_NUM = ""
STRINGF ROUTE = ""
STRINGF GATE = ""
#
AIRLINE=Lookup("FLIGHTS","ALN_ID",MID)
DESTINATION=Lookup("FLIGHTS","APT_DEST_ID",MID)
ORIGIN=Lookup("FLIGHTS","APT_ORIG_ID",MID)
COPS=Lookup("FLIGHTS","A_D_FLAG",MID)
MODEL=Lookup("FLIGHTS","ACM_ID",MID)
FLT_NUM=Lookup("FLIGHTS","FLT_NUMBER",MID)
ROUTE=Lookup("FLIGHTS","RTE_ID",MID)
GATE=Lookup("FLIGHTS","GTE_ID",MID)
#
SET FLIGHT NUMBER = FLT_NUM
SET FLIGHT COPS = COPS
SET FLIGHT AIRLINE = AIRLINE
SET FLIGHT MODEL = MODEL
SET FLIGHT DESTINATION = DESTINATION
SET FLIGHT ORIGIN = ORIGIN
SET FLIGHT ROUTE = ROUTE
#
STRINGL NEXT_RTE_ID = ""
SUB getNextRoute RETURN NEXT_RTE_ID
```

Aeroscript Final Approach Metering (continued)

- The next step in is to test the value returned to determine if a suitable route has been found. If not, then an airspace delay will be incurred, the count obtained again, and the test repeated. To perform the initial test do the following:

Click on the Logic tab.

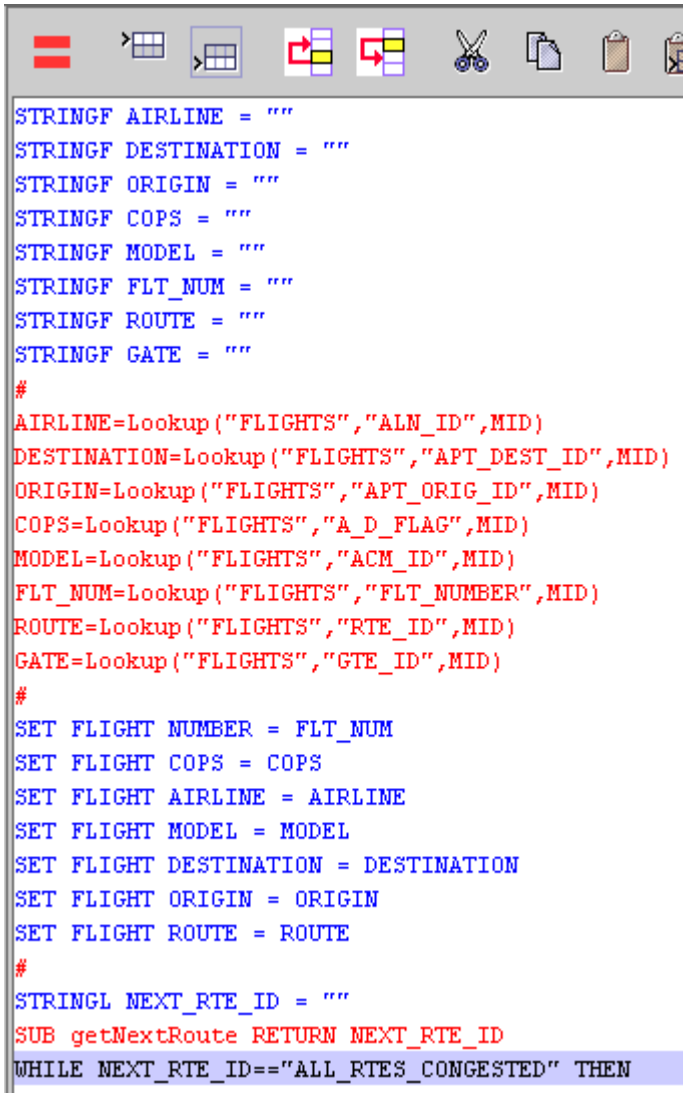
Click on the While sub-tab.

In the Expression field type:

NEXT_RTE_ID == “ALL_RTES_CONGESTED”

Click on the  button to add the line to the routine.

This line checks the value of NEXT_RTE_ID and while it is equal to ALL_RTES_CONGESTED it will direct the mission to execute the lines contained within it's enclosing ENDWHILE statement and continue to loop.



```
STRINGF AIRLINE = ""
STRINGF DESTINATION = ""
STRINGF ORIGIN = ""
STRINGF COPS = ""
STRINGF MODEL = ""
STRINGF FLT_NUM = ""
STRINGF ROUTE = ""
STRINGF GATE = ""
#
AIRLINE=Lookup("FLIGHTS","ALN_ID",MID)
DESTINATION=Lookup("FLIGHTS","APT_DEST_ID",MID)
ORIGIN=Lookup("FLIGHTS","APT_ORIG_ID",MID)
COPS=Lookup("FLIGHTS","A_D_FLAG",MID)
MODEL=Lookup("FLIGHTS","ACM_ID",MID)
FLT_NUM=Lookup("FLIGHTS","FLT_NUMBER",MID)
ROUTE=Lookup("FLIGHTS","RTE_ID",MID)
GATE=Lookup("FLIGHTS","GTE_ID",MID)
#
SET FLIGHT NUMBER = FLT_NUM
SET FLIGHT COPS = COPS
SET FLIGHT AIRLINE = AIRLINE
SET FLIGHT MODEL = MODEL
SET FLIGHT DESTINATION = DESTINATION
SET FLIGHT ORIGIN = ORIGIN
SET FLIGHT ROUTE = ROUTE
#
STRINGL NEXT_RTE_ID = ""
SUB getNextRoute RETURN NEXT_RTE_ID
WHILE NEXT_RTE_ID=="ALL_RTES_CONGESTED" THEN
```

Aeroscript Final Approach Metering (continued)

- If the there are no arrival routes which are uncongested the the mission incurs a delay. To properly account for this delay add a WAIT line to the script:

Click on the Flight tab.

Click on the Wait sub-tab.

In the Indent field type:

2

Click on the Delay tab.

Click on the Expression radio button.

In the Expression field type:

15

Then click on the Initial Code tab.

Click on the Expression radio button.

In the Expression field type:

HA

Click on the Final Code tab.

Click on the Expression radio button.

In the Expression field type:

FH

Then click on the  button to insert the line.

```
#
STRINGL NEXT_RTE_ID = ""
SUB getNextRoute RETURN NEXT_RTE_ID
WHILE NEXT_RTE_ID=="ALL_RTES_CONGESTED" THEN
  SET FLIGHT WAIT =15==HA=FH
```

Aeroscript Final Approach Metering (continued)

- Next create the call to the subroutine again:

Click on the Misc tab.

Click on the Misc sub-tab.

In the Indent field field type:

2

Type

SUB getNextRoute RETURN NEXT_RTE_ID

into the Edit field provided.

Click on the  button to add the line to the routine.

This line calls the getNextRoute sub-routine.

```
#
STRINGL NEXT_RTE_ID = ""
SUB getNextRoute RETURN NEXT_RTE_ID
WHILE NEXT_RTE_ID=="ALL_RTES_CONGESTED" THEN
  SET FLIGHT WAIT =15==HA=FH
  SUB getNextRoute RETURN NEXT_RTE_ID
```

Aeroscript Final Approach Metering (continued)

- Next finish the while loop:

```
#
STRINGL NEXT_RTE_ID = ""
SUB getNextRoute RETURN NEXT_RTE_ID
WHILE NEXT_RTE_ID=="ALL_RTES_CONGESTED" THEN
  SET FLIGHT WAIT =15==HA=FH
  SUB getNextRoute RETURN NEXT_RTE_ID
ENDWHILE
```

Click on the Logic tab.

Click on the ENDWHILE sub-tab.

Click on the  button to add the line to the routine.

This line calls the getNextRoute sub-routine.

To summarize, the while loop checks the route returned, if it isn't a valid route identifier then the mission is given a 15 second HA (finished by the FH code) delay.

After the delay is taken another request for an uncongested route is placed. If a legitimate route is found execution breaks out of the loop. Otherwise the delay and request process repeats until a legitimate route is returned.

Aeroscript Final Approach Metering (continued)

- The next step is to have the mission fly the final approach route indicated by the NEXT_RTE_ID variable.

  Assign and fly the final route via the following steps:

  Click on the Flights tab.

  Click on the RTE sub-tab.

  Click on the Expression assignment type.

  Type NEXT_RTE_ID into the Expression field provided.

  Click on the  button to add the line to the routine.

  This line assigns the route identifier which was stored in the NEXT_RTE_ID variable to the MISSION.

  The mission immediately flies the route. After flying the route the mission will be ready to accept more guidance from the script, or it will exit the model.

```
#
STRINGL NEXT_RTE_ID = ""
SUB getNextRoute RETURN NEXT_RTE_ID
WHILE NEXT_RTE_ID=="ALL_RTES_CONGESTED" THEN
  SET FLIGHT WAIT =15==HA=FH
  SUB getNextRoute RETURN NEXT_RTE_ID
ENDWHILE
#
SET FLIGHT ROUTE = NEXT_RTE_ID
```

Aeroscript Final Approach Metering (continued)

- As described earlier, the next step in the process is to create the subroutine to determine the final arrival route to take.

  This routine will look at the number of aircraft traversing the three available final approaches and choose the one final approach that is least congested.

  If all three final approaches are congested then the word "ALL_RTES_CONGESTED" will be returned.

Aeroscript Final Approach Metering (continued)

- A subroutine in Aeroscript is effectively equal to any other routine in the Aeroscript language. To create the subroutine:

  Press the  button in the Aeroscript editor.

  Type getNextRoute in the pop-up window provided.

  Click on the Ok button provided in the pop-up window.

When you create an Aeroscript routine in this manner all of the lines from the previous routine will be copied into this new routine. You will need to remove those unnecessary lines. To remove the lines:

As a precaution, click the getNextroute identifier in the left column to ensure the routine is the active routine.

Then click on the first line of the script.

Press and hold down the shift key.

While continuing to hold down the shift key, click on the last line of the script.

Click on the  button to remove all lines from the script.

Aeroscript Final Approach Metering (continued)

- Next specify that a Local String variable will be returned from the getNextRoute routine. Please type this:

  STRINGL

  In the Variables out field. Don't forget to press the return (enter) key after typing this data.

  Next you will specify the variable referred to be the STRINGL specification above. Please perform the following steps:

  Click on the Misc tab.

  Click on the Var sub-tab.

  Change the variable type to String.

  Change the variable scope to Local.

  Type the word THE_CHOSEN_RTE in the variable field.

  Type the word ALL_RTES_CONGESTED in the value field.

  Press the  button to insert the line into the routine as shown to the right.

Aeroscript Final Approach Metering (continued)

- Next an integer variable will be defined to hold the number of aircraft flying on a route.

  Each final approach route will be checked for how many aircraft are currently traversing the route. If a route is found with less than two aircraft it will chosen.

  If the number traversing the ARR_FINAL_23R arrival route is two, or greater, then the ARR_FINAL_23L arrival route will be tested.

  If two, or more, aircraft are traversing ARR_FINAL_23L, then arrival route ARR_FINAL_24 will be tested.

  If two, or more, aircraft are traversing ARR_FINAL_24, then the phrase ALL_RTES_CONGESTED will be returned.

Aeroscript Final Approach Metering (continued)

- Define the integer variable that will hold the number of aircraft on final approach on a particular route. To do this:

  Click on the Misc tab.

  Click on the Var sub-tab.

  Change the variable type to Integer.

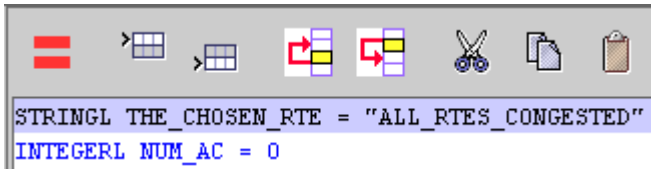
  Change the variable scope to Local.

  Type the word NUM_AC in the variable field.

  Type 0 in the value field.

  Press the  button to insert the line into the routine as shown to the right.

  You have now specified a variable called NUM_AC which is assigned the value of zero.



Aeroscript Final Approach Metering (continued)

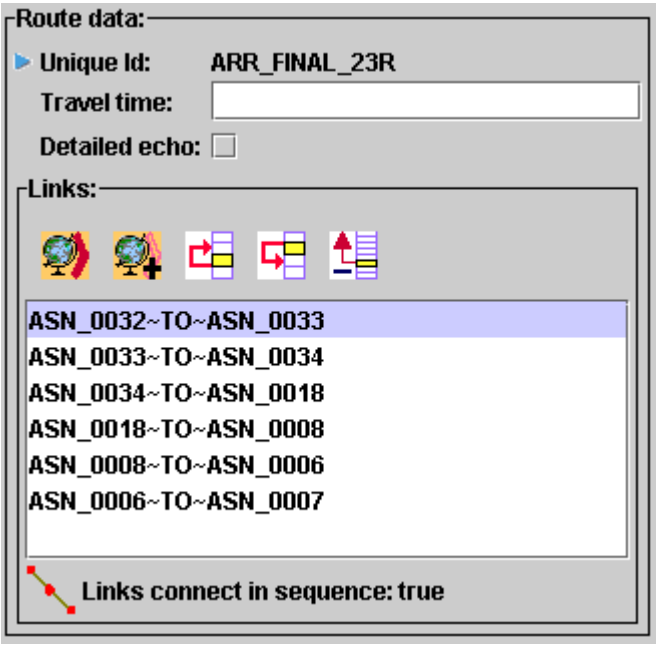
- Next compute the number of aircraft traversing arrival route ARR_FINAL_23R.

  To perform this task we must know which airspace links belong to this particular route.

  Bring up the Route Editor from the Network Builder and click on the ARR_FINAL_23R route. The needed links will be listed on the tool.

  Because it is not guaranteed that your links will match this tutorial's links the links in this tutorial will be show to the right. Substitute your links where appropriate.

  To remember your links you may need to write them on separate sheet of paper or to position the windows on your computer so that you can see the Route Editor and the Aeroscript Editor simultaneously.



Aeroscript Final Approach Metering (continued)

To compute and store the number of aircraft traversing the ARR_FINAL_23R:

Return to the Aeroscript Editor.

Click on the Misc tab.

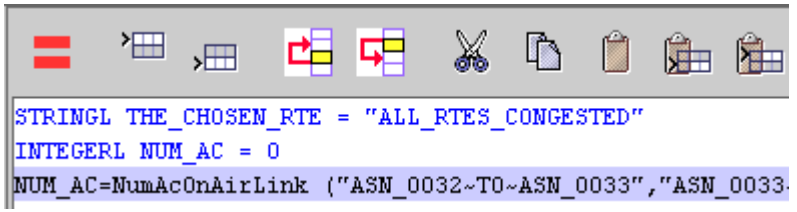
Click on the Misc sub-tab.

In the Edit field type the following (please substitute your own airspace links):

NUM_AC=NumAcOnAirLink (”ASN_0032~TO~ASN_0033”,”ASN_0033~TO~ASN_0034” ,”ASN_0034~TO~ASN_0018”, ”ASN_0018~TO~ASN_0008”,”ASN_0018~TO~ASN_0008”,”ASN_0006~TO~ASN_0007”)

Please note that if your line of data above is more that 255 characters you will need to create two separate lines. The second line would look something like the following:

NUM_AC= NUM_AC + NumAcOnAirLink (.....)



Click on the  button to place the above line(s) into the routine.

Aeroscript Final Approach Metering (continued)

- Next a test is performed to determine if the number of aircraft just computed is less than 2, if it is then the ARR_FINAL_23R route will be assigned.

  To perform the test:

  Click on the Logic tab.

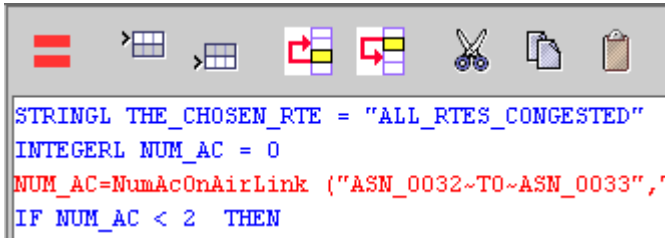
  Click on the If sub-tab.

  Type the following in the expression field:

  NUM_AC < 2

  Press the  button to insert the line into the routine as shown to the right.

  You have now have a test statement which determines if the current number of aircraft on the ARR_FINAL_23R is less than two.



Aeroscript Final Approach Metering (continued)

- Next the variable THE_CHOSEN_RTE is assigned the route value of ARR_FINAL_23R.

  Click on the Misc tab.

  Click on the Misc sub-tab.

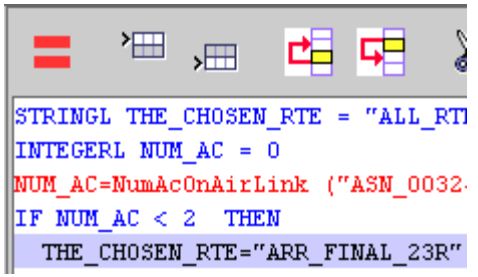
  Change the Indent field to 2.

  Type

  THE_CHOSEN_RTE=”ARR_FINAL_23R”

  in the edit field.

  Press the  button to insert the line into the routine as shown to the right.



Aeroscript Final Approach Metering (continued)

- Next an else statement is added to the routine.

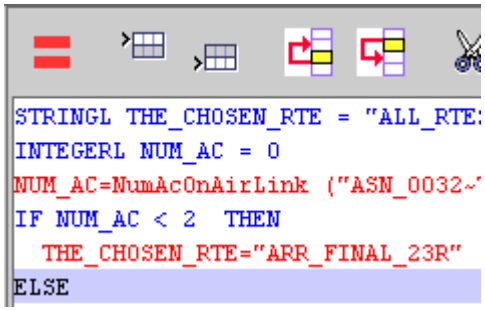
  If the NUM_AC on the ARR_FINAL_23R is greater than or equal to two then we would then want to check for how many aircraft are traversing ARR_FINAL_23L.

  Therefore:

  Click on the Logic tab.

  Click on the Else sub-tab.

  Press the  button to insert the line into the routine as shown to the right.



Aeroscript Final Approach Metering (continued)

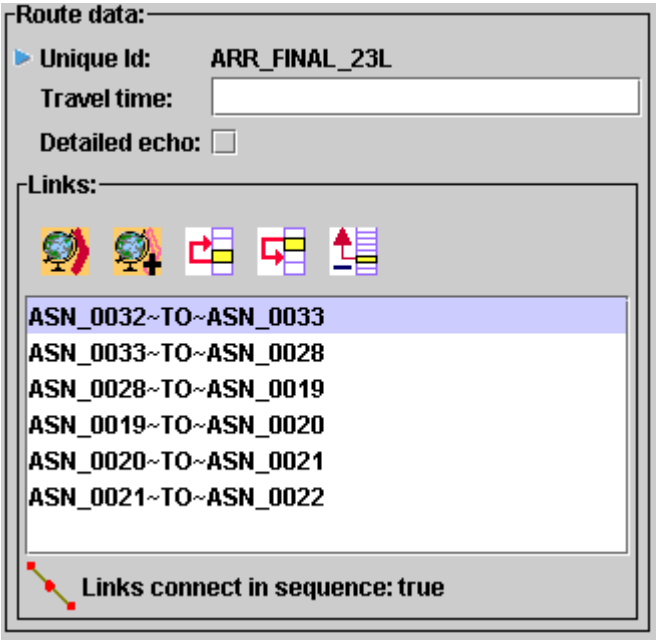
- Next compute the number of aircraft traversing arrival route ARR_FINAL_23L.

  To perform this task we must know which airspace links belong to this particular route.

  Bring up the Route Editor from the Network Builder and click on the ARR_FINAL_23L route. The needed links will be listed on the tool.

  Because it is not guaranteed that your links will match this tutorial's links the links in this tutorial will be show to the right. Substitute your links where appropriate.

  To remember your links you may need to write them on separate sheet of paper or to position the windows on your computer so that you can see the Route Editor and the Aeroscript Editor simultaneously.



Aeroscript Final Approach Metering (continued)

To compute and store the number of aircraft traversing the ARR_FINAL_23L:

Return to the Aeroscript Editor.

Click on the Misc tab.

Click on the Misc sub-tab.

Type 2 in the Indent field.

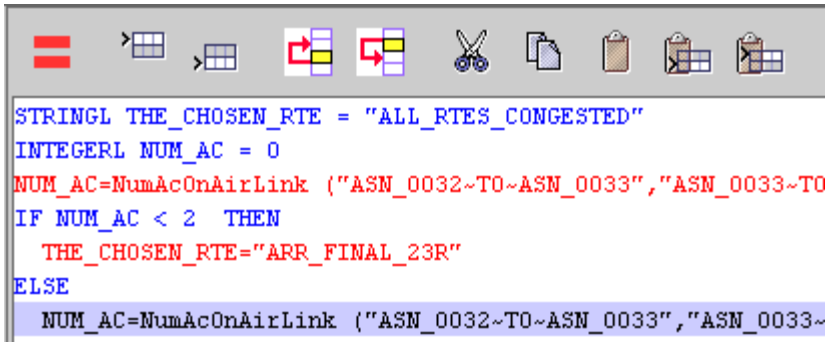
In the Edit field type the following (please substitute your own airspace links):

NUM_AC=NumAcOnAirLink (”ASN_0032~TO~ASN_0033”,”ASN_0033~TO~ASN_0028” , “ASN_0028~TO~ASN_0019”, ”ASN_0019~TO~ASN_0020”,”ASN_0020~TO~ASN_0021”,”ASN_0021~TO~ASN_0022”)

Please note that if your line of data above is more that 255 characters you will need to create two separate lines. The second line would look something like the following:

NUM_AC= NUM_AC + NumAcOnAirLink (.....)

Click on the  button to place the above line(s) into the routine.



Aeroscript Final Approach Metering (continued)

- Next a test is performed to determine if the number of aircraft just computed is less than 2, if it is then the ARR_FINAL_23L route will be assigned.

  To perform the test:

  Click on the Logic tab.

  Click on the If sub-tab.

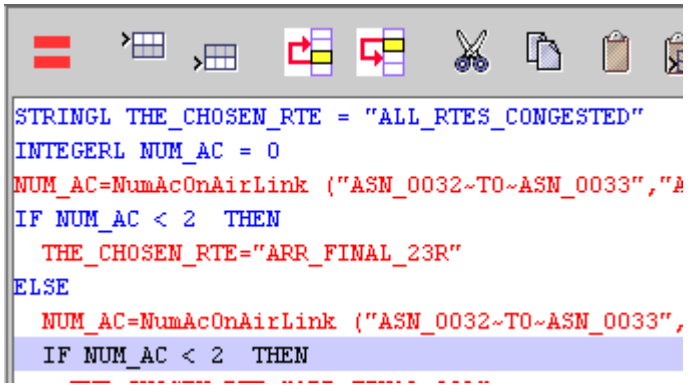
  Change the Indent field to be 2.

  Type the following in the expression field:

  NUM_AC < 2

  Press the  button to insert the line into the routine as shown to the right.

  You have now have a test statement which determines if the current number of aircraft on the ARR_FINAL_23L is less than two.



Aeroscript Final Approach Metering (continued)

- Next the variable THE_CHOSEN_RTE is assigned the route value of ARR_FINAL_23L.

  Click on the Misc tab.

  Click on the Misc sub-tab.

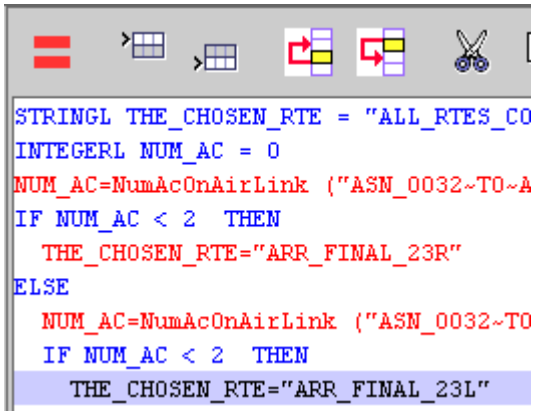
  Change the Indent field to 4.

  Type

  THE_CHOSEN_RTE=”ARR_FINAL_23L”

  in the edit field.

  Press the  button to insert the line into the routine as shown to the right.



Aeroscript Final Approach Metering (continued)

- Next an else statement is added to the routine.

  If the NUM_AC on the ARR_FINAL_23L is greater than or equal to two then we would then want to check for how many aircraft are traversing ARR_FINAL_24.

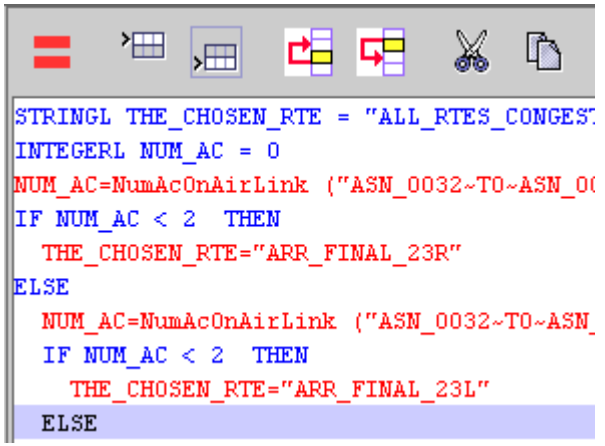
  Therefore:

  Click on the Logic tab.

  Click on the Else sub-tab.

  Change the Indent value to 2.

  Press the  button to insert the line into the routine as shown to the right.



Aeroscript Final Approach Metering (continued)

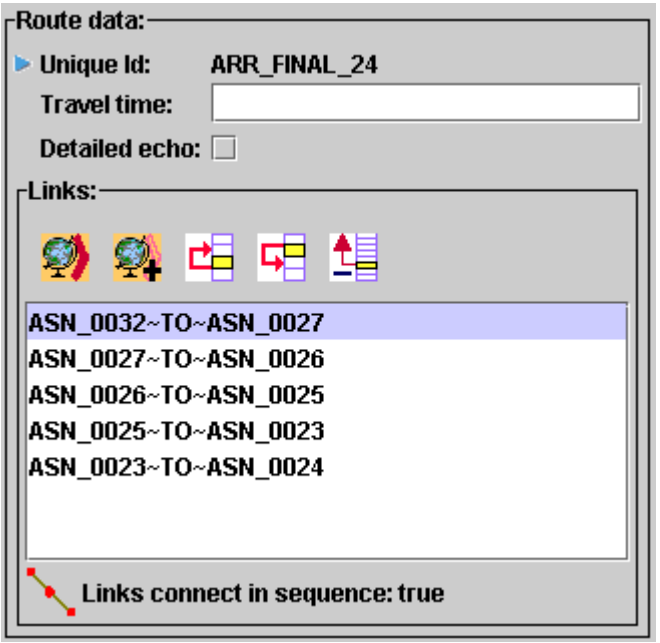
- Next compute the number of aircraft traversing arrival route ARR_FINAL_24.

  To perform this task we must know which airspace links belong to this particular route.

  Bring up the Route Editor from the Network Builder and click on the ARR_FINAL_24 route. The needed links will be listed on the tool.

  Because it is not guaranteed that your links will match this tutorial's links the links in this tutorial will be show to the right. Substitute your links where appropriate.

  To remember your links you may need to write them on separate sheet of paper or to position the windows on your computer so that you can see the Route Editor and the Aeroscript Editor simultaneously.



Aeroscript Final Approach Metering (continued)

To compute and store the number of aircraft traversing the ARR_FINAL_24:

Return to the Aeroscript Editor.

Click on the Misc tab.

Click on the Misc sub-tab.

Type 4 in the Indent field.

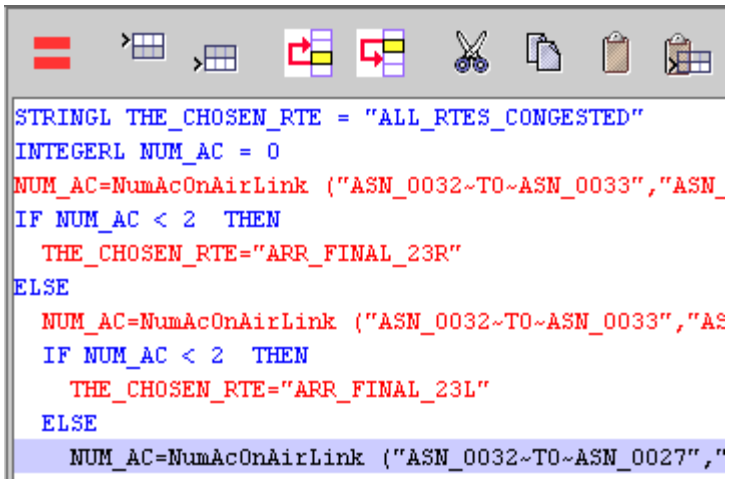
In the Edit field type the following (please substitute your own airspace links):

NUM_AC=NumAcOnAirLink (”ASN_0032~TO~ASN_0027”,”ASN_0027~TO~ASN_0026” , “ASN_0026~TO~ASN_0025”, ”ASN_0025~TO~ASN_0023”,”ASN_0023~TO~ASN_0024”)

Please note that if your line of data above is more that 255 characters you will need to create two separate lines. The second line would look something like the following:

NUM_AC= NUM_AC + NumAcOnAirLink (.....)

Click on the  button to place the above line(s) into the routine.



Aeroscript Final Approach Metering (continued)

- Next a test is performed to determine if the number of aircraft just computed is less than 2, if it is then the ARR_FINAL_24 route will be assigned.

  To perform the test:

  Click on the Logic tab.

  Click on the If sub-tab.

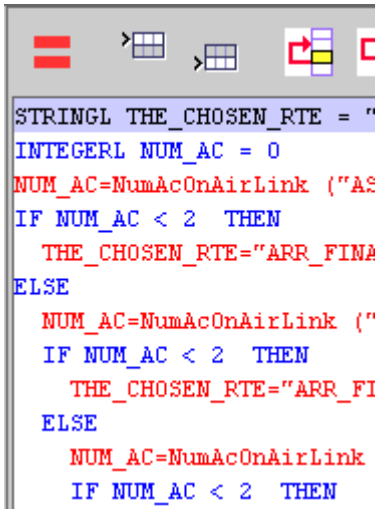
  Change the Indent field to be 2.

  Type the following in the expression field:

  NUM_AC < 2

  Press the  button to insert the line into the routine as shown to the right.

  You have now have a test statement which determines if the current number of aircraft on the ARR_FINAL_24 is less than two.



Aeroscript Final Approach Metering (continued)

- Next the variable THE_CHOSEN_RTE is assigned the route value of ARR_FINAL_24.

  Click on the Misc tab.

  Click on the Misc sub-tab.

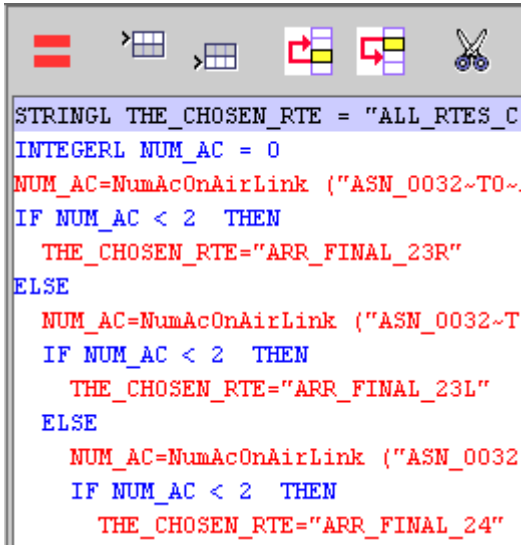
  Change the Indent field to 6.

  Type

  THE_CHOSEN_RTE=”ARR_FINAL_24”

  in the edit field.

  Press the  button to insert the line into the routine as shown to the right.



Aeroscript Final Approach Metering (continued)

- Next an ENDIF statement is added to the routine.

  The ENDIF closes the previous matching IF NUM_AC < 2 THEN statement.

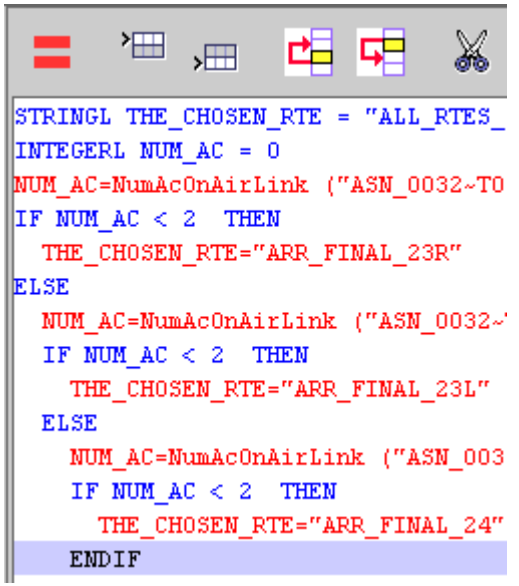
  Therefore:

  Click on the Logic tab.

  Click on the EndIf sub-tab.

  Change the Indent value to 4.

  Press the  button to insert the line into the routine as shown to the right.



Aeroscript Final Approach Metering (continued)

- Next an ENDIF statement is added to the routine.

  The ENDIF closes the previous matching IF NUM_AC < 2 THEN statement.

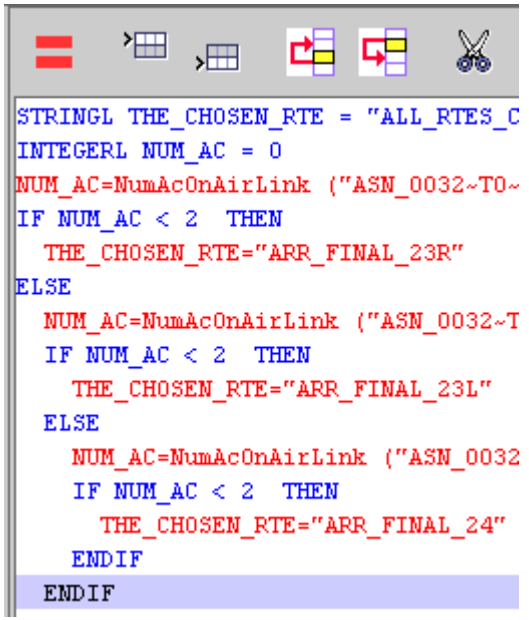
  Therefore:

  Click on the Logic tab.

  Click on the EndIf sub-tab.

  Change the Indent value to 2.

  Press the  button to insert the line into the routine as shown to the right.



Aeroscript Final Approach Metering (continued)

- Next an ENDIF statement is added to the routine.

  The ENDIF closes the previous matching IF NUM_AC < 2 THEN statement.

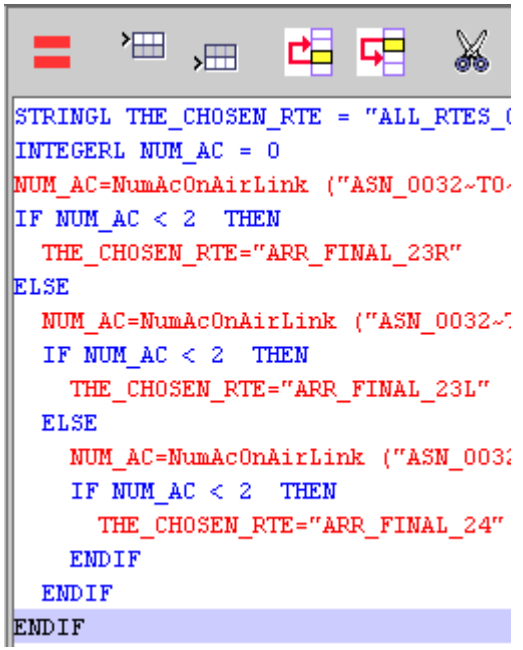
  Therefore:

  Click on the Logic tab.

  Click on the EndIf sub-tab.

  Change the Indent value to 0.

  Press the  button to insert the line into the routine as shown to the right.



Aeroscript Final Approach Metering (continued)

- Finally a RETURN statement is added to the routine.

  The RETURN statement returns back the THE_CHOSEN_RTE to the routine which called the getNextRoute sub-routine.

  Therefore:

  Click on the Logic tab.

  Click on the Return sub-tab.

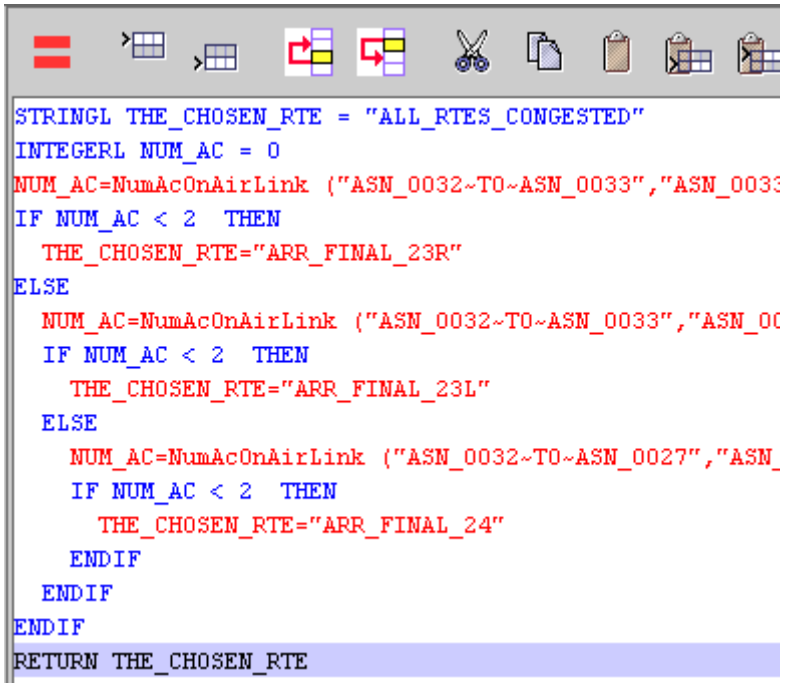
  Change the Indent value to 0.

  Click on the Expression radio button.

  In the Expression field type:

  THE_CHOSEN_RTE

  Press the  button to insert the line into the routine as shown to the right.



```
STRINGL THE_CHOSEN_RTE = "ALL_RTES_CONGESTED"
INTEGERL NUM_AC = 0
NUM_AC=NumAcOnAirLink ("ASN_0032~TO~ASN_0033","ASN_0033
IF NUM_AC < 2  THEN
  THE_CHOSEN_RTE="ARR_FINAL_23R"
ELSE
  NUM_AC=NumAcOnAirLink ("ASN_0032~TO~ASN_0033","ASN_00
  IF NUM_AC < 2  THEN
    THE_CHOSEN_RTE="ARR_FINAL_23L"
  ELSE
    NUM_AC=NumAcOnAirLink ("ASN_0032~TO~ASN_0027","ASN_
    IF NUM_AC < 2  THEN
      THE_CHOSEN_RTE="ARR_FINAL_24"
    ENDIF
  ENDIF
ENDIF
RETURN THE_CHOSEN_RTE
```

Aeroscript Final Approach Metering (continued)

- At this time make sure you save your data and then you may run the JSIMMOD model.

  The results in the 2D Animator should look similar to the following: