Chapter 4
More Detailed Modeling: Airport Revisited

There is normally more than one type of check-in in an airport, as well as more than one type of passenger (i.e. a curbside check-in or a no checked-bags check-in traveler). Our goal is to embellish the airport model from the last chapter, to handle routing to different check-in stations, varying arrival rates of passengers, and varying types of passengers with rate tables and data tables. We will also introduce a means of giving entities characteristics, called state variables, which can be used to provide various distinctions.

Part 4.1: Choice of Paths

As in the previous chapter, our primary concern is with the amount time it takes the passengers to reach the security checking station and in determining the number of needed check-in stations.

Step 1: Delete the EXPERIMENT object in the prior chapter model by right clicking on the Experiment in the [*Navigation*] Panel.[28] We will embellish the previous airport model to allow a choice of three different routes for passengers arriving at the airport. Passengers who need to check bags and get tickets can either check in at the curbside station (10%) or check in at the main station inside the airport (70%), while some passengers can proceed directly to the security check point (20%). Refer to Figure 4.1 which depicts the modified airline check-in process to be modeled.

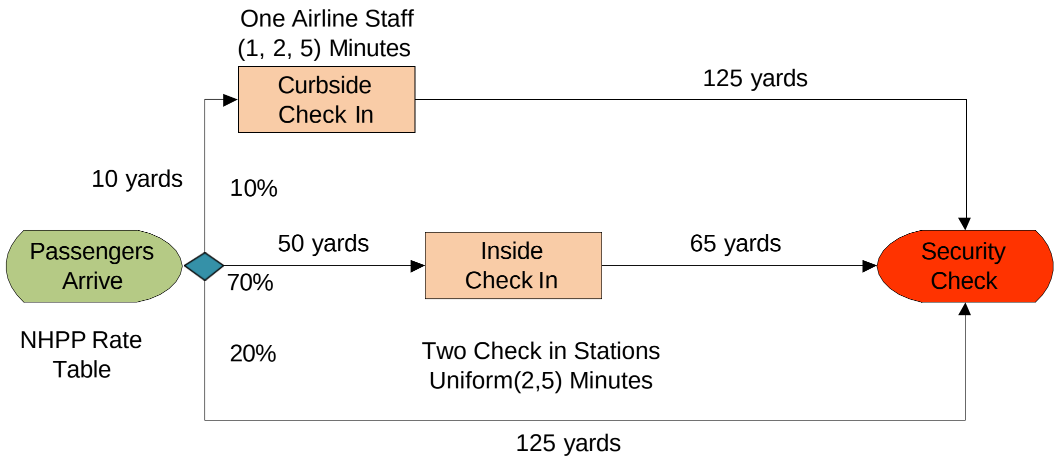


Figure 4.1: Modified Airline Check-in Process

Route1: Passengers who use the curbside check-in need to walk 10 yards to reach the station where a single airline staff member takes between one and five minutes, with a most likely time of two minutes, to check-in a passenger. Once they have checked in, they walk 125 yards to the security check line.

Route2: Passengers who have checked in online and do not need to check any luggage, only need to walk the 125 yards to the security check line once they arrive to the airport.

Route3: Passengers who plan to use the inside check-in first need to walk 50 yards to reach the station where there are currently two airline personnel (instead of four previously) checking passengers into

[28] The [*Navigation Panel*] in the upper right corner contains all of the models, objects, and experiments in the project. You can access the properties, rename the item, or delete the item from the project.

the airport. It takes between three and ten minutes uniformly to process each passenger. Once they have checked in, they then walk 65 yards to reach the security check line.

Step 2: Modify the model from last chapter to include the new curbside check-in (SERVER) as well as the direct path (i.e., Route 2) from the arrivals to the security gate as seen in Figure 4.2. Specify the properties according to description in Figure 4.1.

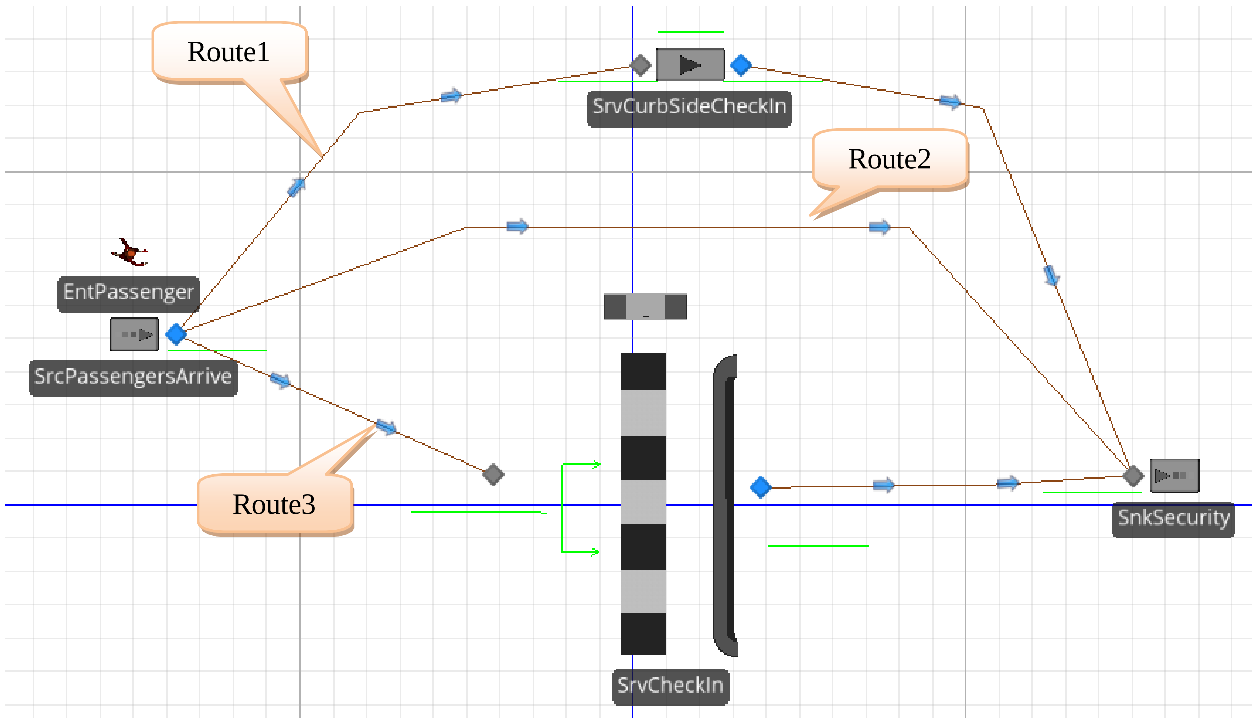


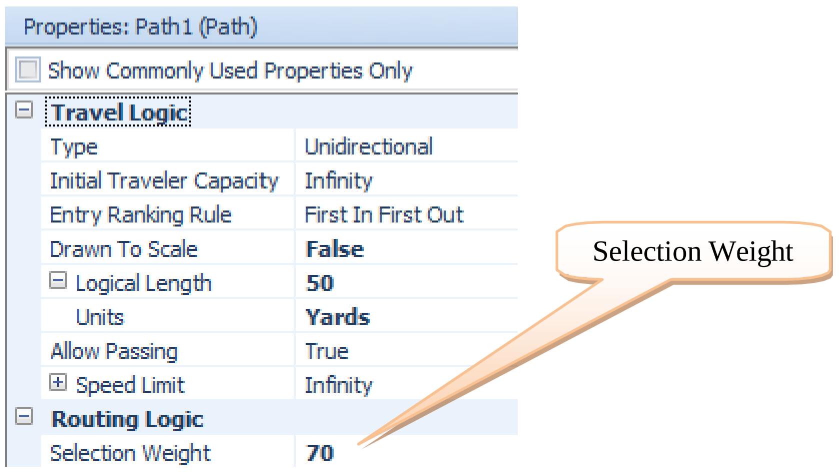
Figure 4.2: Creating Different Routing Paths

Step 3: It has been observed that passengers follow the probabilities in Table 4.1 in determining which route they will follow.

Table 4.1: Passenger Type Percentages

Check-in Type	Percentage
Curbside Check-in	10%
Inside Check-in	70%
No Check-in (Carry on)	20%

Step 4: To model the decision of a passenger to choose their check-in route the Selection Weight property of Path objects will be utilized. The probability of choosing a particular path is that path's individual selection weight divided by the sum of all path selection weights from that node object. Place the proper weights to each different check-in route (see Figure 4.3).


Properties: Path1 (Path)

Show Commonly Used Properties Only

Travel Logic

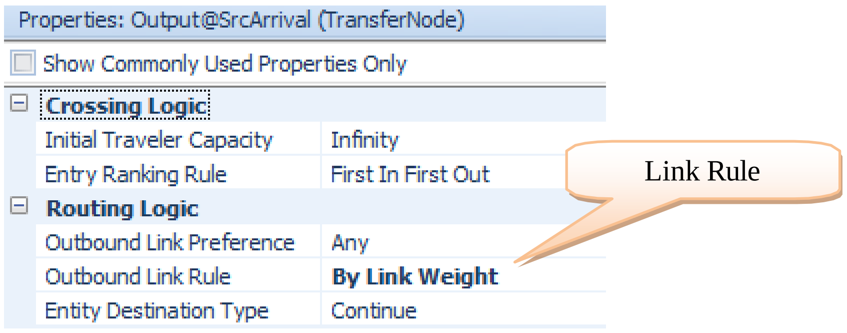
Type	Unidirectional
Initial Traveler Capacity	Infinity
Entry Ranking Rule	First In First Out
Drawn To Scale	**False**
Logical Length	**50**
Units	**Yards**
Allow Passing	True
Speed Limit	Infinity

Routing Logic

Selection Weight	**70**

Figure 4.3: Weighting Paths

Step 5: Remember to select the Outbound Link Rule to "By Link Weight" on the TRANSFERNODE **Output@SrcPassengersArrive** (see Figure 4.4).29


Properties: Output@SrcArrival (TransferNode)

Show Commonly Used Properties Only

Crossing Logic

Initial Traveler Capacity	Infinity
Entry Ranking Rule	First In First Out

Routing Logic

Outbound Link Preference	Any
Outbound Link Rule	**By Link Weight**
Entity Destination Type	Continue

Figure 4.4: Outbound Link Rule

Step 6: Run the model for one, twenty-four hour replication (no "Experiment" needed). You might want to improve on our animation.

Question 1: What is the average number of passengers waiting at Curbside and the Inside Check-in?

____

Question 2: What is the average wait time at each Check-in?

____

Question 3: What is the average time it takes for a passenger to check-in (from entry to security)?

____

Part 4.2: Changing Arrival Rate

Passengers do not arrive at a constant rate (i.e., homogeneous process) throughout the entire day in a real airport (even though they may arrive randomly), so we should expand our model to handle such phenomena. Our approach will be to allow the *arrival rates* to change throughout the day. To model this changing rate, we will use a data object in SIMIO called a *Rate Table* and reference the *Rate Table* we create in the arrival logic of our Source object. It is important to understand that the interarrival times for this rate will be assumed to be Exponentially distributed, but the mean of this distribution changes with time. If an interarrival process has an

[29] Although "Shortest Path" is the default *Outbound Link Rule*, "**By Link Weight**" rule will still be used unless the entity destination has been assigned to a particular node. Therefore, we could have used the default rule.

Exponential distribution, then it is known statistically that the number of arrivals per unit time is Poisson distributed. Since the parameter (its mean) changes with time, it is formally referred to as a "non-homogeneous Poisson" arrival process, also called a NHPP (Non-Homogeneous Poisson Process). Therefore when you use a SIMIO Rate Table you are assuming the arrival process is a NHPP. Table 4.2 shows the hourly rate over the 24 hour day where more passengers on average arrive during the morning and dinner time hours. The zero rates imply no arrivals during that time period.

Table 4.2: Hourly Arrival Rate During Each Hour

Begin Time	End Time	Hourly Rate
Midnight	1am	0
1am	2am	0
2am	3am	0
3am	4am	0
4am	5am	0
5am	6am	30
6am	7am	90
7am	8am	100
8am	9am	75
9am	10am	60
10am	11am	60
11am	Noon	30
Noon	1pm	30
1pm	2pm	30
2pm	3pm	60
3pm	4pm	60
4pm	5pm	75
5pm	6pm	100
6pm	7pm	90
7pm	8pm	30
8pm	9pm	0
9pm	10pm	0
10pm	11pm	0
11pm	Midnight	0

Step 1: Arrivals are only present from 5am to 8pm for a total of 15 hours each day. A simulation run length of 24 hours would cause the time-based statistics like number in station, number in queue, and scheduled utilization to be computed when there were no arrivals, causing lower than expected values for these statistics. Therefore we will set the *Starting Time* within the "*Run Setup"* tab to 5:00 am and the *Ending Type* to *Specific Ending Time* at 8:00 pm as shown in Figure 4.5

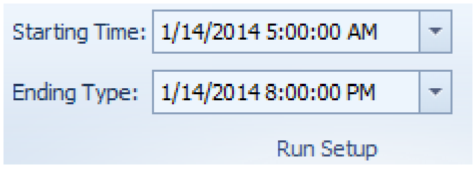
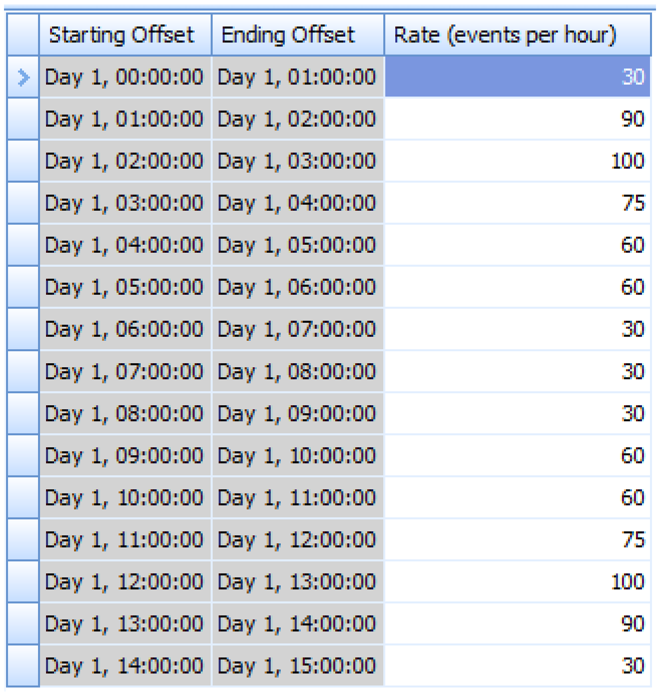

Starting Time: 1/14/2014 5:00:00 AM
Ending Type: 1/14/2014 8:00:00 PM

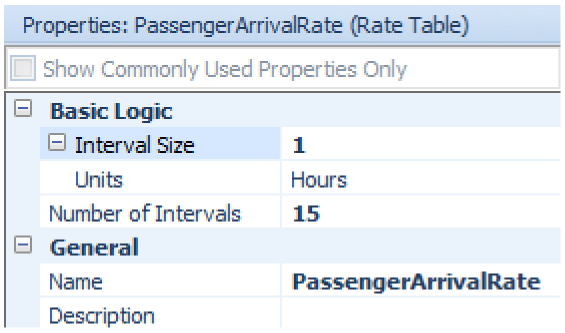
Figure 4.5: Replication Start and Stop Time

Step 2: To create a RATE TABLE, select the "*Data*" tab and then click *Create →Rate Table.* Name the table **PassengerArrivalRate**. Set the *Interval Size* to one hour and the *Number of Intervals* to 15, corresponding to the run length as seen in Figure 4.6. While you can modify the size (e.g., half hour increments) and number of time intervals during which the arrival rate is constant, you cannot modify the rate, which is fixed at *arrivals per hour*.

Step 3: Enter values in the rate table as shown in Figure 4.6.



	Starting Offset	Ending Offset	Rate (events per hour)
›	Day 1, 00:00:00	Day 1, 01:00:00	30
	Day 1, 01:00:00	Day 1, 02:00:00	90
	Day 1, 02:00:00	Day 1, 03:00:00	100
	Day 1, 03:00:00	Day 1, 04:00:00	75
	Day 1, 04:00:00	Day 1, 05:00:00	60
	Day 1, 05:00:00	Day 1, 06:00:00	60
	Day 1, 06:00:00	Day 1, 07:00:00	30
	Day 1, 07:00:00	Day 1, 08:00:00	30
	Day 1, 08:00:00	Day 1, 09:00:00	30
	Day 1, 09:00:00	Day 1, 10:00:00	60
	Day 1, 10:00:00	Day 1, 11:00:00	60
	Day 1, 11:00:00	Day 1, 12:00:00	75
	Day 1, 12:00:00	Day 1, 13:00:00	100
	Day 1, 13:00:00	Day 1, 14:00:00	90
	Day 1, 14:00:00	Day 1, 15:00:00	30



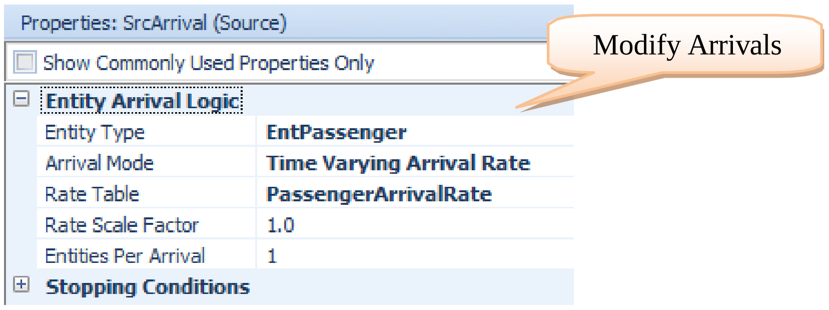
Properties: PassengerArrivalRate (Rate Table)

Show Commonly Used Properties Only

Basic Logic	
Interval Size	**1**
Units	Hours
Number of Intervals	**15**
General	
Name	**PassengerArrivalRate**
Description	

Figure 4.6: The Rate Table (15 intervals)

Step 4: Modify the **SrcPassengersArrive** source accordingly (see Figure 4.7 for details). *SIMIO refers to the NHPP as a "Time Varying Arrival Rate"* ARRIVAL MODE.



Properties: SrcArrival (Source)

Show Commonly Used Properties Only

Entity Arrival Logic	
Entity Type	**EntPassenger**
Arrival Mode	**Time Varying Arrival Rate**
Rate Table	**PassengerArrivalRate**
Rate Scale Factor	1.0
Entities Per Arrival	1
Stopping Conditions	

Figure 4.7: Specifying a Time Varying Arrival Rate

Step 5: Run the model again and observe the results.

Question 4: What are the average Wait Times at each check-in location (Curbside and Inside)?

____

Question 5: What is the average check-in time for a passenger (Curbside and Inside)?

____

If the length of the simulation (replication/run length) exceeds the length of the rate table, then the rate table simply repeats until the simulation replication is terminated. So if the run length should be longer and there should be no arrivals during that time, then the rate table should contain zeroes accordingly.

Part 4.3: State Variables, Properties, and Data Tables

Properties and state variables can be characteristics of the "Model" or of the "ModelEntities". A "property" of an object is a part of the object's definition and is assigned at the time an object is created, but it can't be changed during the simulation. Properties can be expressions, so their evaluation and the time of their creation can yield different values. In contrast, a "state variable" is a characteristic of an object that can be changed during the simulation. Generally, when properties and state variables are associated with the model, they are global in the sense that their values are visible throughout the model. When properties and state variable are associated with entities, then you need to have access to that model entity in order to access the state variable value.

In an actual airport more than one type of passenger arrives. Some are continental travelers while others are inter-continental travelers. Still there are some passengers which have handicaps and require special attention or are with families. The different types of passengers result in different processing times. To model these three cases we will use a combination of state variables and properties to store the information and to reference the information when needed.

There is a 33% chance for each type of passenger to arrive to our airport. The processing times at the curbside check-in and the inside check-in varies with the type of passenger. Specifically, see Figure 4.8 for the processing times.

Passenger Type	**Curbside Check-In Time (minutes)**	**Inside Check-In Time (minutes)**
1	Triangular(1, 2, 5)	Uniform(2, 5)
2	Triangular(2, 3, 6)	Uniform(3, 5)
3	Triangular(3, 4, 7)	Uniform(4, 6)

Figure 4.8: Check-In Times for Different Passenger Types

The MODELENTITY will need to reference these data as specifications at the check-in counters

Step 1: Since the passenger type is not a general characteristic of SIMIO model entities, we need either to find a SIMIO characteristic we can use to represent the type or define such as characteristic ourselves. User-defined characteristics whose values can be re-assigned are the "STATE VARIABLES." A state variable can be a characteristic of the "Model" (namely global) or a characteristic of the "ModelEntity". Here we are interested in a characteristic of a MODELENTITY.

Step 2: To create a MODELENTITY State Variable, first select the "**ModelEntity**" object in the *Navigation* panel. Click *"States*" panel icon in the *"Definitions"* tab. It is convenient to number our passenger type, so select a DISCRETE "Integer" state from the ribbon. Let's name it **EStaPassengerType**[30]. Next we need to assign each MODELENTITY a value for passenger type.

Question 6: When you define your new state variable, what other state variables are already defined for MODELENTITIES?

____

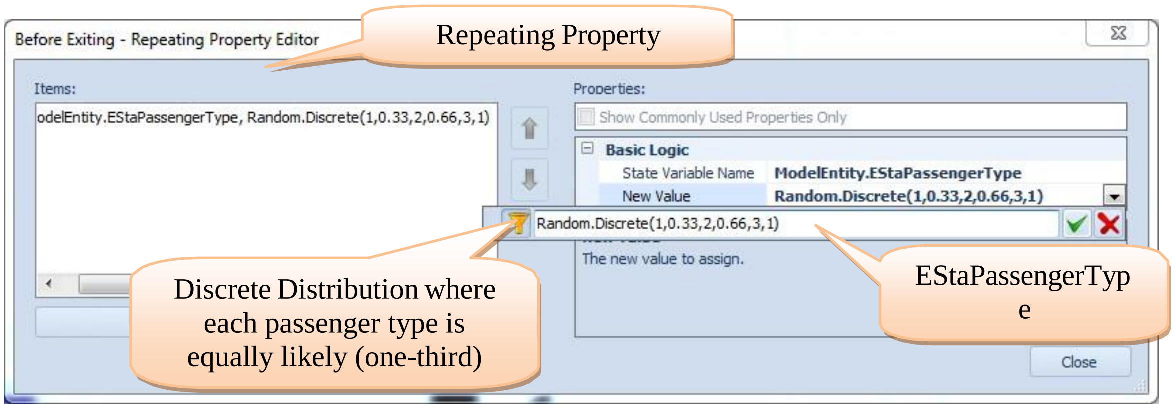
Step 3: We need to assign a value to **EStaPassengerType** for each entity. To obtain a random assignment of passenger type to entities (i.e., 33% chance for each type), we must use the `Discrete` Distribution (entered in cumulative distribution form).

```
Random.Discrete(1, 0.33, 2, 0.66, 3, 1.00)
```

Step 4: One convenient place is to make this assignment at the **SrcPassengersArrive** source object in the "State Assignments" property. Click on the plus box to reveal "*Before Exiting*". Then click the ellipses box at the end of that specification to bring up the "REPEATING PROPERTY EDITOR"[31], which in this case allows state value assignments.

[30] We will use "E" to when denoting "Entity" and "Sta" to denote "State variable".

[31] It's called a Repeating Property Editor, even though, in this case, it is assigning state variables


Before Exiting - Repeating Property Editor
Repeating Property
Items:
Properties:
Show Commonly Used Properties Only
Basic Logic
Discrete Distribution where each passenger type is equally likely (one-third)
EStaPassengerType
Close

odelEntity.EStaPassengerType, Random.Discrete(1,0.33,2,0.66,3,1)

State Variable Name	ModelEntity.EStaPassengerType
New Value	Random.Discrete(1,0.33,2,0.66,3,1)

Random.Discrete(1,0.33,2,0.66,3,1)

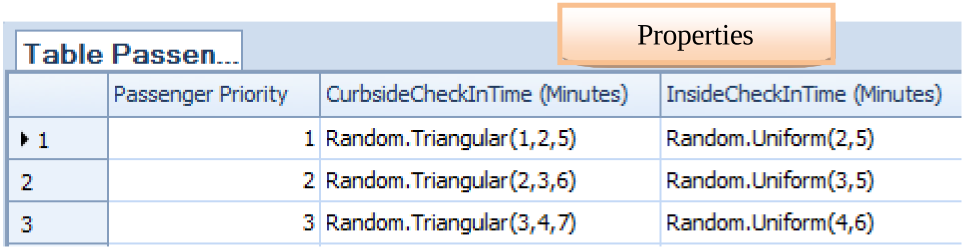
The new value to assign.

Figure 4.9: Assigning Passenger Type

Now each passenger has a (randomly assigned) state variable that describes its passenger type.

Step 5: Next we need to store the information for the check-in times. The expression of these times (e.g. `Random.Unform2,5)`) do not change throughout the simulation. And these check-in times need to be accessed for all entities. The check-in times are not specific to each entity, but are characteristics of the model. So, "properties" are an appropriate way to store these data for the model.

Step 6: A way to organize model properties is through the use of a DATA TABLE. The data table is functionally identical to the table we created for Figure 4.8. To create a DATA TABLE click *"Tables*" in the *"Data"* tab, then *"Add Data Table."* Call it **TablePassenger**. Characteristics, called "properties", can be added to the data tables – these properties are the columns in the table. In our DATA TABLE, the processing times for each of the three types of passengers for each check-in location will be specified. Select the properties from the *"Standard Property"* option in the *Properties* section. **PassengerPriority** will be an *Integer* property whereas the check-in times will be *Expression* properties named **CurbsideCheckInTime** and **InsideCheckInTime**.[32] Priorities 1, 2, and 3 represent Continental, Inter-Continental, and special needs passengers respectively as in Figure 4.10.[33]



	Passenger Priority	CurbsideCheckInTime (Minutes)	InsideCheckInTime (Minutes)
▸ 1	1	Random.Triangular(1,2,5)	Random.Uniform(2,5)
2	2	Random.Triangular(2,3,6)	Random.Uniform(3,5)
3	3	Random.Triangular(3,4,7)	Random.Uniform(4,6)

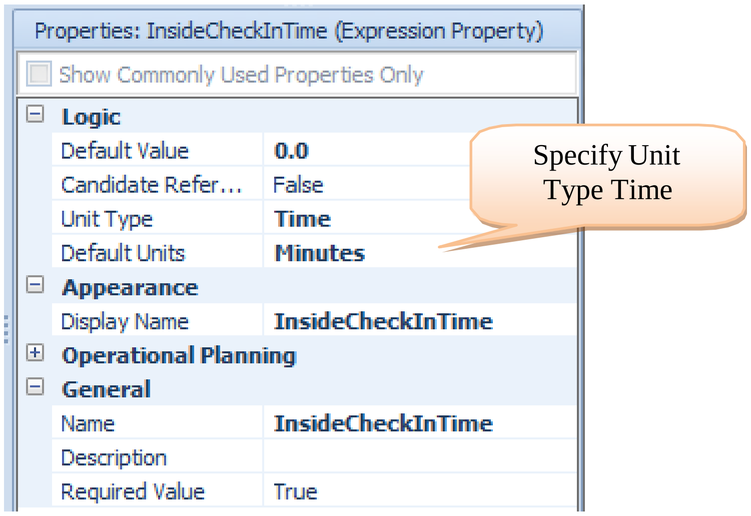
Figure 4.10: The Data Table

Step 7: Since the check-in times are in minutes, we want to specify the "Unit Type" for these properties as seen in Figure 4.11[34].

[32] Note that if you first specify the "*General*" name, it will become the default for the "*DisplayName*" property. Also, by using proper case, the table labels as seen in the Figure 4.10 will have spaces between the words.

[33] Even though the check-in times are expression properties, the familiar drop down expression editor is not available in DATA TABLES. Therefore, the easiest way is to put in all three priorities and type in the first value of the first row. Then copy it to the other rows and just modifying the values of the parameters.

[34] It may seem obvious the Unit Type for the time properties should be specified. However, one can leave the units as unspecified then the numbers will take on the units of the property when they are used (i.e., default value of hours).


Properties: InsideCheckInTime (Expression Property)

Show Commonly Used Properties Only

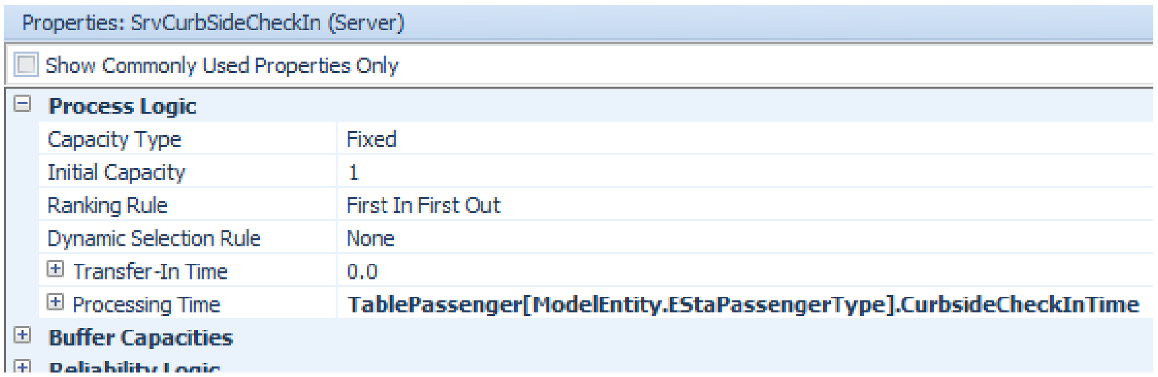
Logic	
Default Value	**0.0**
Candidate Refer...	False
Unit Type	**Time**
Default Units	**Minutes**
Appearance	
Display Name	**InsideCheckInTime**
Operational Planning	
General	
Name	**InsideCheckInTime**
Description	
Required Value	True

Figure 4.11: Inside Check-In Properties

Step 8: Now we must specify the processing times at the check-in stations to depend on the particular entity's passenger type, shown in Figure 4.12. The passenger type is used as the "row reference" in the table. Thus, the specifications needed for the new processing times use the **EStaPassengerType** state variable as:

- *At* **SrvCurbSideCheckIn**:[35] `TablePassenger[ModelEntity.EStaPassengerType].CurbSideCheckInTime` At **SrvCheckIn**: `TablePassenger[`ModelEntity.EStaPassengerType`].InsideCheckInTime`

Notice that the content of the [][36] are used to designate the specific row in the table, similar to arrays in programming languages. Unfortunately, the expression editor does not know about the DATA TABLE properties since they are user-defined, so you must type in the property into the expression. Also be sure that the processing time units in the check-in stations are specified in minutes.


Properties: SrvCurbSideCheckIn (Server)

Show Commonly Used Properties Only

Process Logic	
Capacity Type	Fixed
Initial Capacity	1
Ranking Rule	First In First Out
Dynamic Selection Rule	None
Transfer-In Time	0.0
Processing Time	**TablePassenger[ModelEntity.EStaPassengerType].CurbsideCheckInTime**
Buffer Capacities	
Reliability Logic	

Figure 4.12: Utilizing the Table to Specify the Processing Times

Step 9: Run the model. Remember no arrival occurs before 5am and the simulation is started at 5am so no animation appears before then.

[35] To enter this specification so as to employ the expression editor during entry is to first put in the table name and property as `TablePassenger.CurbSideCheckInTime` and then insert `[ModelEntity.EstaPassengerType]`.

[36] Note, table rows start at one (i.e., the first check in time is `TablePassenger[1].InsideCheckInTime`)

Question 7: What is the average wait times at each check-in location (Curbside and Inside)?

____

Question 8: What is the average check-in time for a passenger?

____

Properties and State Variables: Before leaving this section, let's reflect on the characteristics "properties" and "state variables" because they will be in frequent use. Properties and state variables can be characteristics of the MODEL or of the MODELENTITIES. A "property" of an object is a part of the object's definition and is assigned at the time an object is created, but it can't be changed during the simulation. Properties can be expressions, so their evaluation can yield different values. In contrast a "state variable" is a characteristic of an object that can be changed during the simulation. Generally, when properties and state variables are associated with the model, they are global in the sense their values can be obtained almost anytime. When properties and state variables are associated with entities, then you need to have access to that model entity in order to access the value.

SIMIO makes extensive use of properties and state variables, which you can see by looking at their "definitions" and expanding the (Inherited) Properties/States. Some properties and states are related. For example, the "*InitialPriority*" property of a MODELENTITY becomes the initial value of the "*Priority*" state variable. Other states are given their initial values by properties.

Part 4.4: More on Branching

At times, the waiting lines at the inside Check-in station seem overcrowded. We decided to add a second Inside Check-in station that passengers may use during rush hours. Thus, passengers will be routed to whichever Check-in station has the smallest number in queue. The distance from this second Check-in station to the Security Check is 75 yards as seen *in* Figure 4.13.

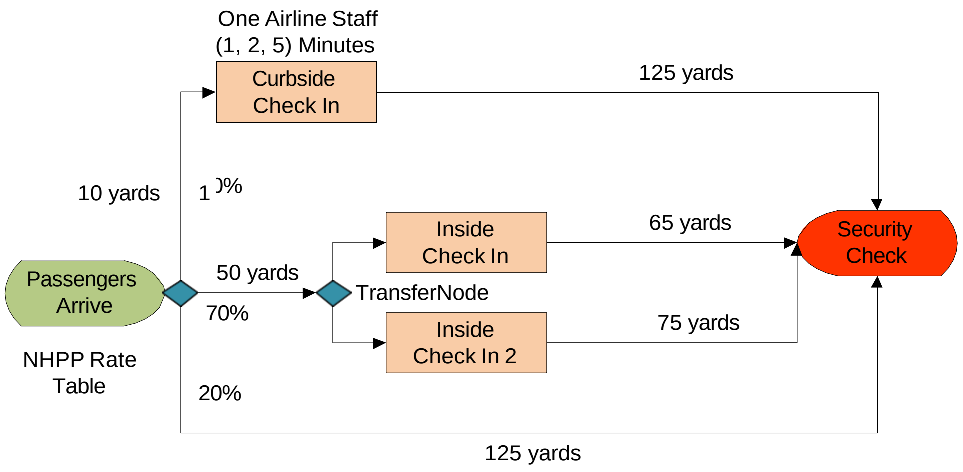


Figure 4.13: Airline Check-in Process with Two Inside Check In Stations

Step 1: Add the second check-in station. Call it **SrvCheckIn2**. Assume it has a capacity of one and it's processing times are the same as those for the regular check-in station. Add the path of 75 yards from this second station to **SnkSecurity**.

Step 2: We need to model the choice made by the "inside" check-in passengers to go to the check-in station with the smallest number in queue. There are several ways to model this situation. Perhaps the most obvious way is to add a TRANSFERNODE that branches passengers to the shortest queue. The distance from the **SrcPassengersArrive** to this TRANSFERNODE is 50 yards (remember that the selection weight is 70 for this path and you will have to delete the original path to the **SrvCheckIn** object).

- The TRANSFERNODE should specify the *Outbound Link Rule* as "By Link Weight"
- Set the *Selection Weight* properties of the two paths from the TRANSFERNODE (the "value" of a "true" expression is one and zero if it is "false")

```
(1) To SrvCheckIn:
   SrvCheckIn.InputBuffer.Contents.NumberWaiting <=
         SrvCheckIn2.InputBuffer.Contents.NumberWaiting

(2) To SrvCheckIn2:
   SrvCheckIn2.InputBuffer.Contents.NumberWaiting <
         SrvCheckIn.InputBuffer.Contents.NumberWaiting
```

Question 9: Which check-in station is chosen if there are "ties" in the size of the InputBuffer (For example, suppose both are empty)?

____

Question 10: What is the utilization of **SrvCheckIn2** and does it make sense to have both check-ins?

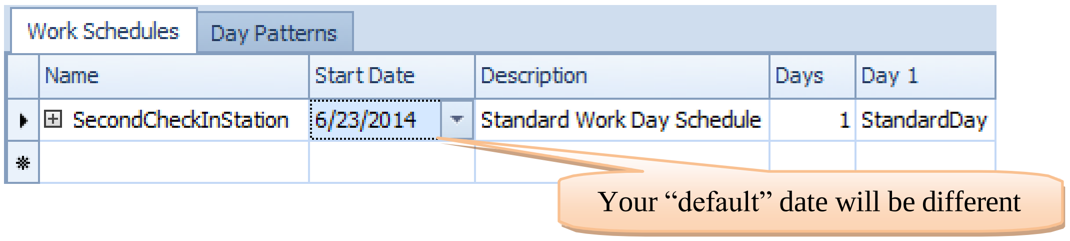
____

Part 4.5: Work Schedules

Suppose you decide only to staff the second inside check-in station from 10am to 6pm. This kind of concern requires a work schedule which alters the capacity of a server over time. Work schedules will be built from the *"Schedules"* component in the *"Data"* tab.

Step 1: Click on *"Work Schedule"* button under in the "Schedules" icon in the "Data" tab to add a new work schedule. By default, the StandardWeek schedule has already been defined as seen in

Step 2: Figure 4.14. Rename this schedule **SecondCheckInStation**. Change the Days to "1" – this schedule will repeat daily.[37] For each day of the work schedule, you need to specify the work pattern for that day. Therefore, you can have different patterns for different days (i.e., weekends, etc.). The **StandardDay** pattern was specified for *Day 1*. The *Start Date* property can be left as the default since we have a daily pattern.



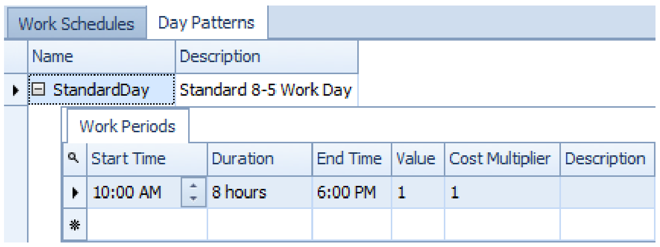
Name	Start Date	Description	Days	Day 1
SecondCheckInStation	6/23/2014	Standard Work Day Schedule	1	StandardDay

Figure 4.14: Creating SecondCheckInStation WORKSCHEDULE

Step 3: In the *"Day Patterns"* tab, you can add DAY PATTERNS by clicking the *Day Pattern* button in the *"Create"* section. SIMIO defines the **StandardDay** pattern of 8am to 5pm with an hour lunch at 12pm. Since we start the day at 10am and end the day at 6pm, we only need one work period in the pattern. Select the second work period (1pm to 5pm) and delete it. Then specify the Start Time to be 10am and the Duration to be eight hours or the *"End Time"* to be 6pm as seen in Figure 4.15. Specify a "1" for the

[37] If there are seven days in the pattern, then the days of the week will appear. In our case there is only the Day 1 pattern. SIMIO will repeat the pattern based on the number of days in the WORK SCHEDULE. For each day in the work schedule you need to specify a pattern. Also, we have the ability to specify exceptions to the day pattern or work periods for particular dates.

Value which will specify the capacity of the second inside check in to be one. If a work period has not been defined for a period in the day pattern, it is assumed to be off shift (i.e., zero capacity).


Work Schedules | Day Patterns

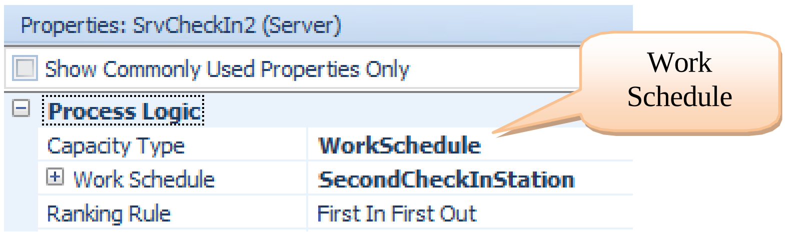
Name	Description
StandardDay	Standard 8-5 Work Day

Work Periods

Start Time	Duration	End Time	Value	Cost Multiplier	Description
10:00 AM	8 hours	6:00 PM	1	1	

Figure 4.15: Setting a Day Pattern with Different Work Periods

Step 4: Go to the "Facility" view and click on the **SrvCheckin2** object. Change the *Capacity Type* to "**WorkSchedule**" and the *Work Schedule* property to **SecondCheckInStation as in** Figure 4.16**.**


Properties: SrvCheckIn2 (Server)

Show Commonly Used Properties Only

Process Logic

Capacity Type	WorkSchedule
Work Schedule	SecondCheckInStation
Ranking Rule	First In First Out

Figure 4.16: Specifying a Work Schedule

Step 5: Now we need to be sure that people who choose the inside check-in only go to the **SrvCheckIn2** station if it is open (i.e., only between hours 10 and 18). This requires another TRANSFERNODE as seen in Figure 4.17. Connect the original TRANSFERNODE to the new one via a Connector and connect the new TRANSFERNODE to the inputs of both check-in stations.

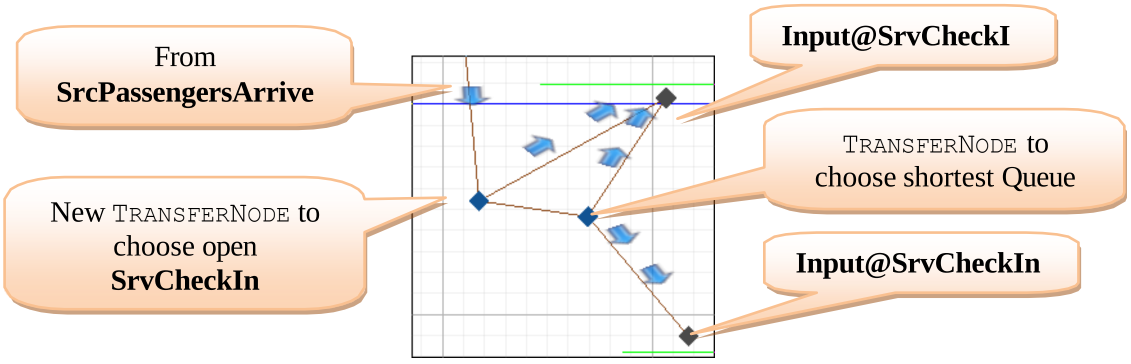

From SrcPassengersArrive
New TRANSFERNODE to choose open SrvCheckIn
Input@SrvCheckI
TRANSFERNODE to choose shortest Queue
Input@SrvCheckIn

Figure 4.17: Adding Another TransferNode to Handle the Logic

- The new TRANSFERNODE should determine routing *By Link Weight* as well.
- Path to **Input@SrvCheckIn** has zero length[38] and *Selection Weight* property: `Run.TimeNow <=5 || Run.TimeNow>=13`[39]
- Path to TRANSFERNODE to choose shortest queue has zero length and *Selection Weight* property of `Run.TimeNow > 5 && Run.TimeNow <13`[40]

[38] May be easier to use a CONNECTOR instead of a PATH – CONNECTORS have zero length and take zero time to traverse.

[39] Note, SIMIO utilizes `||` as the "Or" logical operator and `&&` as the "And" logical operator. Also, this expression works only here since we are simulating just one day and 10 A.M. represents five hours into the simulation since we are starting at 5 A.M. If we were simulating more than one day, then we would need to specify `Math.Remainder(Run.TimeNow, 24)*24` which converts the running time into a number between 0 and 24.

Step 6: Note that when there is no context, the default units of time are hours. Now run the model and obtain these basic statistics.

Question 11: What is the average Check-In time for a Passenger?

____

Question 12: What is the average wait times for Curbside Check-In and Inside Check-In?

____

Question 13: What is the utilization for an employee at the Curbside Check-in?

____

Question 14: What are the utilizations for an employee at the Inside Check-in?

____

Step 7: You may have noticed that in the simulation, there are entities in the queue of the **SrvCheckIn2** that are in line at the time the station closes. These are marooned in this model. We will consider how to handle this kind of problem in a later chapter.

Question 15: What would have to change to the model every time the work schedule of the second check in server was modified?

____

Step 8: If we wanted to determine the best starting time and ending time to have the second check in station open, the link weights using the `Run.TimeNow` expression would have to be changed to match the new start and end times each time. A better approach would be to use the capacity of the server so as it goes to zero we need to close the path and open the path when the capacity goes to one. Change the link weights according and rerun the model comparing the previous results.

- Path to **Input@SrvCheckIn** *Selection Weight* property: `SrvCheckIn2.Capacity == 0`[41]
- Path to TRANSFERNODE to choose shortest queue has zero length and *Selection Weight* property of `SrvCheckIn2.Capacity > 0`

Part 4.6: Commentary

- The specification of the time-varying arrival pattern in SIMIO requires fixed units for the arrival rate, namely arrivals per hour.
- Pay special attention to the SIMIO specification of cumulative probability and its value for the `Random.Discrete` and `Random.Continuous` distributions.
- While work schedules in SIMIO offers considerable flexibility in terms of specifying exceptions etc., they are somewhat complicated to input. One has to be careful in specifying the correct times and duration. Also be careful in specifying the time to begin and end the simulation run.
- In SIMIO, when a server's capacity goes to zero while there is an entity (or entities) being processed, SIMIO chooses, by default, to "Ignore" this so that the items "in process" will be processed while at the same time the server capacity goes to zero. If another behavior is desired, then you will need to implement it using more advanced concepts, which will be described later.

[40] We could have modified the original link weights and avoided having to add the new paths and transfer node. Specify the expression to be (TimeNow <=5 || TimeNow >= 13) || `SrvCheckIn.InputBuffer.Contents.NumberWaiting <= SrvCheckIn2.InputBuffer.Contents.NumberWaiting` for the link weight for the path to **SrvCheckIn** and for the path link weight to **SrvCheckin2** to (Run.TimeNow >5 && Run.TimeNow < 13) && `SrvCheckIn2.InputBuffer.Contents.NumberWaiting < SrvCheckIn.InputBuffer.Contents.NumberWaiting`.

[41] Note, SIMIO utilizes a double equal sign (i.e., "==") as the logical equal.

- Note that if the length of the simulation replication exceeds the work schedule time, then the work schedule is repeated until the replication ends.