Steps to Add Modes:

This involved some changes in the config file and some in the lua files. Here we shall show the addition of two modes Rail_SMS and SMS:

Step 1: Changes in the config file

In the config file: data/simulation.xml, the lines marked in green should be added.

```
<!-- type {1: Public Transit, 2: PrivateBus, 3: Private Car, 4: Sharing, 5: Private bike, 6:
Walk, 7: Taxi}-->
 <travel_modes>

            <mode name="BusTravel" type="1" num_sharing="1"/>
            <mode name="MRT" type="1" num_sharing="1"/>
            <mode name="PrivateBus" type="2" num_sharing="1"/>
            <mode name="Car" type="3" num_sharing="1"/>

            <mode name="Car Sharing 2" type="4" num_sharing="2"/>
            <mode name="Car Sharing 3" type="4" num_sharing="3"/>
            <mode name="Motorcycle" type="5" num_sharing="1"/> <mode
            name="Walk" type="6" num_sharing="1"/>
            <mode name="Taxi" type="7" num_sharing="1"/>
            <mode name="SMS" type="7" num_sharing="1"/>

            <mode name="Rail_SMS" type="1" num_sharing="1"/>

</travel_modes>
```

Important consideration:

The order of the modes must be maintained between the config file and the lua files. For example: in the above example of `data/simrunMidTerm.xml`, the “BusTravel” is the first in the list. Hence the mode number 1 will refer to “BusTravel”, both in lua and C++.

Hence , in the lua files (say tmdo.lua, tmds.lua etc)

`utility[1]` will represent the marginal utility for “BusTravel”, `utility[2]` will represent the marginal utility for “MRT” and so on

---

Step- 2: Changes in lua files:

imd.lua:

i) loop for choices: must be equal to <number of destinations(taz)> * <number of modes>

The changes are marked in green.

```
for i = 1, 1169*11 do

        choice[i] = i
end
```

The same applies to the scale:

```
for i = 1, 1169*11 do
        scale[i]=1

end
```

ii) In the function compute _utilities, statements for the Rail_SMS and SMS modes should be added. These are marked in green:

```
.
.
.
.
.

 --utility function for Taxi 1-1169

for i=1,1169 do
       V_counter = V_counter +1
       utility[V_counter] = beta_cons_Taxi + first_bound .................
end

 --utility function for SMS 1-1169

for i=1,1169 do
       V_counter = V_counter +1
       utility[V_counter] = beta_cons_SMS + first_bound .................
end

 --utility function for rail_SMS 1-1169
 for i=1,1169 do
        V_counter = V_counter +1
        utility[V_counter] = beta_cons_rail_SMS + cost_rail_SMS[i] * .......
 end
```

iii) In the function computeAvailabilities(), the range of the loop must be set correctly. 1 to <number of destinations(taz)> * <number of modes>. The change is marked in green.

```
local function computeAvailabilities(params,dbparams)
for i = 1, 1169*11 do

      availability[i] = dbparams:availability(i)
end
```

iv) The same applies to the scale:

```
for i = 1, 1169* 11 do
        scale[i]=1

 end
```

==========

= *stmd.lua, tmdo.lua, tmds.lua, tmdw.lua* :

Similar changes as shown above:

i) Range of for loop in choice vector.

ii) Defining new utilities and maintaining the order

iii) Range of for loop in computeAvailabilities

iv) Range of for loop in scale vector (if present)

==========

= *tmw.lua:*

i) The choice vector should be defined correctly. The number of choices should be the same as the number of modes. Hence while adding the two modes Rail_SMS and SMS, we need to add the choices shown in green below:

```
        local choice = {
```

```
1,
2,

3,
4,

5,
6,

7,
8,

9,
10,

11}
```

ii) A dictionary(map) should be created as shown below. The terms for the modes Rail_SMS and SMS, should be added, as shown in green.

```
local modes = {['BusTravel'] = 1 , ['MRT'] =2 , ['PrivateBus']
=3 , ['Car'] = 4, ['Car_Sharing_2'] = 5,['Car_Sharing_3'] = 6,
['Motorcycle'] = 7,['Walk'] = 8, ['Taxi'] = 9 , ['SMS'] = 10,
['Rail_SMS'] = 11 }
```

iii) The utilitiy expressions for modes Rail_SMS and SMS must be added in the compute_utilities function, as shown in green below.

```
.
.
.
.
.

utility[9] = beta_cons_Taxi + first_bound .................
utility[10] = beta_cons_SMS + first_bound .................

utility[11] = beta_cons_rail_SMS + cost_rail_SMS[i] * .......
```

iv) In the computeAvailabilities() function, the availabilities for the modes Rail_SMS and SMS should be added.

```
dbparams:getModeAvailability(modes.Car_Sharing_2),
dbparams:getModeAvailability(modes.Car_Sharing_3),
dbparams:getModeAvailability(modes.Motorcycle),
dbparams:getModeAvailability(modes.Walk),
dbparams:getModeAvailability(modes.Taxi),
dbparams:getModeAvailability(modes.SMS),
dbparams:getModeAvailability(modes.Rail_SMS)
```

==========

tme.lua:

i) The choice vector should be defined correctly. The number of choices

should be the same as the number of modes. Hence while removing the two modes Rail_SMS and SMS, we need to add the choices shown in green below:

```
local choice = {

            1,
            2,

            3,
            4,

            5,
            6,

            7,
            8,

            9,
            10,
```

```
                    11}
```

ii) A dictionary(map) should be created as shown below. The terms for the modes Rail_SMS and SMS, should be added, as shown in green.

```
 local modes = {['BusTravel'] = 1 , ['MRT'] =2 , ['PrivateBus']
=3 , ['Car'] = 4, ['Car_Sharing_2'] = 5,['Car_Sharing_3'] = 6,
['Motorcycle'] = 7,['Walk'] = 8, ['Taxi'] = 9 , ['SMS'] = 10,
['Rail_SMS'] = 11 }
```

iii) The utilitiy expressions for modes Rail_SMS and SMS must be added in the compute_utilities function, as shown in green below.

```
.
.
.
.
.

utility[9] = beta_cons_Taxi + first_bound ..................

utility[10] = beta_cons_SMS + first_bound ..................
utility[11] = beta_cons_rail_SMS + cost_rail_SMS[i] * .......
```

iv) In the computeAvailabilities() function, the availabilities for the modes Rail_SMS and SMS should be added as shown in green below:

```
dbparams:getModeAvailability(modes.Car_Sharing_2),
dbparams:getModeAvailability(modes.Car_Sharing_3),
dbparams:getModeAvailability(modes.Motorcycle),
dbparams:getModeAvailability(modes.Walk),
dbparams:getModeAvailability(modes.Taxi),
dbparams:getModeAvailability(modes.SMS),
dbparams:getModeAvailability(modes.Rail_SMS)
```

(v) Each of the modes should be assigned the “mode type” in a map. Hence while removing the two modes Rail_SMS and SMS, we need to add the choices shown in green below:

```
choice["PT"] = {1,2,3,11}

choice["car"] = {4,5,6,9,10}

choice["other"] = {8,9}
```