

# How to Add KPIs to the Simulator

Step 1: Determine what Data-type your KPI has to be

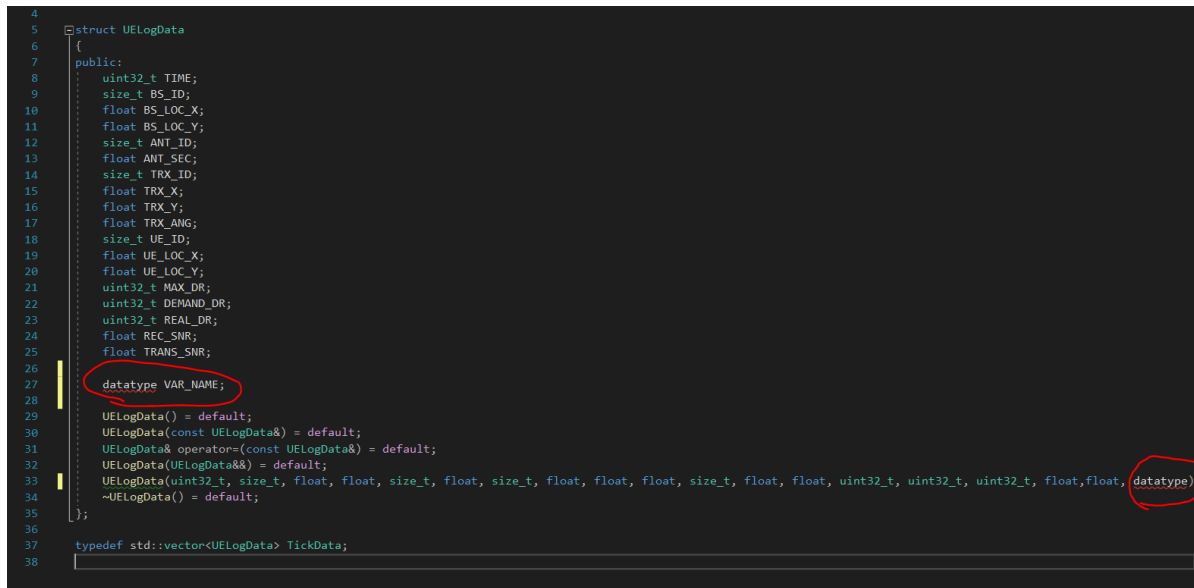
most likely it will either be an uint32\_t or a float

Step 2: Add your variable to UELogData.h and UELogData.cpp

In UELogData.h:

Add it onto the list of public variables

Also add it as a parameter in the function declaration for UELogData::UELogData()

A screenshot of a code editor showing the UELogData.h header file. The code defines a struct UELogData with various public variables. Two instances of the variable 'datatype VAR\_NAME;' are circled in red: one in the public variable list (line 27) and one in the constructor parameter list (line 33).

```
4
5 struct UELogData
6 {
7     public:
8         uint32_t TIME;
9         size_t BS_ID;
10        float BS_LOC_X;
11        float BS_LOC_Y;
12        size_t ANT_ID;
13        float ANT_SEC;
14        size_t TRX_ID;
15        float TRX_X;
16        float TRX_Y;
17        float TRX_ANG;
18        size_t UE_ID;
19        float UE_LOC_X;
20        float UE_LOC_Y;
21        uint32_t MAX_DR;
22        uint32_t DEMAND_DR;
23        uint32_t REAL_DR;
24        float REC_SNR;
25        float TRANS_SNR;
26
27        datatype VAR_NAME;
28
29        UELogData() = default;
30        UELogData(const UELogData&) = default;
31        UELogData& operator=(const UELogData&) = default;
32        UELogData(UELogData&&) = default;
33        UELogData(uint32_t, size_t, float, float, size_t, float, float, float, float, size_t, float, float, uint32_t, uint32_t, uint32_t, float, float, datatype);
34        ~UELogData() = default;
35    };
36
37    typedef std::vector<UELogData> TickData;
38
```

In UELogData.cpp

Add it to the function parameters of UELogData::UELogData()

Add it into the function by writing this line at the bottom of the function:

this->VAR\_NAME = var\_name;

```

1  #include "UELogData.h"
2
3  UELogData::UELogData(uint32_t time, size_t bs_id, float bs_loc_x, float bs_loc_y, size_t ant_id, float ant_sec, size_t trx_id, float trx_x, float trx_y,
4  float trx_ang, size_t ue_id, float ue_loc_x, float ue_loc_y, uint32_t max_dr, uint32_t demand_dr, uint32_t real_dr, float rec_snr, float trans_snr,
5  datatype var_name)
6  {
7      this->TIME = time;
8      this->BS_LOC_X = bs_loc_x;
9      this->BS_LOC_Y = bs_loc_y;
10     this->BS_ID = bs_id;
11     this->ANT_ID = ant_id;
12     this->ANT_SEC = ant_sec;
13     this->TRX_ID = trx_id;
14     this->TRX_X = trx_x;
15     this->TRX_Y = trx_y;
16     this->TRX_ANG = trx_ang;
17     this->UE_ID = ue_id;
18     this->UE_LOC_X = ue_loc_x;
19     this->UE_LOC_Y = ue_loc_y;
20     this->MAX_DR = max_dr;
21     this->DEMAND_DR = demand_dr;
22     this->REAL_DR = real_dr;
23     this->REC_SNR = rec_snr;
24     this->TRANS_SNR = trans_snr;
25
26     this->VAR_NAME = var_name;
27 }
28

```

Note: replace VAR\_NAME with the actual name you want to use for the variable

### Step 3: Add your variable to UERecord.h and UERecord.cpp

In UERecord.h:

Add the variable to the list of declared variables

Also add it as a parameter in the function declaration for UERecord::UERecord()

```

1  #pragma once
2  #include <stdint>
3  #include <vector>
4  #include "Coord.h"
5  #include <utility>
6
7  struct UERecord
8  {
9      size_t userID, antenna, currentTransceiver;
10     uint32_t demand, bitsSent;
11     Coord<float> loc;
12     float currentSNR, powerSent;
13
14     datatype var_name;
15
16     UERecord() = delete;
17     UERecord(const size_t& uid, const Coord<float>& loc, const size_t& at, const size_t& ct, const float& cSNR, const uint32_t& d,
18             const uint32_t& bts, const float& ps, const datatype v_n);
19     UERecord(const UERecord&) = default;
20     UERecord(UERecord&&) noexcept = default;
21     UERecord& operator=(const UERecord&) = default;
22 };
23

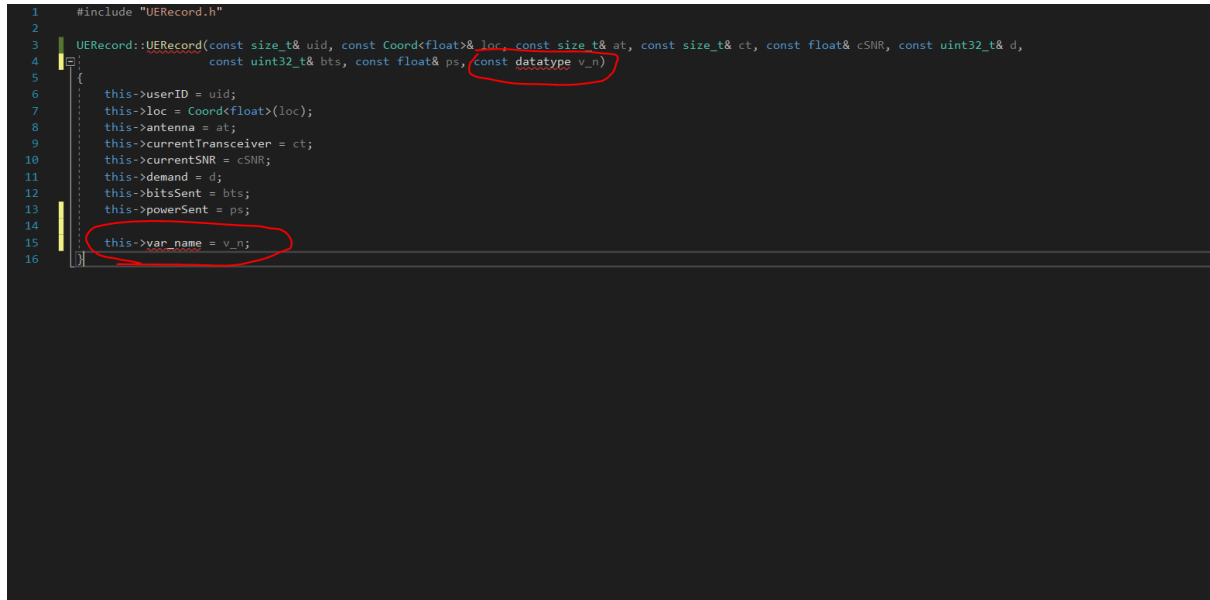
```

In UERecord.cpp

add it to the function parameters and

add it into the function by writing this line at the bottom of the function:

`this->VAR_NAME = var_name;`



```
1  #include "UERecord.h"
2
3  UERecord::UERecord(const size_t& uid, const Coord<float>& loc, const size_t& at, const size_t& ct, const float& cSNR, const uint32_t& d,
4  const uint32_t& bts, const float& ps, const datatype v_n)
5  {
6      this->userID = uid;
7      this->loc = Coord<float>(loc);
8      this->antenna = at;
9      this->currentTransceiver = ct;
10     this->currentSNR = cSNR;
11     this->demand = d;
12     this->bitsSent = bts;
13     this->powerSent = ps;
14
15     this->var_name = v_n;
16 }
```

Note: Replace VAR\_NAME with the actual name you want to use for the variable just like Step 2

## Step 4: add this variable to IRPManager.cpp

This function collects the data from all the User Equipment Records every tick so you need to add your KPI to the list in order to get recorded.

the data type should be `(*uer).var_name` as the variable comes from UERecords

```

106 for (const auto& ueTrPair : ant.getConnectionInfo().getUserTransPairings())
107 {
108     const auto& tr = ant.getConnectionInfo().getTransceivers()[ueTrPair.second];
109     const auto& usr = Simulator::getUE(ueTrPair.first);
110     const auto uer = bs.getUEDB().look_up(ueTrPair.first);
111     if (!uer)
112     {
113         ErrorTracer::error("IRPManager could not look up User Equipment but expected it to be in UEDB in IRPManager::IRPDataCollection()");
114     }
115     else
116     {
117         const auto logData = UELogData(
118             Simulator::getEnvClock(),
119             bs.getBSID(),
120             bs.getLoc().x,
121             bs.getLoc().y,
122             ant.getAntID(),
123             ant.getAngle(),
124             ueTrPair.second,
125             tr.getLoc().x,
126             tr.getLoc().y,
127             tr.getTheta(),
128             ueTrPair.first,
129             usr.getLoc().x,
130             usr.getLoc().y,
131             usr.getMaxDr(),
132             usr.getDemand(),
133             usr.getRecDR(),
134             (*uer).powerSent,
135             usr.getRecPwr(),
136             (*uer).var_name()
137         );
138         td.push_back(logData);
139     }
140 }
141
142

```

## Step 5: Adding this variable to FileIO.cpp

In FileIO::writeInitialSimulationState()

There is a for loop that goes through every User Equipment Record and writes their variables into a file object.

Add your variable into this for loop using the following 2 lines:

```
const auto& v_n = (*uer).var_name;
```

```
file_obj.write(FileIO::chPtrConv(&v_n), sizeof(v_n));
```

```

308 const auto& db = bs.getUEDB().readDB();
309 const auto dbSize = db.size();
310 file_obj.write(FileIO::chPtrConv(&dbSize), sizeof(dbSize));
311 for (const auto& uer : db)
312 {
313     const auto& userID = (*uer).userID;
314     file_obj.write(FileIO::chPtrConv(&userID), sizeof(userID));
315
316     const auto& x = (*uer).loc.x;
317     file_obj.write(FileIO::chPtrConv(&x), sizeof(x));
318
319     const auto& y = (*uer).loc.y;
320     file_obj.write(FileIO::chPtrConv(&y), sizeof(y));
321
322     const auto& ant = (*uer).antenna;
323     file_obj.write(FileIO::chPtrConv(&ant), sizeof(ant));
324
325     const auto& tr = (*uer).currentTransceiver;
326     file_obj.write(FileIO::chPtrConv(&tr), sizeof(tr));
327
328     const auto& snr = (*uer).currentSNR;
329     file_obj.write(FileIO::chPtrConv(&snr), sizeof(snr));
330
331     const auto& demand = (*uer).demand;
332     file_obj.write(FileIO::chPtrConv(&demand), sizeof(demand));
333
334     const auto& bts = (*uer).bitsSent;
335     file_obj.write(FileIO::chPtrConv(&bts), sizeof(bts));
336
337     const auto& ps = (*uer).powerSent;
338     file_obj.write(FileIO::chPtrConv(&ps), sizeof(ps));
339
340     const auto& v_n = (*uer).var_name;
341     file_obj.write(FileIO::chPtrConv(&v_n), sizeof(v_n));
342 }
343
344

```

Note: Replace v\_n and var\_name with the actual name you want to use for your variable

### In FileIO::readSaveFileIntoSim()

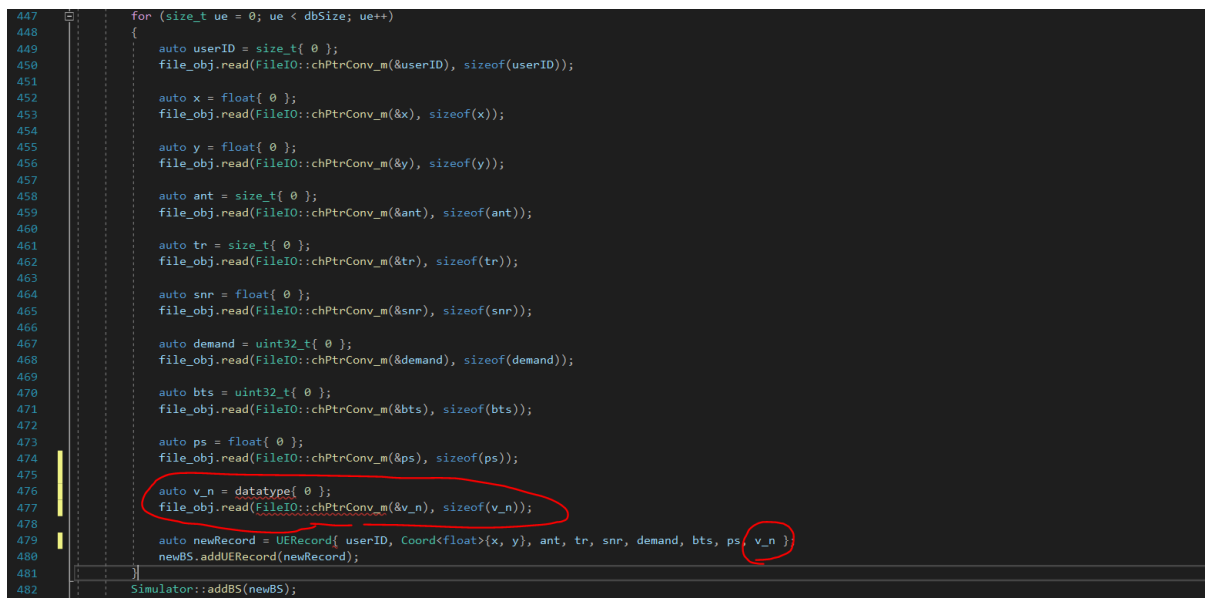
There is a for loop that reads the values from each User Equipment connected to a base station

Add your variable to this for loop using the following 2 lines:

```
auto v_n = datatype{ 0 };
```

```
file_obj.read(FileIO::chPtrConv_m(&v_n), sizeof(v_n));
```

Also add your variable to the auto newRecord = UERecord{ } function call



```
447 for (size_t ue = 0; ue < dbSize; ue++)
448 {
449     auto userID = size_t{ 0 };
450     file_obj.read(FileIO::chPtrConv_m(&userID), sizeof(userID));
451
452     auto x = float{ 0 };
453     file_obj.read(FileIO::chPtrConv_m(&x), sizeof(x));
454
455     auto y = float{ 0 };
456     file_obj.read(FileIO::chPtrConv_m(&y), sizeof(y));
457
458     auto ant = size_t{ 0 };
459     file_obj.read(FileIO::chPtrConv_m(&ant), sizeof(ant));
460
461     auto tr = size_t{ 0 };
462     file_obj.read(FileIO::chPtrConv_m(&tr), sizeof(tr));
463
464     auto snr = float{ 0 };
465     file_obj.read(FileIO::chPtrConv_m(&snr), sizeof(snr));
466
467     auto demand = uint32_t{ 0 };
468     file_obj.read(FileIO::chPtrConv_m(&demand), sizeof(demand));
469
470     auto bts = uint32_t{ 0 };
471     file_obj.read(FileIO::chPtrConv_m(&bts), sizeof(bts));
472
473     auto ps = float{ 0 };
474     file_obj.read(FileIO::chPtrConv_m(&ps), sizeof(ps));
475
476     auto v_n = datatype{ 0 };
477     file_obj.read(FileIO::chPtrConv_m(&v_n), sizeof(v_n));
478
479     auto newRecord = UERecord{ userID, Coord<float>(x, y), ant, tr, snr, demand, bts, ps, v_n };
480     newBS.addUERecord(newRecord);
481 }
482 Simulator::addBS(newBS);
```

Note: Replace v\_n with the actual name you want to use for your variable and replace datatype with whatever datatype your variable is: most likely float or uint32\_t

### In FileIO::appendLog

Add your variable to the log here separated by a comma:

```

492 ::string newFilePath;
493 (FileIO::splitLogFiles)
494
495 if (FileIO::logRowCount >= FileIO::AP_MaxLogLines)
496 {
497     FileIO::incrementLogCount();
498     FileIO::logRowCount = 0;
499 }
500 newFilePath = filePath + "_SIM_" + std::to_string(simNum) + "_LOG_" + std::to_string(FileIO::logCount) + ".csv";
501
502 newFilePath = filePath + "_SIM_" + std::to_string(simNum) + ".csv";
503
504 log = std::ofstream( newFilePath, std::ios::app );
505
506 (FileIO::logRowCount == 0)
507
508 log << "Time,BS_ID,BS_LOC_X,BS_LOC_Y,ANT_ID,ANT_SEC,TRX_ID,TRX_X,TRX_Y,TRX_ANG,UE_ID,UE_LOC_X,UE_LOC_Y,MAX_DR,DEMAND_DR,REAL_DR,BS_Trans_SNR,UE_Rec_SNR,VAR_NAME\n";
509 FileIO::incrementLogCount();
510
511 (Simulator::getIRPManager().getBuffer().size() < 1)
512 return ErrorTracer::error("Network Manager buffer empty when attempting to access log.");
513
514 (const auto& ue : Simulator::getIRPManager().getBuffer().getLastTick())
515
516 log << ue.TIME << ','
517 << ue.BS_ID << ','
518 << ue.BS_LOC_X << ','
519 << ue.BS_LOC_Y << ','
520 << ue.ANT_ID << ','
521 << ue.ANT_SEC << ','
522 << ue.TRX_ID << ','
523 << ue.TRX_X << ','
524 << ue.TRX_Y << ','
525 << ue.TRX_ANG << ','
526 << ue.UE_ID << ','
527 << ue.UE_LOC_X << ','
528 << ue.UE_LOC_Y << ','
529 << ue.MAX_DR << ','
530 << ue.DEMAND_DR << ','
531 << ue.REAL_DR << ','
532 << ue.TRANS_SNR << ','
533 << ue.REC_SNR << ','
534 << ue.VAR_NAME << '\n';
535
536 if (FileIO::splitLogFiles)
537 {
538     FileIO::incrementLogCount();
539 }
540
541 log.close();

```

Note: Whatever you name the variable here will be how it appears on the csv file

Also add your variable to the for loop that adds each new line of data by adding the following line:

```
<< ue.VAR_NAME << '\n';
```

```

510 FileIO::incrementLogCount();
511 }
512
513 if (Simulator::getIRPManager().getBuffer().size() < 1)
514     return ErrorTracer::error("Network Manager buffer empty when attempting to access log.");
515
516 for (const auto& ue : Simulator::getIRPManager().getBuffer().getLastTick())
517 {
518     log << ue.TIME << ','
519     << ue.BS_ID << ','
520     << ue.BS_LOC_X << ','
521     << ue.BS_LOC_Y << ','
522     << ue.ANT_ID << ','
523     << ue.ANT_SEC << ','
524     << ue.TRX_ID << ','
525     << ue.TRX_X << ','
526     << ue.TRX_Y << ','
527     << ue.TRX_ANG << ','
528     << ue.UE_ID << ','
529     << ue.UE_LOC_X << ','
530     << ue.UE_LOC_Y << ','
531     << ue.MAX_DR << ','
532     << ue.DEMAND_DR << ','
533     << ue.REAL_DR << ','
534     << ue.TRANS_SNR << ','
535     << ue.REC_SNR << ','
536     << ue.VAR_NAME << '\n';
537
538 if (FileIO::splitLogFiles)
539 {
540     FileIO::incrementLogCount();
541 }
542
543 }
544
545 log.close();

```

Note: You cannot copy-paste this line as ' are treated differently by Visual Studio

## Step 6: add the variable to EnvironmentalInitialization.cpp

In EnvironmentInitialization::setDefaultUsers()

This function initializes the environment so you need to add additional 0s, or a specific initial value if you need one, to this newRecord for each of the variables you are adding

```

25     const auto rPhase = float{ 2.0f * (Simulator::randF() - 0.5f) * Simulator::PI / Simulator::getNumberOfAntennae() + ant.getAngle() * Simulator::PI /
26     const auto loc = Coord<float>{ static_cast<float>(rRadius * cos(rPhase)), static_cast<float>(rRadius * sin(rPhase)) };
27
28     const auto distanceSquared = float{ static_cast<float>(pow(loc.x, static_cast<int>(2)) + pow(loc.y, static_cast<int>(2))) };
29     const auto SNR = Simulator::generateSNR(distanceSquared);
30     const auto dataRate = DataRateTable::getDataRate(SNR, Simulator::getCurrentChannel());
31
32     //next user to be added will have the current # of users. E.G. if there are 0 UEs then the first ID = 0.
33     const auto currUserID = Simulator::getNumOfUsers();
34
35     //tranceiver set to the UE
36     const auto currentTranceiver = bs.getAntenna(ant.getAntID()).getConnectionInfo_m().addUser(currUserID);
37     if (!currentTranceiver.first)
38         continue;
39
40     const auto currentDemand = uint32_t{ (Simulator::rand() % (dataRate + 1)) };
41
42     const auto newRecord = UERecord{
43         currUserID,
44         Coord<float>{loc.x + bs.getLoc().x, loc.y + bs.getLoc().y},
45         ant.getAntID(),
46         currentTranceiver.second,
47         SNR,
48         currentDemand,
49         0,
50         0,
51         initialvalue
52     };
53     bs.addUERecord(newRecord);
54
55     const auto& numChan = Simulator::getNumOfChannels();
56     auto ch = size_t{ 0 };
57     std::vector<uint32_t> possMaxDrsForUE = std::vector<uint32_t>(numChan, 0);
58     for (auto& dr : possMaxDrsForUE)
59     {
60         for (auto& dr : possMaxDrsForUE)
61     }

```

## Step 7: add the variable to EnvironmentController.cpp

In EnvironmentController::addUsers()

Similar to Environment Initialization this function creates a default UERecord so you will need to add additional 0s, or initial values, for each of variables you are adding

```

220
221 //gets the randomly selected point
222 const auto loc = Coord<float>{ static_cast<float>(rRadius * cos(rPhase)), static_cast<float>(rRadius * sin(rPhase)) };
223
224 const auto distanceSquared = float{ static_cast<float>(pow(loc.x, static_cast<int>(2)) + pow(loc.y, static_cast<int>(2))) };
225 const auto SNR = float{ Simulator::generateSNR(distanceSquared) };
226 auto dataRate = uint32_t{ DataRateTable::getDataRate(SNR, Simulator::getCurrentChannel()) };
227
228 //next user to be added will have the current # of users. E.G. if there are 0 UEs then the first ID = 0.
229 const auto currUserID = size_t{ Simulator::getNumOfUsers() };
230
231 //tranceiver set to the UE
232 const auto currentTranceiver = Simulator::getBS_m(bsID).getAntenna(antID).getConnectionInfo_m().addUser(currUserID);
233 if (!currentTranceiver.first)
234     continue;
235
236 auto currentDemand = uint32_t{ 0 };
237 if (bsfp.endStatus == BSStatus::congestionDemand)
238     currentDemand = dataRate;
239 else
240     currentDemand = Simulator::rand() % dataRate;
241
242 const auto newRecord = UERecord{ currUserID, Coord<float>{ loc.x + bs.getLoc().x, loc.y + bs.getLoc().y }, antID, currentTranceiver.second,
243     SNR, currentDemand, 0, 0, initialvalue };
244 Simulator::getBS_m(bsID).addUERecord(newRecord);
245
246 const auto& numChan = Simulator::getNumOfChannels();
247 auto ch = size_t{ 0 };
248 auto possMaxDrsForUE = std::vector<uint32_t>( numChan, 0 );
249 for (auto& dr : possMaxDrsForUE)
250 {
251     dr = DataRateTable::getDataRate(SNR, ch);
252     if (dr == 0)
253         ErrorTracer::error("\nINVALID DATARATE IN possMaxDrsForUE, POSSIBLE DRTBL ISSUE in EnvironmentController::addUsers(BSFailureParams& bsfp, const int&
254 }
255
256 const auto userLoc = Coord<float>{ loc.x + bs.getLoc().x, loc.y + bs.getLoc().y };

```

## Step 8: add the variable to BaseStation.cpp

In BaseStation::Update()

Add a failure condition for each variable by adding this line to the first for loop

`(*uer).var_name = 0;`

```
154     return this->BSAntennae[0];
155 }
156
157 //FLAG --needs proper failure containment
158 const Antenna& BaseStation::getAntenna(const size_t& ant) const
159 {
160     if (ant < 0 || ant >= this->BSAntennae.size())
161         return this->BSAntennae[0];
162     else
163         return this->BSAntennae[ant];
164 }
165
166 const std::vector<Antenna>& BaseStation::getAntennaVec() const
167 {
168     return this->BSAntennae;
169 }
170
171 bool BaseStation::Update()
172 {
173     this->dataRate = 0;
174
175     if (this->failed)
176     {
177         for (auto& uer : this->userRecords.readWriteDB())
178         {
179             (*uer).bitsSent = 0;
180             (*uer).powerSent = 0;
181             (*uer).var_name = 0;
182         }
183     }
184     else
185     {
186         //unsigned seed = (unsigned)std::chrono::system_clock::now().time_since_epoch().count();
187         //std::default_random_engine e(seed);
188         this->userRecords.shuffle();
189     }
190 }
```

Then add the initial condition to the following for loop by adding the following line

`(*uer).var_name = 0;`

```
184     }
185 }
186 else
187 {
188     //unsigned seed = (unsigned)std::chrono::system_clock::now().time_since_epoch().count();
189     //std::default_random_engine e(seed);
190     this->userRecords.shuffle();
191
192     // iterates through each all the UE in a BaseStation
193     // (BaseStations are iterated through in the main class)
194     for (auto& uer : this->userRecords.readWriteDB())
195     {
196         (*uer).bitsSent = 0;
197         (*uer).powerSent = 0;
198         (*uer).var_name = 0;
199     }
200
201     (*uer).demand = Simulator::getUE((*uer).userID).getDemand();
202     this->outgoingTransmissions.push_back(Transmission{ this->bsID, (*uer).userID, (*uer).antenna, (*uer).currentTransceiver, (*uer).demand });
203 }
204
205 while(!outgoingTransmissions.empty() && this->dataRate + (*outgoingTransmissions.begin()).data < Simulator::getBSMaxDR()) // add the packet to the outgoing
206 {
207     const auto& transmission = *outgoingTransmissions.begin();
208
209     const auto& userID = transmission.destination;
210     auto UER = this->userRecords.look_up_m(userID);
211     if (UER)
212     {
213         this->dataRate += transmission.data;
214         (*UER).bitsSent = transmission.data;
215         const auto& powerTransmitted = this->calculateTransmittedPower(Simulator::AP_SimulationBandwidth, (*UER).currentSNR);
216         (*UER).powerSent = powerTransmitted;
217
218         Simulator::sendUETransmission(userID, transmission.data, powerTransmitted);
219     }
220 }
```



Finally you need to add the code that calculates the actual value for the KPI

This should be done in the if statement at the end of BaseStation::Update by adding the following line:

```
(*UER).var_name = equation ;
```

```
196     (*uer).bitsSent = 0;
197     (*uer).powerSent = 0;
198
199     (*uer).var_name = 0;
200
201     (*uer).demand = Simulator::getUE((*uer).userID).getDemand();
202     this->outgoingTransmissions.push_back(Transmission( this->bsID, (*uer).userID, (*uer).antenna, (*uer).currentTransceiver, (*uer).demand ));
203
204
205     while(!outgoingTransmissions.empty() && this->dataRate + (*outgoingTransmissions.begin()).data < Simulator::getBSMaxDR()) // add the packet to the outgoing
206     {
207         const auto& transmission = *outgoingTransmissions.begin();
208
209         const auto& userID = transmission.destination;
210         auto UER = this->userRecords.look_up_m(userID);
211         if (UER)
212         {
213             this->dataRate += transmission.data;
214             (*UER).bitsSent = transmission.data;
215             const auto& powerTransmitted = this->calculateTransmittedPower(Simulator::AP_SimulationBandwidth, (*UER).currentSNR);
216             (*UER).powerSent = powerTransmitted;
217             (*UER).var_name = equation;
218             Simulator::sendUETransmission(userID, transmission.data, powerTransmitted);
219         }
220     }
221     this->outgoingTransmissions.erase(outgoingTransmissions.begin());
222
223
224
225
226
227
228     return true;
229
230
```

Now you are done

If you run the simulator now you should see a new column of values on the .csv file for your new KPI