Interface Requirements between eSON System and the OSS

Table of Contents

1 Introduction

The information provided in this document outlines the interface requirements between the eSON System and the RMN OSS.

2 Change History

Date	Author	Version	Change
13 Mar 2019	D. Chang	1.0	Initial version.
21 Mar 2019	D. Chang	1.1	Added additional PCI configurable switches; added requirements for PCI counters.
26 Apr 2019	D. Chang	1.2	Remove requirement to enable/disable DMRS collision detection and resolution. Add events for PCI Add counter and events for MCIM and RACh Optimization
5 Jun 2019	D. Chang	1.3	Add MLB enable/disable and configurable parameters

3 Acronyms and Abbreviations

COE	Cell Optimization Engine
CCS	COE Control Server
ECL	eSON Client

4 Interfaces between eSON System and the OSS

As shown in Figure 1, there are two interfaces between the eSON System and the OSS

1. Command and Control Interface – Allows the OSS to enable, disable, and configure algorithm parameters on the eSON System.
2. PM Counters and Events Interface – Interface to allow performance KPIs and events to be reported to the OSS.

The requirements for these interfaces will be discussed in the following sections.

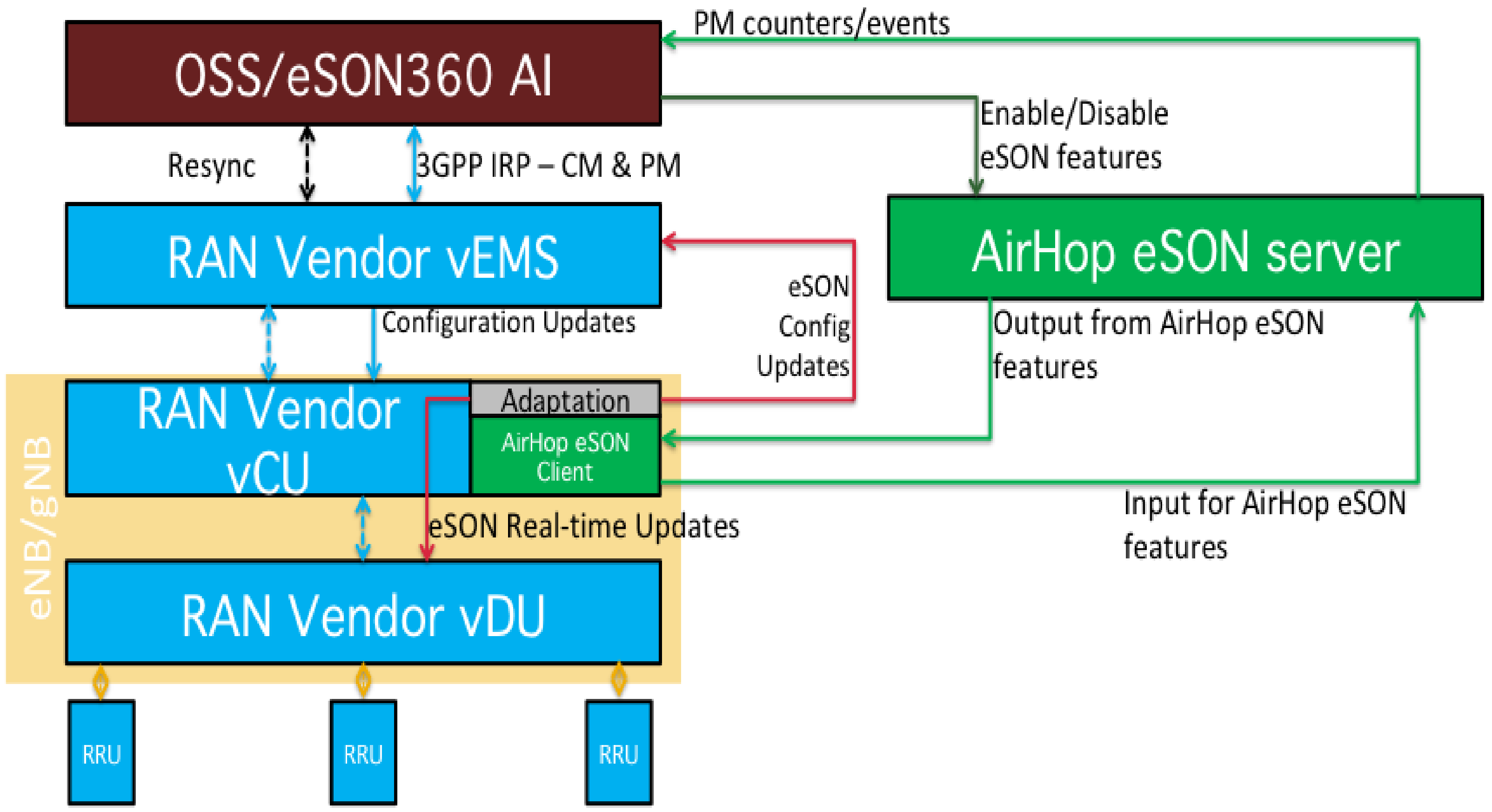

Figure 1 – Airhop eSON System architecture

4.1 Command and Control Interface

4.1.1 eSON Status

The following are requirements are to manage the eSON status from the OSS platform.

- The OSS shall provide a GUI to indicate the current status of the eSON System.
- The OSS shall use the ERA restful API

  GET
  [IP Addr:Port]/status

  to query the current status of the eSON System. The expected response to the query is the current time and eSON System start time. If these times are received the eSON System is online and ready.

4.1.2 PCI Algorithm Configuration

The following are requirements to manage the PCI Optimization algorithm configuration from the OSS platform.

- The OSS shall provide a GUI to enable and disable the PCI Optimization algorithm for all of the cells controlled by the eSON System or for specific ECGI(s).

- The OSS shall allow the operator to apply the PCI enable/disable setting by using the ERA restful API

  PUT
  [IP Addr:Port]/coe/apply?ecgis=all
  content: { "coe" : { "pcioe_enabled" : false } }

  For ECGI-specific queries replace "all" with one or more ECGIs.

- The OSS shall provide a GUI to enable and disable specific parts of the PCI algorithm. More specifically the GUI shall allow the operator to enable and disable

  1. Collision detection and resolution
  2. CRS collision detection and resolution
  3. Confusion detection and resolution

- The OSS shall apply enable and disable settings for items 1-3 in the above requirement by using the ERA restful APIs

  1. *PUT*
     [IP Addr:Port]/coe/apply?ecgis=all
     content: { "pcioe" : { "collision_detection_and_resolution_enabled" : false } }
  2. *PUT*
     [IP Addr:Port]/coe/apply?ecgis=all
     content: { "pcioe" : { "crs_collision_detection_and_resolution_enabled" : false } }
  3. *PUT*

[IP Addr:Port]/coe/apply?ecgis=all
content: { “pcioe” : { “confusion_detection_and_resolution_enabled” : false } }

4.1.3 RACH Optimization Configuration

The following are requirements to manage the RACH algorithm configuration from the OSS platform. The RACH Optimization algorithm includes RSI collision detection and resolution.

- The OSS shall provide a GUI to enable and disable the RACH Optimization algorithm for all of the cells controlled by the eSON System or for specific ECGI(s).
- The OSS shall allow the operator to apply the RACH Optimization enable/disable setting using the ERA restful API

  PUT
  [IP Addr:Port/coe/apply?ecgis=all
  content: { “coe” : { “rach_enabled” : false } }

  For ECGI-specific queries replace "all" with one or more ECGIs.

4.1.4 MCIM Configuration

The following are requirements to manage the algorithm configuration from the OSS platform.

- The OSS shall provide a GUI to enable and disable the MCIM algorithm for all of the cells controlled by the eSON System or for specific ECGI(s).
- The OSS shall allow the operator to apply the MCIM enable/disable setting using the ERA restful API

  PUT
  [IP Addr:Port]/coe/apply?ecgis=all
  content: { “coe” : { “mcimc_enabled” : false } }

  For ECGI-specific queries replace "all" with one or more ECGIs.

4.1.5 MLB Configuration

The following are requirements to manage the algorithm configuration from the OSS platform.

- The OSS shall provide a GUI to enable and disable the MLB algorithm for all of the cells controlled by the eSON System or for specific ECGI(s).
- The OSS shall allow the operator to apply the MLB enable/disable setting using the ERA restful API

  PUT
  [IP Addr:Port]/coe/apply?ecgis=all
  content: { “coe” : { “mlb_enabled” : false } }

  For ECGI-specific queries replace "all" with one or more ECGIs.

- The OSS shall provide a GUI to change the MLB configurable parameters summarized in Table 1.

MLB Parameter	Type (unit)	Range	Default	Description
overload_capacity_threshold	Integer (%)	0-100	80	Load percentage to be considered overloaded.
targetload_capacity_threshold	Integer (%)	0-100	80	Load percentage to be considered as an offload target.
fast_decay_step	Float (dB)	0-2	1	CIO fast decay step size.
slow_decay_step	Float (dB)	0-2	0.25	CIO slow decay step size.
attack_step	Float (dB)	0-2	1	CIO attack step size.
guard_a3_total	Integer (dB)	0-8	2	Minimum allowable distance between the handover boundaries between CellA→CellB and CellB→CellA.
max_a3_total	Integer (dB)	0-20	8	Maximum deviation of the handover boundary from RSRP_A=RSRP_B point.

Table 1 – MLB programmable parameters

- The OSS shall allow the operator to change the value of the MLB Overload Percentage Threshold parameter by using the ERA restful API

  PUT
  [IP Addr:Port]/coe/apply?ecgis=all
  content: { "cell" : { "overload_capacity_threshold" : [new value] } }

  For ECGI-specific queries replace "all" with one or more ECGIs.

- The OSS shall allow the operator to change the value of the MLB Target Percentage Threshold parameter by using the ERA restful API

  PUT
  [IP Addr:Port]/coe/apply?ecgis=all
  content: { "cell" : { "targetload_capacity_threshold" : [new value] } }

  For ECGI-specific queries replace "all" with one or more ECGIs.

- The OSS shall allow the operator to change the value of the MLB CIO Fast Decay Step Size parameter by using the ERA restful API

  PUT
  [IP Addr:Port]/coe/apply?ecgis=all
  content: { "mlb" : { "fast_decay_step" : [new value] } }

  For ECGI-specific queries replace "all" with one or more ECGIs.

- The OSS shall allow the operator to change the value of the MLB CIO Slow Decay Step Size parameter by using the ERA restful API

  PUT
  [IP Addr:Port]/coe/apply?ecgis=all
  content: { "mlb" : { "slow_decay_step" : [new value] } }

  For ECGI-specific queries replace "all" with one or more ECGIs.

- The OSS shall allow the operator to change the value of the MLB CIO Attack Step Size parameter by using the ERA restful API

  PUT
  [IP Addr:Port]/coe/apply?ecgis=all
  content: { "mlb" : { "attack_step" : [new value] } }

  For ECGI-specific queries replace "all" with one or more ECGIs.

- The OSS shall allow the operator to change the value of the MLB Guard A3 Total parameter by using the ERA restful API

  PUT
  [IP Addr:Port]/coe/apply?ecgis=all
  content: { "mlb" : { "guard_a3_total" : [new value] } }

  For ECGI-specific queries replace "all" with one or more ECGIs.

- The OSS shall allow the operator to change the value of the MLB Max A3 Total Abs parameter by using the ERA restful API

  PUT
  [IP Addr:Port]/coe/apply?ecgis=all
  content: { "mlb" : { "max_a3_total" : [new value] } }

  For ECGI-specific queries replace "all" with one or more ECGIs.

4.2 PM Counters and Events Interface

*** Details of counters and events are still under discussion. When the counters and events are finalized this section will be updated accordingly. ***

The PM counters and events interface allows performance counters and events to be sent from the eSON System to the OSS. The initial interface will support the pushing of two data files, a counters file and an events file, from the eSON System to the OSS. The requirements for this interface are

- The OSS shall provide a location and the mechanism to push eSON System counter files and eSON System event files to the OSS.
- The OSS shall be able to ingest the counters and events data in JSON format.
- The OSS shall be able to receive a counters file and an events file from each of the cells in the system every reporting period. The current reporting period is defined as 15 minutes.

The following sections describe the algorithm counters and events that will be included in the PM files.

4.2.1 Physical Cell Identity (PCI) Optimization Algorithm Counters and Events

The goal of the PCI Optimization algorithm is to automatically detect and resolve PCI conflicts including direct PCI collisions, CRS collisions, DMRS collisions, and PCI confusions.

The PCI counters in Table 2 are maintained for each cell that the eSON System manages. The counters are written to the counters file every reporting period.

PCI Counter Name	Description
Check Collision Count	Number of times the algorithm checks for all types of collision (direct, CRS, and DMRS)
Direct Collision Detected Count	Number of times the algorithm detects a direct collision
Direct Collision Proposed Count	Number of times the algorithm proposes a new PCI to resolve a direct collision
Direct Collision Resolved Count	Number of times the LTE cell accepts the new PCI proposed to resolve a direct collision, i.e. direct collision is resolved
CRS Collision Detected Count	Number of times the algorithm detects a CRS collision
CRS Collision Proposed Count	Number of times the algorithm proposes a new PCI to resolve a CRS collision
CRS Collision Resolved Count	Number of times the LTE cell accepts the new PCI proposed to resolve a CRS collision, i.e. CRS collision is resolved
DMRS Collision Detected Count	Number of times the algorithm detects a DMRS collision
DMRS Collision Proposed Count	Number of times the algorithm proposes a new PCI to resolve a DMRS collision
DMRS Collision Resolved Count	Number of times the LTE cell accepts the new PCI proposed to resolve a DMRS collision, i.e. DMRS collision is resolved
Check Confusion Count	Number of times the algorithm checks for a confusion
Confusion Detected Count	Number of times the algorithm detects a confusion
Confusion Proposed Count	Number of times the algorithm proposes a new PCI to resolve a confusion
Confusion Resolved Count	Number of times the LTE cell accepts the new PCI proposed to resolve a confusion, i.e. confusion is resolved

Table 2 - PCI Optimization counters.

The PCI counters represent the number of occurrences during the reporting period. The counters are reset at the end of the period.

Below is an example of the PCI counters format that are written into the counters file.

```
{
  "pcioe" :
  {
    "check_collision_count" : 1,
    "direct_collision_detected_count" : 1,
    "direct_collision_proposed_count" : 1,
    "direct_collision_resolved_count" : 1,
    "crs_collision_detected_count" : 0,
    "crs_collision_proposed_count" : 0,
    "crs_collision_resolved_count" : 0,
    "dmrs_collision_detected_count" : 0,
    "dmrs_collision_proposed_count" : 0,
    "dmrs_collision_resolved_count" : 0,
    "check_confusion_count" : 1,
    "confusion_detected_count" : 0,
    "confusion_proposed_count" : 0,
    "confusion_resolved_count" : 0
  }
}
```

The PCI events in Table 3 are written to the events file every reporting period.

Event Name	Parameters	Description
PCI direct collision detected	collided pci	A PCI collision with a direct (one-hop) neighbor has been found. The *collided pci* parameter indicates the home cell PCI.
PCI direct collision proposed	proposed pci	A new PCI is sent down to the LTE cell to resolve the PCI collision with a direct (one-hop) neighbor.
PCI direct collision resolved	accepted pci	A confirmation has been received from the LTE cell that the proposed PCI to resolve a direct PCI collision has been received
CRS collision detected	collided pci	A CRS collision with a direct (one-hop) neighbor has been found. The *collided pci* parameter indicates the home cell PCI.
CRS collision proposed	proposed pci	A new PCI is sent down to the LTE cell to resolve the CRS collision with a direct (one-hop) neighbor.
CRS collision resolved	accepted pci	A confirmation has been received from the LTE cell that the proposed PCI to resolve a CRS collision has been received.
DMRS collision detected	collided pci	A DMRS collision with a direct (one-hop) neighbor has been found. The *collided pci* parameter indicates the home cell PCI.
DMRS collision proposed	proposed pci	A new PCI is sent down to the LTE cell to resolve the DMRS collision with a direct (one-hop) neighbor.
DMRS collision resolved	accepted pci	A confirmation has been received from the LTE cell that the proposed PCI to resolve a DMRS collision has been received.
PCI confusion detected	confused pci	A PCI confusion with a two-hop neighbor has been found. The *confused pci* parameter indicates the home cell PCI.
PCI confusion proposed	proposed pci	A new PCI is sent down to the LTE cell to resolve the confusion.
PCI confusion resolved	accepted pci	A confirmation has been received from the LTE cell that the proposed PCI to resolve the confusion has been received.
PCI direct collision unresolved	unresolved pci	A PCI collision with a direct (one-hop) neighbor was detected, but was not resolved as the PCI pool is exhausted, i.e. all available PCIs are used by direct (one-hop) neighbors.
CRS collision unresolved	unresolved pci	CRS collision sacrificed for higher priority conflict
DMRS collision unresolved	unresolved pci	DMRS collision sacrificed for higher priority conflict
PCI confusion unresolved	unresolved pci	Confusion sacrificed for higher priority conflict

Table 3 – PCI Optimization events.

Below is an example of the format of the PCI Optimization events that are written into the events file each reporting period.

```
{"timestamp":1553127774593,"module":"pcioe","name":"check_collision"}
{"timestamp":1553127774593,"module":"pcioe","name":"check_confusion"}
{"timestamp":1553127774594,"module":"pcioe","name":"check_collision"}
{"timestamp":1553127774594,"module":"pcioe","name":"direct_collision_detected","collided_pci":200}
{"timestamp":1553127774594,"module":"pcioe","name":"direct_collision_proposed","proposed_pci":201}
{"timestamp":1553127774794,"module":"pcioe","name":"direct_collision_resolved","accepted_pci":201}
{"timestamp":1553127774794,"module":"pcioe","name":"check_confusion"}
{"timestamp":1553127774794,"module":"pcioe","name":"confusion_detected","confused_pci":201}
{"timestamp":1553127774794,"module":"pcioe","name":"confusion_proposed","proposed_pci":202}
{"timestamp":1553127776791,"module":"pcioe","name":"confusion_resolved","accepted_pci":202}
{"timestamp":1553127777803,"module":"pcioe","name":"check_collision"}
{"timestamp":1553127777803,"module":"pcioe","name":"direct_collision_detected","collided_pci":202}
{"timestamp":1553127777803,"module":"pcioe","name":"direct_collision_proposed","proposed_pci":201}
{"timestamp":1553127778099,"module":"pcioe","name":"direct_collision_resolved","accepted_pci":201}
```

Please note that all of the possible PCI events are not shown in the example.

4.2.2 Random Access Channel (RACh) Optimization Counters and Events

The goal of the RACh Optimization algorithm is to maximize the success rate of messages sent on the random access channel. More specifically, the algorithm provides Root Sequence Index (RSI) initial assignment, conflict detection, and conflict resolution as well as Zero Correlation Zone (ZCZ) optimization.

The RACh Optimization counters in Table 4 are maintained per cell.

Counter Name	Description
Check RSI Conflict Count	Number of times the algorithm checks for a collision
RSI Conflict Detected Count	Number of times the algorithm detects a collision
RSI Conflict Resolved Count	Number of times the algorithm proposes a new RSI that resolves a collision
RSI Conflict Unresolved Count	Number of times the algorithm cannot resolve a collision
PRACH Freq Offset Change Count	Number of times the algorithm changes the PRACH frequency offset
PRACH Config Index Changed Count	Number of times the algorithm changes the PRACH configuration index
Timing Advance Processed Count	Number of Timing Advance commands processed by the algorithm
Check ZCZ Config Index Count	Number of times the algorithm checks for ZCZ optimization
ZCZ Config Index Changed Count	Number of times the ZCZ Configuration Index changes

Table 4 – RACh Optimization counters.

The RACh Optimization counters represent the number of occurrences during the reporting period. The counters are reset at the end of the period.

Below is an example of the RACh Optimization counters format that are written into the counters file.

```
{
  "rach" :
  {
    "check_rsi_conflict_count" : 1,
    "rsi_conflict_detected_count" : 1,
    "rsi_conflict_resolved_count" : 1,
    "rsi_conflict_unresolved_count" : 1,
    "prach_freq_offset_change_count" : 0,
    "prach_config_index_changed_count" : 0,
    "timing_advance_processed_count" : 127,
    "check_zcz_config_index_count" : 0,
    "zcz_config_index_changed_count" : 0
  }
}
```

The RACh Optimization counters are written to the counters file every reporting period. The counters are reset after they are written to the counters file.

The RACh Optimization events that are maintained per cell are summarized in Table 5.

Event Name	Parameters	Description
RSI conflict detected	collided rsi config params	A RSI collision with a direct (one-hop) neighbor has been found. The *collided rsi config params* indicates the home cell RSI configuration parameters and includes RSI, PRACH configuration index, PRACH frequency offset and ZCZ configuration index.
RSI conflict resolved	resolved rsi config params	A new RSI is sent down to the LTE cell to resolve the PCI collision with a direct (one-hop) neighbor. The *accepted rsi config params* indicates the home cell RSI configuration parameters and includes RSI, PRACH configuration index, PRACH frequency offset and ZCZ configuration index.
RSI conflict unresolved	unresolved rsi config params	The algorithm cannot find an RSI to resolve the RSI conflict. However, new RSI configuration parameters are assigned to minimize the RSI resource overlap. The *unresolved rsi config params* indicates the home cell RSI configuration parameters and includes RSI, PRACH configuration index, PRACH frequency offset and ZCZ configuration index.

Table 5 – RACh Optimization events.

An example format for the RACh Optimization events is provided below.

```
{"timestamp":1556222355821,"ecgi":338573043720292,"module":"rach","name":"rsi_conflict_detected",”collided_rsi_config”:{"rsi":4
40,"high_speed":false,"zcz_config_index":0,"prach_freq_offset":0,"prach_config_index":0}}
{"timestamp":1556222355822,"ecgi":338573043720292,"module":"rach","name":"rsi_conflict_resolved","resolved_rsi_config":{"rsi":4
20,"high_speed":false,"zcz_config_index":0,"prach_freq_offset":0,"prach_config_index":0}}
{"timestamp":1556222359325,"ecgi":338573043720292,"module":"rach","name":"rsi_conflict_unresolved","unresolved_rsi_config":{"r
si":440,"high_speed":false,"zcz_config_index":0,"prach_freq_offset":0,"prach_config_index":0}}
```

4.2.3 Multi-Cell Interference Management (MCIM) Counters and Events

The goal of the MCIM algorithm is to minimize the interference for edge UEs in order to maximize the UE data throughput and spectral efficiency of the network.

The MCIM counters maintained per cell are summarized in Table 6.

Counter Name	Description
Evaluate Subband Mask Count	Number of times the MCIM algorithm has run. This number should increment every algorithm period (default: 2sec)
Evaluate Min Num Subband Count	Number of times the Min Num Subband algorithm has run. This number should increment every algorithm period (default: 6sec)
Edge UE Count Average	Average number of Edge UEs considered for subband mask evaluation
Edge UE Green Subband SINR average	Average SINR on green subbands for Edge UEs
Edge UE Red Subband SINR average	Average SINR on red subbands for Edge UEs
Edge UE Throughput Min	Minimum Edge UE throughput
Edge UE Throughput Max	Maximum Edge UE throughput
Edge UE Throughput Average	Average Edge UE throughput
Center UE Count Average	Average number of Center UEs per algorithm period (APPLICABLE ONLY IF CENTER UE CQIs ARE REPORTED OTHERWISE 0)
Center UE Throughput Min	Minimum Center UE throughput (APPLICABLE ONLY IF CENTER UE CQIs ARE REPORTED OTHERWISE 0)
Center UE Throughput Max	Maximum Center UE throughput (APPLICABLE ONLY IF CENTER UE CQIs ARE REPORTED OTHERWISE 0)
Center UE Throughput Average	Average Center UE throughput (APPLICABLE ONLY IF CENTER UE CQIs ARE REPORTED OTHERWISE 0)
Subband Mask Change Count	Number of times the subband mask changed
Stable Subband Mask Count	Number of consecutive algorithm periods the current subband mask has not changed
Stable Subband Mask Max	Maximum number of consecutive algorithm periods during which the subband mask did not change
All Green Subband Mask Count	Number of times the algorithm outputs an all green subband mask
All Green Subband Mask Percent	Percentage of time the algorithm outputs an all green subband mask
Green Subband Min	Minimum number of green subbands
Green Subband Max	Maximum number of green subbands
Green Subband Average	Average number of green subbands
Min Num Subband Min	Minimum Min Num Subband
Min Num Subband Max	Maximum Min Num Subband
Min Num Subband Average	Average Min Num Subband

Table 6 – MCIM counters.

The MCIM counters represent the number of occurrences during the reporting period. The counters are reset at the end of the period.

Below is an example of the MCIM counters format that are written into the counters file.

```
{
 "mcimc" :
   {
     "evaluate_subband_mask_count" : 31,
     "evaluate_min_num_subband_count" : 10,
     "edge_ue_count_average" : 1,
     "edge_ue_green_subband_sinr_average" : 21.850109,
     "edge_ue_red_subband_sinr_average" : 0.247171,
     "edge_ue_throughput_min" : 1,
     "edge_ue_throughput_max" : 254,
     "edge_ue_throughput_average" : 130,
     "center_ue_count_average" : 0,
     "center_ue_throughput_min" : 0,
     "center_ue_throughput_max" : 0,
     "center_ue_throughput_average" : 0,
     "subband_mask_change_count" : 1,
     "stable_subband_mask_count" : 27,
     "stable_subband_mask_max" : 27,
     "all_green_subband_mask_count" : 3,
     "all_green_subband_mask_percent" : 0.096774,
     "green_subband_min" : 6,
     "green_subband_max" : 13,
     "green_subband_average" : 6,
     "min_num_subband_min" : 6,
     "min_num_subband_max" : 13,
     "min_num_subband_average" : 6
   }
}
```

There are no defined events for MCIM.