## Use

The code we use to process the SanJole XML files is ssm\_wavejudge\_xml\_parser.py. It makes use of lte\_helpers.py and ltetbs.csv, which must be in the same directory. Use of this code is detailed below.

### ssm\_wavejudge\_xml\_parser.py

Proper use of this code requires Python3, which can be downloaded for free if not already installed. This code requires an xml input file like so.

>py ssm\_wavejudge\_xml\_parser.py -f Test\_File.xml

Additional options are outlined below.

|  |  |  |
| --- | --- | --- |
| Option | Description | Default |
| -h, --help | Displays use and options. | - |
| -f FILE, --file FILE | Input file to parse and process. Required. | - |
| -d DURATION, --duration DURATION | Length of processing time bin in milliseconds. | 1 |
| -b BANDWIDTH, --bandwidth BANDWIDTH | Bandwidth in resource blocks. | 50 |

Output of this code is a directory with the same name as the input file. In that directory, there is a subdirectory containing the parsed files, separated by DCI format. There is also a subdirectory containing the processed files, separated by uplink and downlink.

## Uplink Processing

We process uplink from DCI Format 0, with Resource Allocation Type 0. Other types may be included in the future, but appear infrequently enough to be considered negligible at the time of writing.

### Resource Allocation Type 0

Here, the Resource Allocation (RA) is the Resource Indication Value (RIV). This is converted to RBs using the MITRE code in lte\_helpers.py and ltetbs.csv. That code uses the standard RIV math, which can be found at the following link.

<https://www.sharetechnote.com/html/Handbook_LTE_RIV.html>

We also use the MITRE code to calculate the TX Bit allocation. As of 7/23/2020, these numbers have not been validated.

## Downlink Processing

In Downlink, we process DCI Formats 1, 1A, 1B, 1C, 1D, 2, 2A, 2B, and 2C. For this project we are currently interested only in the number of RBs allocated; not the specific RBs allocated. For this reason, some of the complexities of the downlink RA types could be neglected in the code.

### Resource Allocation Type 0

RA Type 0 is one of two types used by DCI Formats 1, 2, 2A, 2B, and 2C. When this type is used, RA in the SanJole data is an integer representing a bitmap. This bitmap refers to resource block groups (RBGs) allocated, where the number of RBs per RBG is determined by the system bandwidth. More details can be found at the following link.

<https://www.sharetechnote.com/html/Handbook_LTE_RAType.html>

Our code manages this by first setting the RBG to RB conversion based on the bandwidth. Then, it takes the RA and counts the set bits in its binary representation to get the number of RBGs allocated. The RBG count is then multiplied by the conversion factor to get the number of RBs. Since the last RBG sometimes has one less RB than the other RBGs depending on the bandwidth, the final step (if the bandwidth calls for it) is to correct the RB count. To do this, we check if that particular RBG (the least significant bit) is allocated and to subtract 1 from the number of RBs allocated if it is.

### Resource Allocation Type 1

RA Type 1 is the second of two types used by DCI Formats 1, 2, 2A, 2B, and 2C. Like in RA Type 0, the RA here is an integer bitmap and RBs are split into RBGs whose size depends on the system bandwidth. However RA Type 1 also splits the RBGs into a number of subsets equal to the RBG size. A UE is assigned to one subset and the bitmap refers to individual RBs allocated within the subset. More details can be found at the following link.

<https://www.sharetechnote.com/html/Handbook_LTE_RAType.html>

Our code determines the RB allocation by first setting the RBG to RB conversion based on the bandwidth. Then, it takes the RA and counts the set bits in its binary representation to get the number of RBs allocated.

### Resource Allocation Type 2

RA Type 2 is used by DCI Formats 1A, 1B, 1C, and 1D. In this type, the RA is an RIV. In order to convert to RBs, this RIV is divided by the RB bandwidth and rounded down to the nearest integer before adding 1. This provides the number of RBs which are allocated continuously from a start RB. The start RB is given by the remainder of that division. More details can be found at the following link.

<https://www.sharetechnote.com/html/Handbook_LTE_RAType.html>

The code for RA Type 2 simply integer divides the RA by the bandwidth and then adds 1 to get the RB allocation.