Parsing Case Study-

1. **Input [Hex String]**
2. **Output [JSON]**
3. **Goal - Understanding the Schema definition.**

Input -

“1111halo000001437000000096000002437000000096000003437000000096000004437000000096”

Output -

```
{
  token: ‘halo’,

  '000001':
   { volt: '230',
     curr: '75',
     load: 0.5081967213114754,
     'curr+volt': 305 },

  '000002':
   { volt: '230',
     curr: '75',
     load: 0.5081967213114754,
     'curr+volt': 305 },

  '000003':
   { volt: '230',
     curr: '75',
     load: 0.5081967213114754,
     'curr+volt': 305 },

  '000004':
   { volt: '230',
     curr: '75',
     load: 0.5081967213114754,
     'curr+volt': 305 }
}
```

Schema Defined -

```
{
  p_id :[0,4,"1111"],
  p_token :[4,8],
  len:4,
  key_inc:18,
  node_id:[8,14],
  spc:{load:"volt curr - volt curr + /",'curr+volt':"volt curr +"},
  volt :[14,22,"HS",'F'," volt 10 -"],
  curr :[22,26,'HS','I',"curr 2 /"]
}
```

Each Schema must contain these:
1. p_id
2. p_token
3. len
4. key_inc
5. node_id
6. spc
7. remaining user defined attribute

p_id :
It is a unique identifier allocated only once to a type of packet. It differentiate one coming packet from the remaining ones.

Example:
Schema

p_id :[0,4,"1111"]

0 = Element starting position - 1
4 = Element ending Position
"1111" = Equivalent to substring(0,4)

Input

"1111halo000001437000000096000002437000000096000003437000000096000004437000000096"

p_token :
It is a unique identifier distributed by the our platform to the end - devices for authentication.

Example:
Schema

p_token :[4,8]

4 = Element starting position - 1
8 = Element ending Position

Should be with database for authorisation.

Input

"1111halo000001437000000096000002437000000096000003437000000096000004437000000096"

len :
It is the number of the nodes incoming.

Example:
Schema

len: 4

4 = Number of nodes data incoming.

Input

"1111halo 000001437000000096 000002437000000096 000003437000000096 000004437000000096"

key_inc :
It is the size of the data of one node.

Example:
Schema

key_inc:18,

18 = Data of one node (starting from its node ID to the ending of the attribute

Input

“1111halo 000001437000000096 000002437000000096000003437000000096000004437000000096”

0	0	0	0	0	1	4	3	7	0	0	0	0	0	0	0	9	6
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

“000001” - Node ID
“43700000” - Hex string of voltage [4 BYTE - Float]
“0096**”** - Hex string of current [2 BYTE - Integer]
Total 0f 18 character in string.

node_id :
It is a identifier of a node.

Example:
Schema

node_id:[8,14]

8 = position of the first node ID starting element - 1
14 =position of the first node ID

Input
For First node ID ‘0’ [start] is at 9 position and ‘1’ [stop] is at ’14’ position.
“1111halo’0’0000’1’437000000096000002437000000096000003437000000096000004437000000096”

User Defined Attributes :
Here we have taken two attributes for demonstration. Voltage and current (volt and curr as for short names).

Example:
Schema

volt :[14,22,"HS",'F'," volt 10 -"],
curr :[22,26,'HS','I',"curr 2 /“]

Voltage case (Attribute names as volt)
14 = position of the first node ID starting element - 1
22 = position of the first node ID ending element
“HS” = Incoming type [HS] means Hex String.
“F” = Output type is Float means 4 bytes or 8 characters
“volt 10 -” is the postfix formula to the applied after the conversion of hex string to float

Note - the postfix string must be space separated for all the individual elements.

Input
For voltage ‘4’ [start] is at 15 position and 0 is at 22 position.
For current ‘0’ [start] is at 23 position and ‘6’ [end] is at 26 position.
“1111halo000001 ’4’370000’0’ ‘0’09’6’000002437000000009600000343700000000960000044370000000096”

Similarly we can do this to current.

spc :
These are the special cases of the calculation to be applied on the attributes defined by user and must be calculated for each node. (All are to be in postfix form).

Example:
Schema

spc:{load:"volt curr - volt curr + /",'curr+volt':"volt curr +”}

spc is the key as in identifier for the special cases. If there is no special cases, no of this key in the the schema document.

The Value of the the **spc** is an object which contains the user defined cases on the attributes which he defined above for example voltage and current.

load : “volt curr - volt curr + /“ is the postfix from of (volt-curr)/(volt+curr) defined by user as load. It must be a postfix and can be defined only for the attributes he mentioned.
The operation allowed till now for postfix are : +, -, * , / , ^ , %.