

EtherDATA supports (a very constrained version of) JSON i/o.

The standard JSON parser compiles to ~225k of flash - which is why I have imposed constraints to limit the scope.

Here are the rules:

- only supported input is a JSON string of the form;

```
{
  "cmd": "<function>",
  "args": [
    "<arg0>",
    "<arg1>",
    ...
    "<argN>"
  ]
}
```
- NO WHITE SPACE (except arg strings as appropriate)

Where;

<function>: the name of any supported function

<arg0...N>: any of JSON INTEGER, NULL, TRUE, FALSE or STRING

Here are some examples (in fact these are the only currently supported commands):

```
{"cmd": "dbRename", "args": [<addr>, "<name>"]}
```

Responds with;

```
{
  "dbRename": "<addr>", "name": "<name>"
}
```

```
{"cmd": "dbSetValue", "args": ["<group>", "<ident>", <value>]}
```

Responds with;

>no response<

If you wish to verify a change do a getFrom or similar

```
{"cmd": "dbGetValue", "args": ["<group>", "<ident>"]}
```

Responds with;

```
{
  "group": "<group>",
  "<ident>": "<value>"
}
```

```
{"cmd": "dbSubscribe", "args": ["<group>", "<ident>"]}
```

Responds with recurring results (rate defined by the value sampling rate);

```
{
  "group": "<group>",
  "<ident>": "<value>"
}
```

```
{"cmd": "dbUnsubscribe", "args": ["<group>", "<ident>"]}
```

Responds with;

< no response>

```
{"cmd": "emNeighborTable", "args": ["<node>"]}
```

Responds with;

```
{
  "emNeighborTable": "<group>",
  "set": [
    {
      "name": "<node>",

```

```

        "addr": <addr>,
        "rssi": <rssi>
    },
    {
        "name": "<node>",
        "addr": <addr>,
        "rssi": rssi
    }
]
}

{"cmd":"emResolveName","args":[<addr>]}

Responds with;
{
    "emResolvedName": "<node>",
    "addr": <addr>
}

{"cmd":"emResolveAddr","args":["<node>"]}

Responds with;
{
    "emResolvedName": "<node>",
    "addr": <addr>
}

```

Where;

<group>: any valid group name
 <ident>: any valid data value identifier (name)
 <value>: the value of the identified data item
 <node>: any valid node name (same as group name)
 <addr>: any valid node address (4 hexadecimal digit with no leading '0x')
 <rssi>: last measured RSSI value for data recieved from specified <node>

The string "local" may be used as a special group name referring to the local node (group).

Requirements;

- All requests should terminate with CR <or> NEWLINE.
- One line, one transaction - don't aggregate requests.
- Only one 'get' request (dbGetValue, emNeighborTable) pending at any time.

Here are the tested test cases:

These test the parser - do not use!!

```

{"cmd":"test","args":[]}
{"cmd":"test","args":[1]}
{"cmd":"test","args":["A"]}
{"cmd":"test","args":["A b","bodyTemp"]}
{"cmd":"test","args":[1,"bodyTemp"]}
{"cmd":"test","args":[true,"bodyTemp"]}
{"cmd":"test","args":[false,"bodyTemp"]}
{"cmd":"test","args":[false,"bodyTemp"]}
{"cmd":"test","args":[null,"bodyTemp"]}
{"cmd":"test","args":[1,null]}

```

NO WHITESPACE (except inside quoted string)!!!

These produce valid data (assuming you have the groups and idents as shown)

```

{
    "cmd":"dbRename","args":[8123,"B"]}
    "cmd":"dbRename","args":[1234,"happyhappynode"]}
    "cmd":"dbGetValue","args":["", "msgData"]}
    "cmd":"dbSetValue","args":["", "msgData","The rain in spain falls?"]}
    "cmd":"dbSetValue","args":["", "bodyTemp",3924]}
    "cmd":"dbGetValue","args":["", "bodyTemp"]}
    "cmd":"dbRename","args":[1234,"happyhappynode"]}

```

```
{ "cmd": "dbSetValue", "args": [ "A", "bodyTemp", 3924 ] }
{ "cmd": "dbSetValue", "args": [ "", "buzzer", true ] }
{ "cmd": "dbSetValue", "args": [ "Ex9", "buzzer", true ] }
{ "cmd": "dbSetValue", "args": [ "F", "buzzer", 0 ] }
{ "cmd": "dbSetValue", "args": [ "B", "buzzer", 1 ] }
{ "cmd": "dbSetValue", "args": [ "F", "buzzer", 2 ] }
{ "cmd": "dbSetValue", "args": [ "F", "buzzer", 3 ] }
{ "cmd": "dbSetValue", "args": [ "C", "bodyTemp", 3924 ] }
{ "cmd": "dbGetValue", "args": [ "A", "accelValue" ] }
{ "cmd": "dbGetValue", "args": [ "A", "emNeighborTable" ] }
{ "cmd": "dbGetValue", "args": [ "", "emNeighborTable" ] }
{ "cmd": "dbGetValue", "args": [ ".", "emNeighborTable" ] }
{ "cmd": "dbGetValue", "args": [ "", "batteryValue" ] }
{ "cmd": "dbGetValue", "args": [ "", "buzzer" ] }
{ "cmd": "dbGetValue", "args": [ "CC2541EM", "bodyTemp" ] }
{ "cmd": "dbSubscribe", "args": [ "A", "bodyTemp" ] }
{ "cmd": "dbSubscribe", "args": [ "B", "bodyTemp" ] }
{ "cmd": "dbSubscribe", "args": [ "C", "bodyTemp" ] }
{ "cmd": "dbSubscribe", "args": [ "E", "bodyTemp" ] }
{ "cmd": "dbSubscribe", "args": [ "E", "batteryValue" ] }
{ "cmd": "dbSubscribe", "args": [ "F", "accelValue" ] }
{ "cmd": "dbSubscribe", "args": [ "CC2541EM", "batteryValue" ] }
{ "cmd": "dbSubscribe", "args": [ "CC2541EM", "bodyTemp" ] }
{ "cmd": "dbUnsubscribe", "args": [ "Alfa", "bodyTemp" ] }
{ "cmd": "dbUnsubscribe", "args": [ "B", "bodyTemp" ] }
{ "cmd": "dbUnsubscribe", "args": [ "C", "bodyTemp" ] }
{ "cmd": "dbUnsubscribe", "args": [ "F", "accelValue" ] }
{ "cmd": "dbUnsubscribe", "args": [ "CC2541EM", "bodyTemp" ] }
{ "cmd": "emNeighborTable", "args": [ "" ] }
{ "cmd": "emNeighborTable", "args": [ "." ] }
{ "cmd": "emResolveName", "args": [ 4660 ] }
{ "cmd": "emResolveAddr", "args": [ "" ] }
{ "cmd": "emResolveName", "args": [ 21500 ] }
{ "cmd": "emResolveAddr", "args": [ "A" ] }
{ "cmd": "emResolveName", "args": [ 18742 ] }
```