
C# interview task - WebAPI variant

Design and implement a basic Web API application in .NET 6+ environment, which will parse provided strings/packets into an object. Packets can have different formats with different fields based on the first byte. Feel free to use any well-known software patterns and abstractions suitable for the task. You do not have to try to write a super effective and sleek code. It should be easily understandable in the first place.

Application should implement two endpoints. More details can be found in OpenAPI (Swagger) documentation (C# interview task - WebApi variant.yaml)

1. First endpoint (/service/data) will accept data as described in this document. Endpoint request data format is not defined (you can choose one). Messages will be parsed and printed in the console.
    a. When a message with format **0x0A** is received, this message should be saved in DB. DB does not have to be regular external DBMS, it can be implemented in-memory.
2. Second endpoint (/api/data) will respond with the list of messages from the database. It will return the messages as JSON.

Packet formats

0x0A format:

```
0x0A 23 006C 8D9A87 (35 | 0 | 108.141.154.135 )
```

- Byte 0, format code
- Byte 1, integer value (Identifier)
- Byte 2, encoded state (0=REQ, 1=RESP, 2=ACK, 3=END)
- Byte 3-6, 4byte number (IP address)

0x0E format:

```
0x0E 55 9809 41424344 ( 85 | 2456 | ABCD )
```

- Byte 0, format code
- Byte 1, integer value between 0 - 100 (Battery)
- Byte 2-3, integer value, in LSB (Temperature)
- Byte 4-7, four ASCII characters

0x0F format:

```
0x0F 04 DD 7A B1 05 36 AF 41(12.45 | 122[01111010] | 1457 | 13999 |
A)
```

- Byte 0, format code
- Byte 1-2, decimal number encoded as integer / 100 (Battery)
- Byte 3, encoded bits/flags (bits from left to right > -, Active, Error, Connected, Multi, Single, Acknowledged, Repeat)
- Byte 4-5, integer value LSB (Axis1)

- Byte 6-7 integer value MSB (Axis2)
- Byte 8, ASCII char,
    - if A, then two previous values must be multiplied by 10
    - if B, then two previous values must be subtracted by 100
    - if C, then two previous values must be added by 25

Create a class **Parser**, with a method, which accepts the packet as a string or byte array. Based on the format code (first byte) it will decide how to parse the data. It will return a **Packet** object with all properties of the packet according to the defined packet format. The object will be printed contents to the console formatted as JSON. Structure of the JSON is not strictly defined, but it should contain the fields according to the defined packet structure.

Input

```
0x0E55980941424344
0x0A23006C8D9A87
0x0F04DD7AB10536AF41
0x0F03F72C1699626843
0x0F0017276B0473D941
0x0A5E0027473D9D
0x0AE602F990BE69
0x0F11E8742847604642
0x0AB8014A8FF39A
0x0A150351CE4E84
0x0A0F01EA387943
0x0E2EF32051474753
0x0E30210C47465355
0x0F02263AA0EE559842
0x0A7F02D643C4FC
0x0E57D60F50585146
0x0F210C1B809AC81A43
0x0F1E8F1277ECFA5941
0x0A860365C399F2
0x0E542C09414D5446
0x0A560092ACC07D
0x0F02004B183BF00242
0x0A2602C9ECA1CD
0x0E431E0B55474451
```

Example of parsed value of first three inputs

```
{ Type: 0x0E, Battery: 85, Temperature: 2456, Chars: 'ABCD' }
{ Type: 0x0A, Id: 35, State: REQ, IpAddr: '108.141.154.135' }
```

```
{ Type: 0x0F, Battery: 12.45, Flags: 122 or [01111010] or ['Active', 'Error', 'Connected', 'Multi', 'Acknowledged'], Char: 'A', Axis1: 1457*10, Axis2: 13999*10 }
```