Assignment

We are building our applications based on a Microservice architecture that enables us to use different technologies, however, C# / .NET Core is one of the central and most important stacks.

In order for you to get an idea about what we are working on and for us to see how comfortable you are in these technologies, here is an assignment for you 😊

Your task is to implement and build service in C# / .NET Core that provides the following functionalities:

- Host a REST API (including a Swagger Definition)
- Ability to create a product definition with the following properties
    - **Company Prefix: Numeric**
    - **Company Name: String**
    - **Item Reference: Numeric**
    - **Product Name (Item name): String**
- Ability to post inventory data, which comes in the following format
    - **Inventory Id: Alphanumeric - up to 32 characters**
    - **Inventory Location: String**
    - **Date of inventory: DateTime**
    - **List of tags encoded using SGTIN-96 standard which represents inventoried items at a particular location (more on this later)**
- For the sake of simplicity, you can simply store the data in memory (i.e.: you don't have to persist it in a database)
- Ability to request the following data/information from the API
    - Count of inventoried items grouped by a specific product for a specific inventory
    - Count of inventoried items grouped by a specific product per day
    - Count of inventoried items grouped by a specific company

Remarks

We kindly ask you to provide us with the source code (Visual Studio 2019 solution/source files in a GitHub repository) of your implemented solution. Please, also roughly note the time effort that you have invested in the implementation (hours spent on the task). We will look at your implementation from the following angles:

- Basic aspects: Can we build and run your solution?
- Functional aspects: Does your solution implement the required functionality?
- API Design: Documentation, Ease of Use, and Robustness
- Error Handling: How well do you handle unexpected cases and errors

Bonus

If you have time (and feel like it), there is always headroom. Keywords are

- Database / Persistence Layer
- Docker
- Hosting the service in the cloud, preferably MS Azure
- Surprise us!

What is SGTIN-96? 😊

SGTIN or Serialized Global Trade Item Number EPC scheme is used to assign a unique identity to an instance of a trade item, such as a specific instance of a product or SKU.

The easiest way how to understand is to compare it with the UPC number which you can find under the barcode of any trade item bought in store. The main difference is that UPC only represents a specific product type of some manufacturer while the SGTIN number gives a specific instance of that same product type of the same manufacturer.

The easiest way how to understand is to compare it with the UPC number which you can find under the barcode of any trade item bought in store. The main difference is that UPC only represents a specific product type of some manufacturer while the SGTIN number gives a specific instance of that same product type of the same manufacturer.

UPC will only tell us that item is Milka Oreo chocolate, but SGTIN will tell us the same alongside with unique serial item of that chocolate. This allows us to be aware of specific instances of the items around us.

Now some more technical details about the SGTIN standard. It comes in 3 different types:

- SGTIN-64
- SGTIN-96
- SGTIN-198

The number after SGTIN represents a number of bits that the SGTIN number can hold. So, for SGTIN-96 we expect 96 bits of information.

SGTIN consists of the following important elements:

- GS1 company prefix
- Item reference (item type)
- Serial number

In the SGTIN EPC you will also find the following elements that are very important for the process of decoding the number:

- Header
- Filter
- Partition

We already told that SGTIN-96 EPC consists of 96 bits and they are distributed in the following order:

- Header (8 bits)
- Filter (3 bits)
- Partition (3 bits)
- GS1 company prefix (20 – 40 bits)
- Item reference (24-4 bits)
- Serial reference (38 bits)

Header value

Header value of SGTIN-96 is always fixed to the same value which is represented in binary as:

0011 0000

Header Value (binary)	Header Value (hexadecimal)	Encoding Length (bits)	Coding Scheme
0011 0000	30	96	SGTIN-96
0011 0110	36	198	SGTIN-198

Filter

Filter values are numeric from 0 to 7 (decimal). The full list of possible filter values is:

Type	Filter Value	Binary Value
All Others	0	000
Point of Sale (POS) Trade Item	1	001
Full Case for Transport	2	010
Reserved	3	011
Inner Pack Trade Item Grouping for Handling	4	100
Reserved	5	101
Unit Load	6	110
Unit inside Trade Item or component inside a product not intended for individual sale	7	111

Partition

As you probably saw on the previous page company prefix and item reference parts are not fixed in size. So how to decode something which we don't know the exact length? Say hello to Partition value. Partition values represent how much out of 44 bits are spend on company prefix and how much on item reference. Partition numeric value can only be between 0 and 6 which clearly can be seen in the following table:

Partition Value (*P*)	GS1 Company Prefix		Indicator/Pad Digit and Item Reference	
	Bits (*M*)	Digits (*L*)	Bits (N)	Digits
0	40	12	4	1
1	37	11	7	2
2	34	10	10	3
3	30	9	14	4
4	27	8	17	5
5	24	7	20	6
6	20	6	24	7

So, if our partition value is 3 we then know that the company will be represented with 30 bits and item reference with 14 bits. Note that company prefix and item reference are always represented with exactly 44 bits (or 13 digits) when dealing with the SGTIN-96 standard of encoding.

Company prefix

This numeric value represents the company prefix. Based on this prefix (code) we can exactly know which company manufactured the item that we are trying to decode.

All the company prefixes are defined by a “company” called GS1 who is responsible for managing the database of all company prefixes. To get a prefix from GS1 you need to pay a hefty fee (of course 😊), but that is not relevant for our task.

Company prefix can be represented using 6 to 12 digits.

Item reference

Item reference is a numeric value that can be represented using 1 to 7 digits. Item reference is defined by the company or manufacturer of the specific item that we are trying to encode/decode using SGTIN-96.

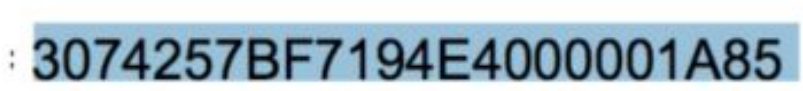
Serial reference

This is a unique serial number for a combination of the company prefix and item reference. This number will distinct specific instance of a product.

In SGTIN-96 this is a numeric value and its range is from 0 to 274 877 906 943. As you can see there are a lot of serial number combinations.

So, to finalize the whole representation of the SGTIN EPC lets represent it in 2 ways:

1. The first one is Hexadecimal format (a format that you will send to the inventory endpoint)



2. The second one is a binary representation which is important when decoding the EPC

Header (8bits) = 0011 0000
F: Filter (3bits) = 011

Company Prefix (7values) and Item Ref (6values) → Partition = 5

Partition (3bits) = 101
C: Company Prefix (24bits) = 000010010101111011111101
I: Item Ref (20bits) = 11000110010100111001
S: Serial (38bits) = 000…0001101010000101

EPC binary (96bits) =
00110000011101000010010101111011111101110001100101001110010000000000000000000000001101010000101

Note: The binary values above have been padded with leading 0's to fill their respective bit spaces. For example, with a partition value of "5", the Company Prefix bit space is 24 bits but the binary conversion is only 14 bits long thus it is padded with ten "0" bits.

Conclusion

So, based on this short course and introduction to the SGTIN-96 standard and its definition try to use this new knowledge to solve assignments at the beginning of this document. To make your life easier alongside this document you will find 2 files that contain some test data that you can use to test your solution (**tags.txt** a list of SGTIN-96 tags and **data.csv** - some dummy product data).

Feel free to use Google to find out more about SGTIN and how encoding/decoding is done. Also, if you will have any additional questions or there is some confusion with the task and its requirements feel free to contact us :)