TSB Profile Generator

By MalsKippetje | SneakerSven #1037

Insta: @sneaker.sven

Table of contents

Introduction

Welcome to the TSB Profile Generator!

With this generator you can create thousands of *The Shit Bot* ready profiles in mere seconds. All the program needs is a text file containing your creditcard details, which you can extract from your current TSB profiles export using this generator, too!

This guide will show you the principles by which the generator jiggs your address, and how to use the generator. First, the functionality for retrieving your credit card details from your TSB export will be explained. Secondly, the profile generating process will be explained. The program itself will also guide you through the different processes. This guide is to provide more detailed information about the generator.

Your creditcard details are untouched and will not be tampered with. To ensure the transparancy of the generator, the code is open source and anyone can clone the repository. You are free to clone the code and make any changes as you see fit. A link to the repository can be found in the *#welcome* channel of the TSB Profile Generator discord. Permanent invite link here.

If you have any problems using the generator, dont hesitate to contact the devs on the discord channel!

Good luck pooping! 💩

How to run

This chapter shows the steps you have to take in order to run the generator.

Steps

1. Download the latest version of the generator from the #latest-version channel on the discord server
2. Place the .jar file in a folder that you can remember
   - We recommend creating a folder on your desktop or in your documents folder, and only putting the .jar file inside of it
3. Make sure you have at least Java version 13 installed on your pc
   - You can download the latest version here. This is a Development kit, but it works like normal
4. Open a command line. For windows, search *cmd* in start and hit enter
5. Check if Java is installed correctly by running *java --version*
   - Depending on your Java version, you might only have to type one "-"
6. Navigate to the folder where you placed the .jar file of the generator
   - You do this with the *cd* command. Eg: *cd Desktop\TSBPG*. When you're typing the name of a folder, you can hit tab to auto complete it's name. Google knows plenty about the *cd* command, too. Make sure to use the \ slash, not the /
7. When you're in the correct folder, type the command *java -jar* and hit tab
   - If the .jar file is the only file in that folder, pressing tab will auto complete the command by entering the name of the .jar file. Eg: *java -jar TSB_Profile_Generator_v1.1.3.jar*
8. The program will now start up! Follow the steps the generator provides

You only have to do steps 1 through 5 the first time you run the generator. After the first time, you only have to do steps 4, and 6 through 8.

If you want to update your generator with the new latest version, simply delete the old .jar file from your folder and place the newly downloaded .jar file in that folder.

When you run the generator for the first time and choose one of the two functionalities to run, the generator will create the input folder in which you have to place the input files. This folder will be located at *C:/TSB Profile Generator/input*.

We know that using the command line isn't favorable. If you need any help, you can ask your questions in the #help channel on the discord server.

Retrieve credit card details

The program can retrieve your credit card details from your profiles export from TSB. Exporting your profiles within TSB will create a .json file. You have to place this file in the input folder of the generator. This folder is located at *C:/TSB Profile Generator/input*. This folder will be created when you first run this functionality. The program will ask you to place your TSB export into the folder, and to rerun the program after you've done that.

The .txt that gets generated by this functionality can be used for generating TSB profiles. The generating of the profiles and the jigging of the addresses will be explained further in the next chapter of this guide.

The generated .txt will be located in the C:/TSB Profile Generator/output folder.

If you don't have a TSB profiles export ready to use, that is no problem! You can type the credit card details yourself in the .txt input file.

Credit card details input pattern

You should use the following pattern in the .txt input file:
< PROFILENAME > ; < PHONE NUMBER> ; < CC NUMBER > ; < EXPIRY DATE > ; < CVV >

Example:
REVOLUT 1;+31678654312;0000 0000 0000 0000;11 / 25;053

Dont forget the ; in between each field, don't add spaces surrounding them. Only one credit card per line!

Profile sorting

The program will sort all your profiles based on their profile name. Both numerically and alphabetically. This is either based on a number at the end of your profile name, or the entire profile name. So for "REVOLUT 1", "REVOLUT 2" it will order by the 1 and the 2 numerically. And for "THE FIRST PROFILE", "THE SECOND PROFILE" it will order by the name alphabetically.

If you have both numeric and alphabetic names, it will first generate numeric profiles and then the alphabetical ones after.

Generate profiles

This chapter touches the jigging of your address. This functionality takes your credit card details in a .txt file as input, and will give you a .json file containing TSB ready profiles as output.

The input .txt can also be generated using your TSB profiles output. For generating the input file, refer to the previous chapter.

The program will ask you about some jigging settings, and it will ask for your address details. The jigging patterns, jigging settings and address details will be explained in the following subchapters.

The generated .json file will be located at C:/TSB Profile Generator/output.

Jigging settings

After choosing your jigging pattern, the program will ask you for some more jigging settings. Provided below is an ordered list of all settings the generator asks input for.

Setting no#	Description
1	Choose jigging pattern *(see next subchapter)*
2	Use randomized delivery phone numbers. This is a yes or no question.
2a	(Only shown if setting no# **2** is yes) Beginning of phone number to use for valid random phone numbers. *EG: +316 for Dutch phone numbers.*
2b	(Only shown if setting no# **2** is yes) Amount of random numbers that have to be added to the beginning of each phone number to generate valid phone numbers. *EG: 8 for Dutch phone numbers.*
3	Use randomized first and last names for each profile. If you choose no, you will have to put in the first and last name to use at the address details. This is a yes or no question. *(for now these names are Dutch first and last names)*
4	Add 0 to 2 random characters to the end of city field, with a space in between the city name and the characters. This is a yes or no question.
5	Use shipping address as billing address. This is a yes or no question.
6	Amount of times that the program has to generate a profile with a jigged address for each provided credit card.

Table 1: Jigging settings

Address 1 and 2 jigging patterns and examples

Let's start by explaining the jigging patterns. The program will first ask you about your jigging settings. First you choose one of the 6 Address 1 jigging patterns, or a mix of all patterns. Below is a table showing all available jigging patterns.

Note that the 'TSBPG' in the examples stands for a randomized piece of text, and the '534' for a randomized number.

For the examples below, the user input for Address 1 is "Streetname 1".

Pattern no#	**Address 1 jigging patterns**
1	TSBPG Streetname 1
2	TSBPG Streetname 1 534
3	TSBPG Streetname 1 534 TSBPG
4	TSBPG Streetname 1 TSBPG
5	Streetname 1 534 TSBPG
6	Streetname 1 TSBPG
7	Randomized mix of all of the patterns above

Table 2: Address 1 jigging patterns

Do you have any suggestions for another jigging pattern? Please let us know in the *#suggestions-and-todo-list* channel in the discord server!

The address 2 field will also get jigged by the program. This has a 75% chance of happening, so that not every profile uses an address 2. If you type anything into address 2, it will not get jigged. If you leave it empty, it will randomly choose a prefix from the list of prefixes below, and add a random suffix. The pattern for this suffix is 1 random number between 1 and 5, and a random letter between A and E.
See the table below for a list of all the available prefixes, and for address 2 jigging examples.

Address 2 prefixes
Appt., Appartement, Floor, Verdieping, Suite, Room, Kamer
Address 2 jigging examples
Appt. 3A Suite 5E Kamer 1C Appartement 2D

Table 3: Address 2 jigging patterns

Address details and jigging

After you set up the jigging settings, the program will ask for your address details. If you want a field to be empty, you can type nothing and just hit enter. If you leave address 2 empty, it will be jigged as shown in the subchapter called 'Address 1 and 2 jigging patterns and examples'. Provided below is a table showing all the address details the program will ask input for.

Address field no#	**Description**
1	First name *Will only be shown when you don't want random first and last names*
2	Last name *Will only be shown when you don't want random first and last names*
3	Address 1 *This will be jigging using the patterns shown in the subchapter called 'Jigging patterns and examples'*
4	Address 2 *This will be jigging using the patterns shown in the subchapter called 'Jigging patterns and examples'*
5	Zip *This field will not be jigged by the program*
6	City *This will be jigged using the patterns stated in the subchapter called 'Jigging settings' (setting no# 4)*
7	Country *This field will not be jigged by the program*
8	State *This field will not be jigged by the program*

Table 4: Address details

Example of jigging

This chapter shows a few examples of jigging your address using the TSB Profile Generator. The example address we are going to use to jigg, will be stated first. As will the jigging settings that get used.

Example input

Jigging settings no#	Input
1	7
2	Y
2a	+316
2b	8
3	Y
4	Y
5	Y
6	1

Table 5: Jigging settings example

Address field	Input
Address 1	Colosseum 1
Address 2	
Zip	1213NL
City	Hilversum
Country	Netherlands
State	

Table 6: Example input for jigging examples

The input .txt file we are going to use looks like this:
REVOLUT 1;+31612345678;0000 0000 0000 0001;3 / 26;001
REVOLUT 2;+31612345678;0000 0000 0000 0002;3 / 26;002
REVOLUT 3;+31612345678;0000 0000 0000 0003;3 / 26;003
REVOLUT 4;+31612345678;0000 0000 0000 0004;3 / 26;004
REVOLUT 5;+31612345678;0000 0000 0000 0005;3 / 26;005
REVOLUT 6;+31612345678;0000 0000 0000 0006;3 / 26;006

On the next page, a table containing the result of jigging our provided address will be shown. We used 6 credit card details so that we get a nice mix of jigged addresses. When you use more credit cards, and choose to jigg your address for each credit card more than once, you will ofcourse get a better mix of jigged addresses. As to not fill up the entire guide with example output, we only used 6.

Example output profiles

The output that is shown below has been copied from the output .json file. The whitespaces between each profile object are just for readability, and won't be added to the actual output .json file.


```
[
    {
        "cc": {
            "profileName": "REVOLUT 1 (J1)",
            "ccCvc": "001",
            "ccNumber": "0000 0000 0000 0001",
            "phone": "+31685810050",
            "ccExpiry": "3 / 26"
        },
        "date": 1614710609721,
        "isRussianAddress": null,
        "isMexicanAddress": null,
        "isPhilippinesAddress": null,
        "shipping": {
            "zip": "1213NL",
            "firstName": "Jessica KD",
            "lastName": "Peters",
            "country": "Netherlands",
            "address": "BA Colosseum 1 HJ",
            "address2": "Verdieping 5A",
            "city": "Hilversum K",
            "state": null
        },
        "isJapaneseAddress": null,
        "billing": {"billingSameAsShipping": true}
    },


    {
        "cc": {
            "profileName": "REVOLUT 2 (J1)",
            "ccCvc": "002",
            "ccNumber": "0000 0000 0000 0002",
            "phone": "+31621422900",
            "ccExpiry": "3 / 26"
        },
        "date": 1614710609721,
        "isRussianAddress": null,
        "isMexicanAddress": null,
        "isPhilippinesAddress": null,
        "shipping": {
            "zip": "1213NL",
            "firstName": "Hilda B",
            "lastName": "Janssen",
            "country": "Netherlands",
            "address": "Colosseum 1 811 WQW",
            "address2": "Suite 1B",
            "city": "Hilversum J",
            "state": null
        },
```

```
    "isJapaneseAddress": null,
    "billing": {"billingSameAsShipping": true}
},


{
    "cc": {
        "profileName": "REVOLUT 3 (J1)",
        "ccCvc": "003",
        "ccNumber": "0000 0000 0000 0003",
        "phone": "+31637629174",
        "ccExpiry": "3 / 26"
    },
    "date": 1614710609726,
    "isRussianAddress": null,
    "isMexicanAddress": null,
    "isPhilippinesAddress": null,
    "shipping": {
        "zip": "1213NL",
        "firstName": "Emiel",
        "lastName": "de Lange",
        "country": "Netherlands",
        "address": "QPKM Colosseum 1 198 SQN",
        "address2": "Room 5B",
        "city": "Hilversum FP",
        "state": null
    },
    "isJapaneseAddress": null,
    "billing": {"billingSameAsShipping": true}
},

{
    "cc": {
        "profileName": "REVOLUT 4 (J1)",
        "ccCvc": "004",
        "ccNumber": "0000 0000 0000 0004",
        "phone": "+31620808420",
        "ccExpiry": "3 / 26"
    },
    "date": 1614710609726,
    "isRussianAddress": null,
    "isMexicanAddress": null,
    "isPhilippinesAddress": null,
    "shipping": {
        "zip": "1213NL",
        "firstName": "Johanna BA",
        "lastName": "de Wit",
        "country": "Netherlands",
        "address": "RFCZ Colosseum 1",
        "address2": "Appartement 1B",
        "city": "Hilversum ZY",
        "state": null
    },
    "isJapaneseAddress": null,
    "billing": {"billingSameAsShipping": true}
},
```

```
    {
        "cc": {
            "profileName": "REVOLUT 5 (J1)",
            "ccCvc": "005",
            "ccNumber": "0000 0000 0000 0005",
            "phone": "+31664753082",
            "ccExpiry": "3 / 26"
        },
        "date": 1614710609726,
        "isRussianAddress": null,
        "isMexicanAddress": null,
        "isPhilippinesAddress": null,
        "shipping": {
            "zip": "1213NL",
            "firstName": "Pien BL",
            "lastName": "de Bruin",
            "country": "Netherlands",
            "address": "VMW Colosseum 1 PT",
            "address2": "Suite 2B",
            "city": "Hilversum G",
            "state": null
        },
        "isJapaneseAddress": null,
        "billing": {"billingSameAsShipping": true}
    },


    {
        "cc": {
            "profileName": "REVOLUT 6 (J1)",
            "ccCvc": "006",
            "ccNumber": "0000 0000 0000 0006",
            "phone": "+31676080916",
            "ccExpiry": "3 / 26"
        },
        "date": 1614710609726,
        "isRussianAddress": null,
        "isMexicanAddress": null,
        "isPhilippinesAddress": null,
        "shipping": {
            "zip": "1213NL",
            "firstName": "Puck",
            "lastName": "Bos",
            "country": "Netherlands",
            "address": "ETF Colosseum 1",
            "address2": "Verdieping 3B",
            "city": "Hilversum U",
            "state": null
        },
        "isJapaneseAddress": null,
        "billing": {"billingSameAsShipping": true}
    }
]
```


Table 7: Example output of jigged addresses