Unturned currently saves game data in the Windows registry. Whenever we read from anywhere in Windows, the data will most likely be Windows-1252 encoding. Unturned saves data as UTF-8 encoded **obfuscated** strings.
It turns out, that every character in the original string has been added to 32. So we subtract 32 from the character’s unicode value and MOD by 255 to get a ASCII character. Then we reverse the string.

```
//x = input[i].unicode
// (x + 32 + 255*n) = resulting unicode value
//In Decryption function: (input.unicode[i]) = (x + 32 + 255*n)
// [(x + 32 + 255*n)-32) % 255] = (x + 255*n) % 255 = result
// result = x, as long as (x + a) where a is a multiple of 255
//May get different characters when encrypted, but the decrypted output should always be the
same.
```

This is called Residual Arithmetic.
(x + y) mod N = ((x mod N) + (y mod N)) mod N

Data flow diagram thing:

- game saves the string in UTF8 encoding (probably) to the registry
- we read that string so it gets read in Default encoding
- to de-obfuscate it, we clearly need the same encoding as it was encrypted in
- transform that thing from Default to UTF8
- shift the int value of each character by 32 (why 32 Nelson, WHYYY?, god knows)
- MOD every value to 255 so we get valid bytes
- rebuild the string with FINALLY READABLE CHARACTERS

(more on encoding on Wikipedia, I don't understand it better either)

Example:

```
"â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â
€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ
€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â
€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€º
â€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€ºâ€ºâ€±â€ºâ€°â€°â€°â€¸â€±â€»â€ºâ€ºâ€°â€ºâ€±â
€â€»â€ºâ€ºâ€±â€ºâ€¶â€°â€°â€¸â€»â€ºâ€ºâ€°â€ºâ€±â€â€»â€°â€°â€µâ€·â€±â€»â€µâ€»â€µ"
```

(ugly string, directly read from the registry)

-notice the aE parts and, unlike my other way of reading it, they should NOT be left out
-what UTF8 means is that we take 1 to 4 bytes to recreate a character
-seeing on the wiki what should I end up with, I tried many different approaches to it and ended up with byte shifting (I believe that's its name)
-this is not entirely my work because I knew the end result, but this way i got the algorithm (I have no idea where the guy who edited the wiki got that XD, but I thank him alot)

-deobfuscating that string with the algorithm provided, we end up with:
"5;5;17500;-1:0::;8006:1::;-1:0::;18000:1::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;"

-now making the connections
-I have a rucksack, so 17.5 kg with 5x5 inventory
-slot 2 is taken by a baseball bat and slot 4 is taken by one wooden board

-"5;5;17500" is related to backpack (as I have already figured out and explained into the old save format)
-"-1:0" can only be an empty item
-baseball bat id is 8006 and board is 18000

-problem solved
-any general observation made in the old save format seems to still apply (regarding separators)

Now on to separate item types explanation:
We will use a completely new inventory, with each item type:

"5;5;17500;8002:1::;10002:8::;20000:1:f:;8008:1:b:;7001:1:4_10002_-1_-1_-1_0_y_:;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;-1:0::;"
rucksack
slot 1 = pickaxe
slot 2 = swift magazine with 7/7
slot 3 = filled gas can

slot 4 = torch turned on

slot 5 = colt with no attachments, 4/7 bullets on safety

First 3 values separated by ";":

    -first: backpack width

    -second: backpack height

    -third: backpack max weight (multiplied by 1000, in KG)

    Example: a 3x5 inventory with max weight of 61.33 KG

            "3;5;61330;"

Normal items (pickaxes, boards, berries, arrows, that sort of stuff)

    -as we know, every thing is delimited by ";" and the first 3 values are backpack information

    -next value is the first item in the inventory: the pickaxe

    "8002:1::"

    -pickaxe = ID 8002

    -as we already know from the old save format examination, every item has other delimiters that specify other values (in this case ":")

    -we have one pickaxe, so it's logical to assume that the second value is the item count

Magazine items:

    -next value by ";"

    "10002:8::"

    -swift magazine ID = 10002

    -bullets = 7

    -as we know already (old save format), the item count of a magazine is the number of bullets + 1 (we can have empty mags)

Fillable items (gas can, canteen, are there others?):

    -next value by ";"

    "20000:1:f:"

    -gas can ID = 20000

    -we have only one of those items

    -new value! "f"

    -my spidey sense is telling me that the word "full" starts with "f"

    -to be sure, empty the gas can in game and re-read the inventory

"20000:1:e:" (empty gas can)
-surprise, surprise, we have "e/f" (empty/full)

Toggleable items (torches, handlamps, night vision, miner helmet, tactical light/laser):
-next value by ";"
"8008:1:b:"
-torch ID = 8008
-count = 1
-another value! "b"
-it surely must be the turned on/off value
-going back to game and turning it off
"8008:1:d:"
-I see no connection between b/d and on/off, but let's pretend we know (b&d rings me a bell though =)), if you know what I mean)
-b=on, d=off

-for some reason, wether the nightvision/helmet is on/off is not reflected in the inventory string, even though the item retains its state (not even in the raw inventory string)
-probably the clothes string retains the state
-for another reason, the tactical light/laser doesn't show its state in the deobfuscated string, but differences in the raw inventory are visible
-the weapon seems to retain the state of the tactical laser/light (both deobfuscated and raw)

Weapon items (aka items with attachments):
--next value by ";"
"7001:1:4_10002_-1_-1_-1_0_y_:"
-colt ID = 7001
-count = 1
-now, we already know there is an attachment separator string (this case, "_")
-we have previousely analyzed attachments and learned that the structure
"4_10002_-1_-1_-1_0_y_" (attachments)
"MODE-SIGHT-BARREL-TACTICAL-MAGAZINE-BULLET" (extract from the old format, reverse it and it will make sense)
-4 bullets in a swift magazine (swift mag ID = 10002)
-3 kinds of attachments, we have none
-weapon on safety (mode 0)

-that "y", it specifies wether the tactical light/laser is ON/OFF (further observations: "y/n"=ON/OFF)

example: swissgewehr with 21/30 nato mag, open circle rail, tac light off, no barrel on safety

```
                      "7000:1:21_10000_-1_-1_9001_0_n_:"

                      7000     :     1     :     21     _     10000     _     -1     _     -1
_     9001     _     0     _     n     _     :
                      item    split  cou   split  bull  attac   magaz   attac   tac   ...    bar
...    sight   ...   mode  ...  light  ...  split
                       ID      ter   nt      ter   ets   hment    ine     hment   tic
        rel                                                          laser                ter
                                                                            split   type
split   al
ON/OFF
                                                                             ter
        ter
```

Empty (no) items:

"-1:0::"

-easily interpretable: ID = -1, count = 0

SEPARATORS:

"_": attachment separator (low priority)

":": item values separator (normal priority)

";": master separator (high priority)

SAVE FORMATS:

```
        Skills:
                <available skill points>;<survival lvl>;<endurance lvl>;<sneakybeaky
lvl>;<marksman lvl>;<warrior lvl>;<outdoors lvl>;<craftsman lvl>;<immunity lvl>;

        Life:
                <health>;<100-hunger>;<100-thirst>;<100-disease>;<t/f (true/false)
bleeding>;<t/f (true/false) broken leg>;

        Position:
                <x>;<height (water is roughly 15)>;<z>;<angle>
                where angle = 0-359 (clockwise)
                                      0-north
                                      90-east
                                      180-south
                                      270-west
                map is roughly 2000x2000

        Vehicles:

<name>_<vehicleVariationIndex>:<health>:<gas>:<x>:<y>:<z>:<rx>:<ry>:<rz>:<R>:<G>:<B>:;

        Clothes:
                <shirt>;<pants>;<head>;<backpack>;<armor>;
                -1 means having no clothing on the specified slot
```

Vehicle Editor:
X,Y get converted to map coordinates.
Then Z is assigned the value of the Y coordinate. This is because the game uses the Left-Hand Coordinate system with positive Z toward North, and positive X toward East.
Y is height.

The editor uses Right Hand Coordinate system, with positive Z pointing into the screen. Positive Y pointing South, and Positive X pointing East.

**Code should be edited, changing Rz to Ry. Arrow is wrong.