


```
/* ***** */
string tstr = "ZZZZZ";
ulong x = CONVERSION.BASE.ToLong(tstr);
ulong sval = x*84589UL+45989UL;
Console.WriteLine("real big value: {0}",sval);
/* ***** */
```

As you quickly see, this is a one way mapping – i.e. not a symmetrical process.

If you want more control over the generation – rather you want a bit more randomness that is not as easily recognizable, consider a 28 or 32 bit linear feedback shift register (if 32, then consider taps at 32,7,5,3,2,1 and 0). Load the value from the base 36 representation as the initial condition, and then shift out as much precision as you need (64 shifts gives you the range of 0..long.MaxValue).

I don't know if this helps – but it is an interesting exercise.

regards
roy fine

"William Stacey [MVP]" <staceywREMOVE@mvps.org> wrote in message news:useetjTxEHA.1300@TK2MSFTNGP14.phx.gbl...

- > *Thanks Roy. Could you expand the range by also including lower case a–z in addition to uppercase A–Z?*
- > *If so, could you spoon feed me again with the updated logic if possible.*
- > *What I am looking to do is:*
- > *1) 5 base chars (0..9, a...z, A..Z) for count – max range up to long.MaxValue if possible. Right now max range of ZZZZZ is 60,466,175.*
- > *2) Take a hash of count + some secret string(s) using PasswordDeriveBytes and convert as many bytes as possible to an alpha base encoding for a max of "HHHH–HHHH–HHHC–CCCC" (five Count positions and 11 Hash positions) to get a Product key (e.g. MSs). I should be able to recalc the hash at the client using stripped out count and the shared secret to verify the calculated hash matches the hash in the Product Key supplied. May need to break the hash bytes into two longs (16 bytes) and maybe clear high order byte before the conversion to long so I can pass each long to the BaseXX converter to get the string. Right now I can do 7 bytes and be sure to stay in 4738381338321616895 range.*
- >
- > *I realize the secret is vulnerable, but think I can make it good enough for my needs as just an activation code. Anyway, hope above makes some sense.*
- > *Basically looking to encode bigger ranges (max longs or max ulongs, or doubles if possible) with as few chars as possible in the valid set. Many thanks again. Cheers.*

>
> --
> William Stacey, MVP
> <http://mvp.support.microsoft.com>
>
> "Roy Fine" <rlfine@twf.obfuscate.net> wrote in message
> news:u79ypMTxEHA.1396@tk2msftngp13.phx.gbl...
>
>> "Justin Rogers" <Justin@games4dotnet.com> wrote in message
>> news:eTwFu0SxEHA.3908@TK2MSFTNGP12.phx.gbl...
>>> Ah, I wasn't aware alerting the author about changes to the code
>>> that you've made is a string being attached. The licensing agreement
>>> at the top is primarily a joke for all those that read it. It even
>> requires
>>> that you laugh... I put it there to reduce my own liability and to
point
>>> out that the software isn't supported. Take it for what you will.
>>>
>>
>> what the "recommended interpretation" of :
>>
>> <quote>
>> The use of this software is for test and performance purposes only.
>>
>> <\quote>
>>
>> and
>>
>> <quote>
>> In all seriousness,
>> excluding the laughter, laughter in itself does not void this license
>> agreement, nor compromise it's ability to legally bind you.
>> <\quote>
>>
>>>
>>> --
>>> Justin Rogers
>>> DigiTec Web Consultants, LLC.
>>> Blog: http://weblogs.asp.net/justin_rogers
>>>
>>> "Roy Fine" <rlfine@twf.obfuscate.net> wrote in message
>>> news:%23sZN8bSxEHA.2196@TK2MSFTNGP14.phx.gbl...
>>>>
>>>> "Justin Rogers" <Justin@games4dotnet.com> wrote in message
>>>> news:uPrmBSxEHA.1264@TK2MSFTNGP12.phx.gbl...
>>>>> Apparently it whacked the link...
>>>>>
>>>>> http://weblogs.asp.net/justin_rogers/articles/253641.aspx
>>>>>
>>>>>
>>>>>

>>>> nice – but the code i posted works, begs for a bit of optimization,
> but
>>>> comes with **no** strings attached.
>>>>
>>>> rlf
>>>>
>>>>
>>>>
>>>>> --
>>>>> Justin Rogers
>>>>> DigiTec Web Consultants, LLC.
>>>>> Blog: http://weblogs.asp.net/justin_rogers
>>>>>
>>>>>
>>>>> "Justin Rogers" <Justin@games4dotnet.com> wrote in message
>>>>> news:Om9Pw9RxEHA.1260@TK2MSFTNGP12.phx.gbl...
>>>>>> Code-Only: Arbitrary alphabet encoding (aka BaseN encoding) for
> base2
>>>> through
>>>>> base36.
>>>>>>
>>>>>> The notes are extensive as to the direction the library may or
may
>> not
>>>> go
>>>>>> depending on what
>>>>>>> problems people are trying to solve. What I've realized is that
> there
>>>>> are a
>>>>>>> number of additional
>>>>>>> and interesting problems associated with alphabet encoding, such
as
>>>>>>> permutations, cyclic
>>>>>>> rotations, error correction, and the like that may be interesting
> to
>>>>> build
>>>>>>> into the libraries. An
>>>>>>> example of an error correction alphabet would be the base32
> encoding
>>>>> which
>>>>>>> removes
>>>>>>> characters that may be confused for other characters when read by
a
>>>>> human.
>>>>>>>
>>>>>>>
>>>>>>> --
>>>>>>> Justin Rogers
>>>>>>> DigiTec Web Consultants, LLC.
>>>>>>> Blog: http://weblogs.asp.net/justin_rogers
>>>>>>>

>>>>>>
>>>>>>
>>>>>> "William Stacey [MVP]" <staceywREMOVE@mvps.org> wrote in message
>>>>>> news:u\$MPFPQxEHA.3908@TK2MSFTNGP12.phx.gbl...
>>>>>>> Hey thanks a lot Roy. Care to post the other base as well?
> Either
>>>> way,
>>>>>>> thanks again!!
>>>>>>>
>>>>>>> --
>>>>>>> William Stacey, MVP
>>>>>>> <http://mvp.support.microsoft.com>
>>>>>>>
>>>>>>> "Roy Fine" <rlfine@twt.obfuscate.net> wrote in message
>>>>>>>> news:#wPFSCQxEHA.3976@TK2MSFTNGP09.phx.gbl...
>>>>>>>> William,
>>>>>>>>
>>>>>>>>> this is something that i did some time ago – actually for a
>>>> different
>>>>>>>>> base,
>>>>>>>>>> but it was easy enough to change to handle base32.
>>>>>>>>>>
>>>>>>>>>>> you did not specify the symbol set for your number base – i
> will
>>>> assume
>>>>>>>>> 0,1,2,3... X,Y,Z. if yours is different, change the tokens
>> string
>>>>>>>>> accordingly.
>>>>>>>>>>
>>>>>>>>>>> for performance reasons, the weights of the digits are
computed
>> at
>>>>>>>>> compile
>>>>>>>>>>> time.
>>>>>>>>>>>
>>>>>>>>>>>> note – there is absolutely no error checking, and it handles
>> only
>>>>>>>>> positive
>>>>>>>>>>>> values, and assumes that all character codes are upper case.
>>>>>>>>>>>>
>>>>>>>>>>>>> regards
>>>>>>>>>>>>> roy fine
>>>>>>>>>>>>>
>>>>>>>>>>>>>
>>>>>>>>>>>>>> namespace CONVERSION{
>>>>>>>>>>>>>>> // handles positive only values up to
>>>>>>>>>>>>>>> 4,738,381,338,321,616,896 –
>>>>>>>>>>>>>>> I;
>>>>>>>>>>>>>>> public class BASE32{
>>>>>>>>>>>>>>>> static string tokens =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";

