Serial Key Generator with Expiry Date

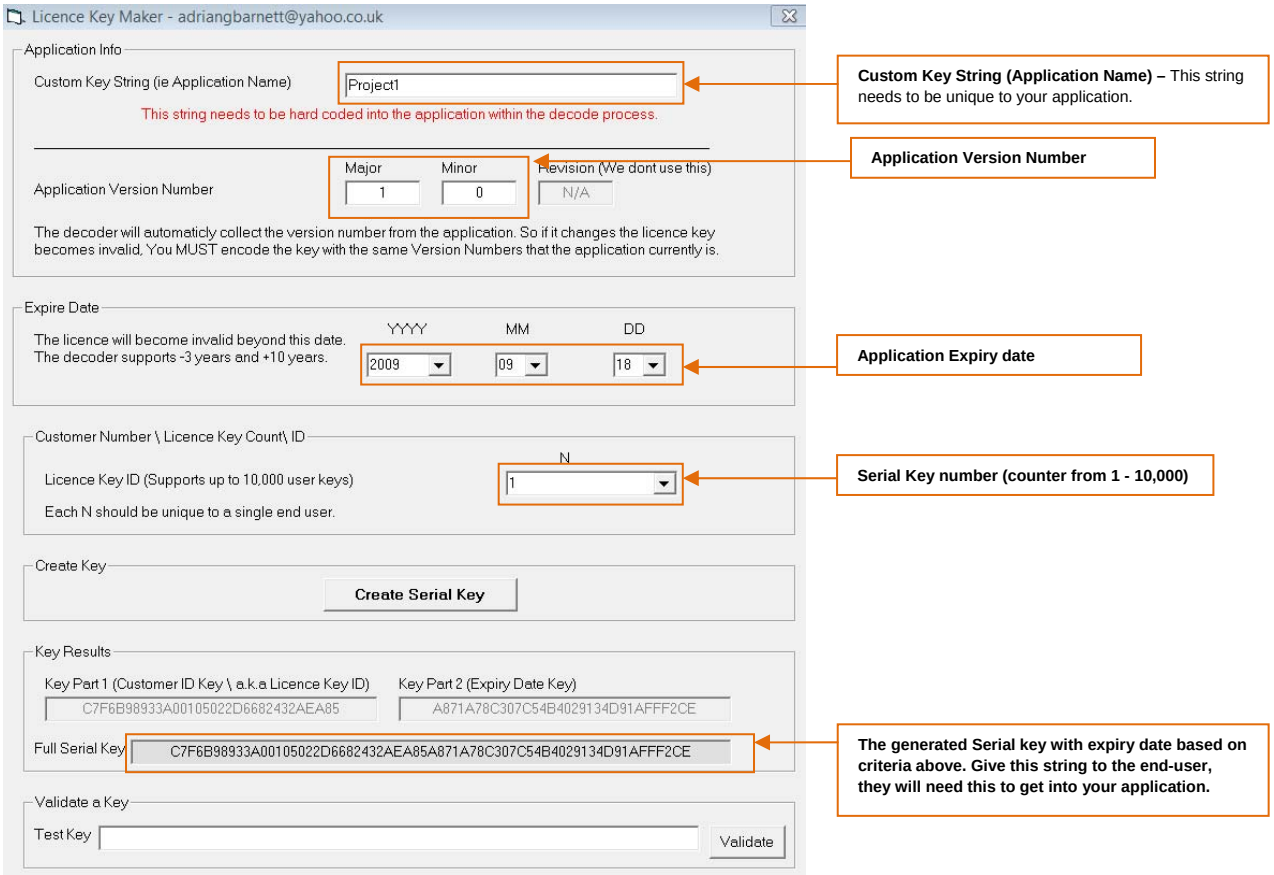
Created by adriangbarnett@yahoo.co.uk September 2009, (Experimental Code)

SUMMARY

This tool generates between 1 and 10,000 unique Serial key's 64 characters in length (512 bit) which is derived from MD5 digest. Within the encoded Serial key is an expiry date. The decoder determines if a key is valid and expired. The encoded MD5 hash key is non reversible, so to decode, the decoder needs to recreate the Serial key by sending specific strings to MD5, and then comparing the output hash against the Serial key (hash) that the end-user has submitted to unlock the application. Hopefully this demo will inspire some new ideas by others.

DEFINITIONS

User\Customer\End User – The end user who is aware of the Full Serial Key, the user of your application.
Full Serial Key\Serial Key – 62 character length hash key string.
CustomerHashKey – The first 32 characters of the Full Serial Key (does not contain any info relating to Expire key)
DateHashKey – The last 32 characters of the Full Serial key, contains the same data as *CustomerHashKey* and ALSO info about the expiry key.
CustomerNumber\LicenceID\Licence Number - The tool supports up to 10,000 individual licence keys.
Application\Your application – A programme that uses the Decoder to validate a licence key.



ENCODE

The Encoder works in 2 stages.

The first 32 characters of the 64 character Full Serial Key is the **CustomerHashKey**
The last 32 characters of the 64 character Full Serial Key is the **DateHashKey**

Encode Key Part 1 - CustomerHashKey

To generate this key (**CustomerHashKey**) we need:

1. **Custom Key String (Application Name) –** This string needs to be unique to your application, perhaps the application name, the string can be customized. What ever this string may be, the string needs to also be hard coded into the decoder function, which also obviously needs to be within your application to validate a key.

2. **Application Version Number -** This MUST be the exact version number of your application decoding the key. ***app.major*** **&** ***app.minor***. When decoder function within your application is run, it will take the ***app.major*** **&** ***app.minor***. I chose not to use ***app.revision*** in the encode\decoding; otherwise we need to make a key for every minor revision\patch.

3. **Licence Number** – The tool supports between 1 and 10,000 individual licence keys. Select a number between 1 and 10,000.

These strings ***Custom Key String*** **&** ***Application Version Number*** **&** ***Licence Number*** are merged together and sent to the MD5 hash Creator. i.e.: **MD5.CalculateMD5(*Custom Key String* & *Application Version Number* & *Licence Number)***. The output is the first 32 characters of the 64 character Serial Key which is referred to as the **CustomerHashKey**

Encode Key Part 2 – DateHashKey

To generate this key we need:

1. **Custom Key String (Application Name) –** Same as Encode Key part 1
2. **Application Version Number –** Same as Encode Key part 1
3. **Licence Number** – Same as Encode Key part 1
4. **End Year** – YYYY format, i.e. 2009.
5. **End Month** - MM format, i.e. 09
6. **End Day** - DD format, i.e. 23

These strings ***Custom Key String*** **&** ***Application Version Number*** **&** ***Licence Number*** **&** ***YYYY*** **&** ***MM*** **&** ***DD*** are merged together and sent to the MD5 hash creator. i.e.: MD5.CalculateMD5(***Custom Key String*** **&** ***Application Version Number*** **&** ***Licence Number*** **&** ***YYYYMMDD)***. The output is the last 32 characters of the 64 character Serial Key which we refer to as the **DateHashKey**

Now that **CustomerHashKey** and the **DateHashKey** has been generated, merge them bother together to form a **Full Serial Key**, and give this to the end user. **Full Serial Key =** ***CustomerHashKey*** **&** ***DateHashKey***.
Example Output: C7F6B98933A00105022D6682432AEA85A871A78C307C54B4029134D91AFFF2CE

DECODE

The Encoder works in 2 stages.

The end user sends there **Full Serial Key** (64 length hash key) to the decoder, example: the user types in "*C7F6B98933A00105022D6682432AEA85A871A78C307C54B4029134D91AFFF2CE*" and clicks Validate.

The first 32 characters of the 64 character Full Serial Key is the **CustomerHashKey**
The last 32 characters of the 64 character Full Serial Key is the **DateHashKey**

Decode Key Part 1 - CustomerHashKey

To decode we need to re-create the **CustomerHashKey** from scratch so that we can compare the MD5 hash to the first 32 characters of the **Full Serial Key** that the user submitted.

To generate this key we need:

1. **Custom Key String (Application Name) –** Same as the Encode string.
2. **Application Version Number -** This must be the exact version number of your application that and that is the same as used when we encoded the key in the first place (***app.major*** **&** ***app.minor*****).**
3. We need to attempt to FIND the **Licence Number** by creating a hash key for each Licence number between 1 and 10,000.
4. Once a licence number is generated (i.e. ***Licence Number*** **=** ***9***) we re-create a MD5 Hash key of (***Custom Key String*** **&** ***Application Version Number*** **&** ***Licence Number***), Example: **TestString1** = **MD5.CalculateMD5**(***Custom Key String*** **&** ***Application Version Number*** **&** ***Licence Number***
5. We then compare output of the MD5 hash key (**TestString1**) with the first 32 characters of the **Full Serial Key** the end user submitted, if there is a match, then we begin to check the expiry date key. We pass the ***Licence Number*** to the Decoder Part 2.
6. If we do not find a match between 1 and 10,000 then assume the key is not valid, and return an error code.

Decode Key Part 2 - DateHashKey

This function will only begin is the Decoder Part 1 is successful. To decode we need to re-create the **DateHashKey** from scratch so we can compare the MD5 hash to the last 32 characters of the **Full Serial Key** the user submitted.

To generate this key we need:

1. **Custom Key String (Application Name) –** Same as the Encode string.
2. **Application Version Number -** This must be the exact version number of your application that and that is the same as used when we encoded the key in the first place (***app.major*** **&** ***app.minor*****).**
3. **Licence Number –** this was passed to the **Decoder part 2 function** from the **Decode Key Part 1** function
4. We need to FIND a matching date, so we now start to generate every possible date within a 10 year range. The generated date string will be in the format of YYYYMMDD.
5. We then re-create a MD5 Hash key of (***Custom Key String*** **&** ***Application Version Number*** **&** ***Licence Number,*** **YYYYMMDD**), Example: **TestString2 = MD5.CalculateMD5**(***Custom Key String*** **&** ***Application Version Number*** **&** ***Licence Number,*** **YYYYMMDD**)
6. We then compare output of the MD5 hash key (**TestString2**) with the last 32 characters of the **Full Serial Key** the end user submitted, if there is a match, we can check if the date is valid\expired\about to expire. We return an error code based on the compare result of valid\expired\about to expire.
7. If there is no match with the after generating all possible combination of dates within a -5 years and +10 year range of the current date, then return an "Expired Key" error message. (It takes approximately 3 seconds to check 10,000 licence keys over a 15 year period).

CHANGING AN EXPIRY DATE FOR A LIVE REALEASED KEY

If you want to change an expiry date for an existing licence key that has been released to the end user, as long as certain special strings that were used to encode the original Full Serial remain the same, such as (***Custom Key String*** **&** ***Application Version Number*** **&** ***Licence Number***), you only need to send the end user the last 32 characters of the newly generated Full Serial. The first 32 characters will not have changed; the Expiry Date is saved only in the last 32 characters of the Full Serial Key – or – if you prefer send the entire new key serial.

THE DECODER FUNCTION

Self explanatory, but just in-case. The decoder module needs to be built into your application that is used to validate licence keys.

DECODER RETURN VALUES

The decoder will return a numeric value depending on the status of the serial key it has tried to validate.

Return Code from Decoder	Explanation of return Value	Return value from Function ***ValLic()*** back to your application
0	Invalid Key – The decoder could not match the serial key.	1
1	Critical Error in Function DecodeLicKey() . We should never see this unless there is a problem with the data sent to the function.	1
2	Critical Error in Function DecodeDateKey () . We should never see this unless there is a problem with the data sent to the function.	1
3	The first part of the Serial key is Valid, but the Second part (Expiry Date) is not valid.	1
4	The Key is Valid and the Key has not expired.	0
5	The key is Valid and the Key will expire today.	0
6	The key is Valid, but the key has expired.	1
Anything else	The Decoder returned something we didn't expect, no idea what or why.	1