

# Serial Key Maker API

## Introduction

This document describes the various methods and properties on the Serial Key Maker API. It also shows best practice usage for implementing Serial Key Maker's licensing solution into your applications.

## The Code

To assist with this documentation, download and unzip the following project files:

### VS 2005

[http://www.serialkeymaker.com/downloads/samples/VS2005/SKM\\_API\\_Documentation.zip](http://www.serialkeymaker.com/downloads/samples/VS2005/SKM_API_Documentation.zip)

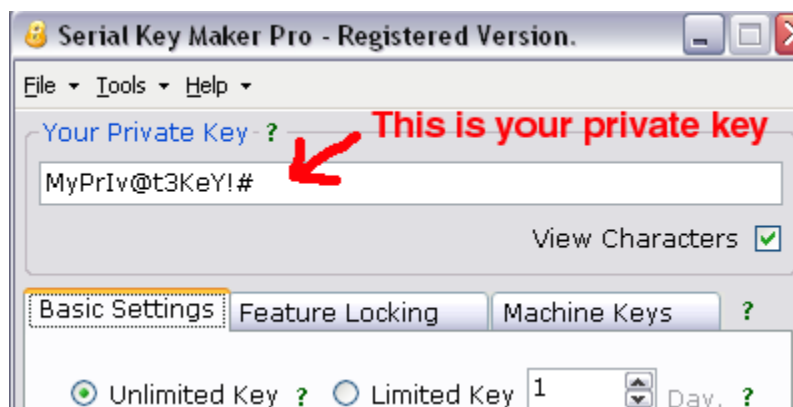
### VS 2008

[http://www.serialkeymaker.com/downloads/samples/VS2008/SKM\\_API\\_Documentation.zip](http://www.serialkeymaker.com/downloads/samples/VS2008/SKM_API_Documentation.zip)

## Creating a License Object (PG.SerialKeyMaker.Utility.API.classLicense)

In order to use the Serial Key Maker API, you need to create a valid license object. To do this you need the **private password key** that was used to create the license keys you want to test.

Recall, you create a license key with the Serial Key Maker user interface, you had to enter a *private password key*.



The Private Password Key needs to travel *with* your application so that the API can use it to decrypt the License Key. It should therefore be *encrypted* so that it cannot be easily read. Store it with your app, or in the registry, or in a file that your application can read.

You can **optionally** pass in a logging object that will handle internal error logging for you. You can pass in `Nothing` or `null` if you wish. The logging object is discussed below.

### C#

```
PG.SerialKeyMaker.Utility.API.classLicense objLicenseObject;

// --- encrypt this when using a real application. DO NOT hard-code it
// like this
string strPrivateKey = "MyPrIv@t3KeY!#";

PG.SerialKeyMaker.Utility.API.
classLogging objLogging = null;
objLicenseObject = new
PG.SerialKeyMaker.Utility.API.classLicense(strPrivateKey, objLogging);
```

### VB.Net

```
Dim objLicenseObject As G.SerialKeyMaker.Utility.API.classLicense

'--- encrypt this when using a real application. DO NOT hard-code it
'like this
Dim strPrivateKey As String = "MyPrIv@t3KeY!#"

Dim objLogging As PG.SerialKeyMaker.Utility.API.classLogging = Nothing
objLicenseObject = New
PG.SerialKeyMaker.Utility.API.classLicense(strPrivateKey, objLogging)
```

## **classLogging (PG.SerialKeyMaker.Utility.API.classLogging)**

The logging object (PG.SerialKeyMaker.Utility.API.classLogging) is an internal object that captures and writes log messages for you.

Use is optional - you can pass in nothing to the licensing constructor if you do not wish to use it. If you do wish to use it, you must explicitly turn logging on - it is off by default.

To create a licensing object, you do the following:

### C#

```
PG.SerialKeyMaker.Utility.API.classLogging objLogging;

// --- create instance of the object, passing in the log path.
```

```
// --- if it does not exist it will be created
objLogging = new
PG.SerialKeyMaker.Utility.API.classLogging("c:\\dev\\skmlog\\skm_logging.txt");
```

### VB.Net

```
Dim objLogging As PG.SerialKeyMaker.Utility.API.classLogging
'--- create instance of the object, passing in the log path.
'--- if it does not exist it will be created
```

```
objLogging = New
PG.SerialKeyMaker.Utility.API.classLogging("c:\\dev\\skmlog\\skm_logging.txt")
```

## classLogging Properties

### LoggingIsEnabled

#### C#

```
// --- No logging will happen unless this is explicitly enabled
// --- DEFAULT: False
objLogging.LoggingIsEnabled = true;
```

#### VB.Net

```
'--- No logging will happen unless this is explicitly enabled
'--- DEFAULT: False
objLogging.LoggingIsEnabled = True
```

### ConsoleIsOn

#### C#

```
//--- .ConsoleIsOn: passes ALL messages to the console.
//--- This includes messages sent to the file.
//--- DEFAULT: False
objLogging.ConsoleIsOn = true;
```

#### VB.Net

```
'--- .ConsoleIsOn: passes ALL messages to the console.
'--- This includes messages sent to the file
'--- DEFAULT: False
objLogging.ConsoleIsOn = True
```

## UseHourlyLoggingFilename

### C#

```
//--- .UseHourlyLoggingFilename(): appends the
//--- hour to the log file name
//--- DEFAULT: True
//--- NOTE: Known Bug1:
When you disable hourly logging, it also turns
off appending the date
//--- Known Bug2:
There is no way to toggle only the date,
independent of the time
```

```
objLogging.UseHourlyLoggingFilename = false;
```

### VB.Net

```
'--- .UseHourlyLoggingFilename(): appends the
'--- hour to the log file name
'--- DEFAULT: True
'--- NOTE: Known Bug1:
When you disable hourly logging, it also turns
off appending the date
'--- Known Bug2:
There is no way to toggle only the date,
independent of the time
```

```
objLogging.UseHourlyLoggingFilename = False
```

## LogFileExtension

### C#

```
//--- .LogFileExtension: ReadOnly.
//--- Displays what the current extension is
objLogging.LogToFile("Log file extension is: " +
objLogging.LogFileExtension);
```

### VB.Net

```
'--- .LogFileExtension: ReadOnly.
'--- Displays what the current extension is
objLogging.LogToFile("Log file extension is: " &
objLogging.LogFileExtension)
```

## LogFileName

### C#

```
// --- .LogFileName: ReadOnly. Displays log file name  
objLogging.LogToFile("Log file name is: " + objLogging.LogFileName);
```

### VB.Net

```
'--- .LogFileName: ReadOnly. Displays log file name  
objLogging.LogToFile("Log file name is: " & objLogging.LogFileName)
```

## LogPathDirectory

### C#

```
// --- .LogPathDirectory: ReadOnly. Display current log folder  
objLogging.LogToFile("Log folder name is: " +  
objLogging.LogPathDirectory));
```

### VB.Net

```
'--- .LogPathDirectory: ReadOnly. Display current log folder  
objLogging.LogToFile("Log folder name is: " &  
objLogging.LogPathDirectory)
```

## WholeLogFilePathandName

### C#

```
// --- .WholeLogFilePathandName(): Display the entire file and path  
(including date/time if on)  
objLogging.LogToFile("WholeLogFilePathandName is " +  
objLogging.WholeLogFilePathandName));
```

### VB.Net

```
'--- .WholeLogFilePathandName(): Display the entire file and path  
(including date/time if on)  
objLogging.LogToFile("WholeLogFilePathandName is " &  
objLogging.WholeLogFilePathandName)
```

## LocalMachineCode

To use the MachineCode to lock your license key to one particular machine you need to code your application so that it presents the "LocalMachineCode" property to the user in an about box or something similar.

Your user then communicates their local machine code to you, and you then use the Serial Key Maker user interface to generate a license key specific to that machine code.

You cannot change the value of LocalMachineCode (it is readonly) as it is generated internally to the Serial Key Maker API. It uses a combination of the machine's MAC address, Video card serial number and hard drive partition serial number to generate a hash.

#### **C#**

```
Console.WriteLine("The local machine code is: " +  
p_objLicenseObject.LocalMachineCode);
```

#### **VB.Net**

```
Console.WriteLine("The local machine code is: " &  
p_objLicenseObject.LocalMachineCode)
```

## **classLogging Methods**

### **LogToConsole**

#### **C#**

```
// --- .LogToConsole: Writes to the console  
objLogging.LogToConsole(PG.SerialKeyMaker.Utility.API.classLogging.MessageTypes.e_in  
"This is a test.");
```

#### **VB.Net**

```
'--- .LogToConsole: Writes to the console  
objLogging.LogToConsole(PG.SerialKeyMaker.Utility.API.classLogging.MessageTypes.e_in  
"This is a test.")
```

### **LogFilePathandName**

#### **C#**

```
// --- .LogFilePathandName: WriteOnly. Change the logfile.  
// --- Needs to be the whole path.  
objLogging.LogFilepathandName = "c:\\dev\\skmlog\\skm_logging2.txt";
```

#### **VB.Net**

```
'--- .LogFilePathandName: WriteOnly. Change the logfile.  
'--- Needs to be the whole path.  
objLogging.LogFilepathandName = ("c:\dev\skmlog\skm_logging2.txt")
```

## DeleteLogFile

### C#

```
// --- .DeleteLogFile(): Deletes the current logfile  
// --- causes untrapped runtime error if file does not  
// --- exist, so trap errors
```

```
objLogging.DeleteLogFile();
```

### VB.Net

```
'--- .DeleteLogFile(): Deletes the current logfile  
'--- causes untrapped runtime error if file does not  
'--- exist, so trap errors
```

```
objLogging.DeleteLogFile()
```

## DeleteLogFolder

### C#

```
// --- .DeleteLogFolder(): Deletes the current logfile  
// --- causes untrapped runtime error if file does not  
// --- exist, so trap errors
```

```
objLogging.DeleteLogFolder();
```

### VB.Net

```
'--- .DeleteLogFolder(): Deletes the current logfile  
'--- causes untrapped runtime error if file does not  
'--- exist, so trap errors
```

```
objLogging.DeleteLogFolder()
```

## LogRuntimeError

### C#

```
// --- .LogRuntimeError(): Special Logging method to record a to // ---  
record a runtime error  
// --- Parameters: Source, Message, StackTrace
```

```
objLogging.LogRuntimeError("Source Here", "Error Message Here",  
"StackTrace Here");
```

### VB.Net

```
'--- .LogRuntimeError(): Special Logging method to record a to  
'--- record a runtime error  
'--- Parameters: Source, Message, StackTrace
```

```
objLogging.LogRuntimeError("Source Here", "Error Message Here",  
"StackTrace Here")
```

## classLogging Enumerations

Used for console messages when using the "LogToConsole" method.

```
objLogging.MessageTypes.e_intInfo()  
objLogging.MessageTypes.e_intError()  
objLogging.MessageTypes.e_intNone()  
objLogging.MessageTypes.e_intSuccess()
```

## ValidatedKey Object

The ValidatedKey object is the crux of the Serial Key Maker API. It contains the details of the license key supplied to the user, and contains all the data tied to a particular license key.

To create a validated key object, you need to (at minimum) pass in a Serial Key Maker generated license key, which is a 20 character string in the following format:

```
THCNT-WIRHO-MOAIF-ITRUS
```

(this is a key generated with the private key:MyPrIv@t3KeY!#, using the Serial Key Maker user interface)

### Constructors

There are two overloads to creating a ValidatedKey Object. One is for use without a Machine Code, and the other is for use with the machine code property to tie the license key to the user's machine.

*You will need a valid license object to make a ValidatedKey object.*

### C#

```
PG.SerialKeyMaker.Utility.API.ValidatedKey objValidatedKey;
```

```

//--- Overload 1
objValidatedKey = p_objLicenseObject.ValidateKey("THCNT-WIRHO-MOAIF-
ITRUS");

//--- Overload 2
objValidatedKey = p_objLicenseObject.ValidateKey("THCNT-WIRHO-MOAIF-
ITRUS", p_objLicenseObject.LocalMachineCode);

```

### VB.Net

```

Dim objValidatedKey As PG.SerialKeyMaker.Utility.API.ValidatedKey

'--- Overload 1
objValidatedKey = p_objLicenseObject.ValidateKey("THCNT-WIRHO-MOAIF-
ITRUS")

'--- Overload 2
objValidatedKey = p_objLicenseObject.ValidateKey("THCNT-WIRHO-MOAIF-
ITRUS", p_objLicenseObject.LocalMachineCode)

```

## ValidatedKey Properties

### DateCreated

#### C#

```

// --- .DateCreated: Display the date the license
// --- key was created
Console.WriteLine("Date Created: " + objValidatedKey.DateCreated);

```

#### VB.Net

```

'--- .DateCreated: Display the date the license
'--- key was created
Console.WriteLine("Date Created: " & objValidatedKey.DateCreated)

```

### DateValidThrough

#### C#

```

// --- .DateValidThrough: Displays the
// --- date the license key expires.
Console.WriteLine("Date Valid Through: " +
objValidatedKey.DateValidThrough);

// --- TIP: Use the "DateValidThrough" property to

```

```
// --- display to your user
// --- how many days are left of their demo.
// --- As in:
// --- Demo Days Left = (objValidatedKey.DateValidThrough - Today)

Console.WriteLine("Days left on your demo: " +
Microsoft.VisualBasic.DateAndTime.DateDiff(Microsoft.VisualBasic.DateInterval.Day,
Microsoft.VisualBasic.DateAndTime.Now,
objValidatedKey.DateValidThrough,Microsoft.VisualBasic.FirstDayOfWeek.Sunday,Microsoft
```

## VB.Net

```
'--- .DateValidThrough: Displays the
'--- date the license key expires.
```

```
Console.WriteLine("Date Valid Through: " &
objValidatedKey.DateValidThrough)
```

```
'--- TIP: Use the "DateValidThrough" property to display
'--- to your user how many days are left of their demo.
'--- As in:
'--- Demo Days Left = (objValidatedKey.DateValidThrough -
Today)
```

```
Console.WriteLine("Days left on your demo: " &
Microsoft.VisualBasic.DateAndTime.DateDiff(DateInterval.Day,
Microsoft.VisualBasic.DateAndTime.Now,
objValidatedKey.DateValidThrough).ToString)
```

## Expires

### C#

```
// --- .Expires: Displays whether the license key expires
// --- If it does expire, then you have a DEMO key
```

```
Console.WriteLine("Expires: " + objValidatedKey.Expires.ToString());
```

### VB.Net

```
'--- .Expires: Displays whether the license key expires
'--- If it does expire, then you have a DEMO key
Console.WriteLine("Expires: " & objValidatedKey.Expires.ToString)
```

## Features

### C#

```
// --- Features 1 through 5:
// --- A license key can embed which features are enable
```

```

// --- NOTE: Features are linear. That is, if "Feature 4" is
// --- enabled, then features 1 through 3 are enabled too.

Console.WriteLine("Feature 1 enabled: " +
objValidatedKey.Feature1.ToString());

Console.WriteLine("Feature 2 enabled: " +
objValidatedKey.Feature2.ToString());

Console.WriteLine("Feature 3 enabled: " +
objValidatedKey.Feature3.ToString());

Console.WriteLine("Feature 4 enabled: " +
objValidatedKey.Feature4.ToString());

Console.WriteLine("Feature 5 enabled: " +
objValidatedKey.Feature5.ToString());

```

## VB.Net

```

'--- Features 1 through 5:
'--- A license key can embed which features are enable
'--- NOTE: Features are linear. That is, if "Feature 4" is
'--- enabled, then features 1 through 3 are enabled too.

Console.WriteLine("Feature 1 enabled: " &
objValidatedKey.Feature1.ToString)

Console.WriteLine("Feature 2 enabled: " &
objValidatedKey.Feature2.ToString)

Console.WriteLine("Feature 3 enabled: " &
objValidatedKey.Feature3.ToString)

Console.WriteLine("Feature 4 enabled: " &
objValidatedKey.Feature4.ToString)

Console.WriteLine("Feature 5 enabled: " &
objValidatedKey.Feature5.ToString)

```

## FreeformText

You can embed free form text into the license key. If you do, the license key will NOT be a neat 25 character string, but will rather be an encrypted string that contains the text.

TIP: You typically supply this key in the form of a file - you cannot expect your user to type it in!

As an example, the following key has some free form text in it, in addition to locking 3 features and being a 300 day demo key:

### C#

```
string strLongLicenseKey =
    "PGieNwikouu28z54lJpOhsodkFZ917nMPw9gzb6uScxjH0M/
    NgxVS8rWlvpwRI1enl3hpuQizsZWYqA95FHlYlC1+1rNaCMaJmmqv+m2/
    K9ksSCr+55ftFw3Lc0nxDPZhn2E+rQxO/pmJNxBoLDJ6Q==";

objValidatedKey = p_objLicenseObject.ValidateKey(strLongLicenseKey);

Console.WriteLine(objValidatedKey.FreeformText);
```

### VB.Net

```
Dim strLongLicenseKey As String =
    "PGieNwikouu28z54lJpOhsodkFZ917nMPw9gzb6uScxjH0M/
    NgxVS8rWlvpwRI1enl3hpuQizsZWYqA95FHlYlC1+1rNaCMaJmmqv+m2/
    K9ksSCr+55ftFw3Lc0nxDPZhn2E+rQxO/pmJNxBoLDJ6Q=="

objValidatedKey = p_objLicenseObject.ValidateKey(strLongLicenseKey)

Console.WriteLine(objValidatedKey.FreeformText)
```

## FreeformTextItems

If you separate individual items in your free form text with the "pipe" character (|), then the API will automatically split it into the FreeformTextItems array.

### C#

```
foreach (string itm in objValidatedKey.FreeformTextItems)
{
    intCounter++;
    Console.WriteLine(("Item "
    + (intCounter.ToString() + ": " + itm)));
}
```

### VB.Net

```
For Each itm As String In objValidatedKey.FreeformTextItems

    intCounter += 1
    Console.WriteLine("Item " & intCounter.ToString & ": " & itm)
```

Next

## IsCurrentlyValid

Is the key you tested valid?

### C#

```
if (objValidatedKey.IsCurrentlyValid)
{
    Console.WriteLine("The license key IS valid.");
}
else
{
    Console.WriteLine("The license key is NOT valid.");
}
```

### VB.Net

```
If objValidatedKey.IsCurrentlyValid Then

    Console.WriteLine("The license key IS valid.")

Else

    Console.WriteLine("The license key is NOT valid.")
End If
```

## Key

### C#

```
// --- .Key: String - Echo's back the current
// --- key that was tested
// --- This string does not contain the free form text items etc
Console.WriteLine("The current license key is: " +
objValidatedKey.Key);
```

### VB.Net

```
'--- .Key: String - Echo's back the current
'--- key that was tested
'--- This string does not contain the free form text items etc

Console.WriteLine("The current license key is: " &
objValidatedKey.Key)
```

## MachineCode

### C#

```
// --- .MachineCode: String - The local machine's Machine Code,  
// --- This is the same value as the LocalMachineCode  
// --- property of the classLicense object  
Console.WriteLine(("The MachineCode key is: "  
+ (objValidatedKey.MachineCode + (" and is the same as the  
LocalMachineCode: " + p_objLicenseObject.LocalMachineCode))));
```

### VB.Net

```
'--- .MachineCode: String - The local machine's Machine Code,  
'--- This is the same value as the LocalMachineCode  
'--- property of the classLicense object  
  
Console.WriteLine("The MachineCode key is: " &  
objValidatedKey.MachineCode & " and is the same as the  
LocalMachineCode: " & p_objLicenseObject.LocalMachineCode)
```

## MachineCodeValidates

### C#

```
// --- .MachineCodeValidates: Boolean - Did the local machine  
// --- code validate?  
// --- Will return true if no machine code is used.  
if (objValidatedKey.MachineCodeValidates)  
{  
    Console.WriteLine("The machine code DOES validate.");  
}  
else  
{  
    Console.WriteLine("The machine code DOES NOT validate.");  
}
```

### VB.Net

```
'--- .MachineCodeValidates: Boolean - Did the local machine  
'--- code validate?  
'--- Will return true if no machine code is used.  
If objValidatedKey.MachineCodeValidates Then  
  
    Console.WriteLine("The machine code DOES validate.")  
  
Else
```

```
        Console.WriteLine("The machine code DOES NOT validate.")

    End If
```

## PublicKeyValidates

### C#

```
// --- .PublicKeyValidates: Boolean
// --- Did the public key validate
if (objValidatedKey.PublicKeyValidates)
{
    Console.WriteLine("The public key DOES validate.");
}
else
{
    Console.WriteLine("The public key DOES NOT validate.");
}
```

### VB.Net

```
'--- .PublicKeyValidates: Boolean
'--- Did the public key validate
If objValidatedKey.PublicKeyValidates Then

    Console.WriteLine("The public key DOES validate.")

Else

    Console.WriteLine("The public key DOES NOT validate.")

End If
```

## ValidVersion

### C#

```
// --- .ValidVersion: String - Reports what the this
// --- version of the API valid for this key.
Console.WriteLine("ValidVersion is " +
objValidatedKey.ValidVersion);
```

### VB.Net

```
'--- .ValidVersion: String - Reports what the this
'--- version of the API valid for this key.
Console.WriteLine("ValidVersion is " & objValidatedKey.ValidVersion)
```

## More Code

For more code samples visit:

<http://www.serialkeymaker.com/download.htm>