# Key Generator SDK

Version 3.1 / August 2004

by Stefan Weber

***IMPORTANT:*** *Before you continue reading, please make sure that you have the most recent version of this document from the "Start Page/Documentation" section of your control panel account.*

## Contents

# 1 Overview

This document describes the interface between our order processing system and your individual key generators. It defines a set of rules that you should follow closely to ensure proper operation of your key generator with our license servers.

Here the term “key” refers to any type of information that is generated upon the successful completion of a sale and sent to your customers, this could be:

- a license serial number
- a file containing a cryptographic key that unlocks you application
- a HTML file including the key as well as additional registration information
- whatever you can put into a file …

The Key Generator SDK (Software Developers Kit) comprises of

- this documentation,
- code examples in a number of different programming languages and
- a test program that calls your key generator and displays its results.

***NOTE:*** *Please read this document carefully before writing and submitting a custom key generator. We will not accept new key generators that do not comply with the latest key generator specification.*

## 1.1 Supported Types of Key Generators

Your key generator can be implemented in one of several different ways:

- As a CGI program that is hosted on your server and called via HTTP/ HTTPS or
- as a program that runs on our servers:
  - Windows NT / 2000 / XP console application executables (EXE, compiled for the x86 32bit architecture).
  - JAVA programs (currently JRE Version 1.3 thru 1.4.2 supported)
  - .NET Framework Version 1.1 assemblies.

## 1.2 Supported Types of Keys

For historical reasons this SDK supports two different types of keys.

- **Textual keys** are made of plain text (e.g. “ABCD-1234-WXYZ”). They will be displayed inline inside the delivery email that is sent to your customer. Optionally you can provide a key template that surrounds your key with additional information. (see section 5 “Delivery e-mail Key Templates”).
- **Binary keys** can include anything and are forwarded to the customer as an attachment to the delivery email. Their content is described by a MIME media type (e.g. text/xml) and a filename that has to be provided by your generator. Obviously binary keys are a superset of textual keys.
  (see http://www.iana.org/assignments/media-types/)

***CONCLUSION:*** *The textual key generator API is no longer state of the art. Therefore you should write a binary key generator.*

# 2 API for non-CGI key generator programs

## 2.1 The Key Generation Process

Before going into the details of the key generator interface, here is an overview of the steps involved in the key generation process:

1. Order processing system calls key server
    a. Key server writes input values to a temporary file (see section 4)
    b. Key server executes the key generator:
        i. Key generator reads input file
        ii. Key generator decodes & checks input parameters
        iii. Key generator calculates license key(s)
        iv. Key generator writes output files
        v. Key generator returns exit code
    c. Key server checks exit code and reads output files
    d. Key server checks if a key template is available for textual keys and inserts the generated key into the template.
    e. Key server saves output files / error log into the database
2. Order processing prepares the “delivery to customer” email,
    a. binary keys will be attached with the MIME type and filename returned by the key generator
    b. textual keys will be displayed inline

## 2.2 Command line arguments, Input/ Output files

EXE and JAVA generators are called with three command line arguments, all of which are file names. e.g.

```
KeyGen.exe c:\temp\input-12653.txt c:\temp\input-12653.out1
c:\temp\input-12653.out2
```

- First is the input file to be read by the key generator.
- The 2nd and 3rd parameter are output filenames which don’t exist upon invocation of the key generator, since they should be created by the key generator itself. Output filenames are provided by the calling server in order to be able to set their directory, make their names unique, read their contents and delete them after their contents has been copied to the database.

## 2.3 Input File Format

The input file format is somewhat similar to Windows INI file sections. Each line contains a key-value-pair separated by a '='. The values are terminated at the end of line (CR/LF), and may contain non-ASCII characters depending on the input encoding. e.g.

```
PURCHASE_ID=12345678
RUNNING_NO=1
PURCHASE_DATE=04/07/2004
PRODUCT_ID=100000
LANGUAGE_ID=2
QUANTITY=1
REG_NAME=Peter "Test" Müller
LASTNAME=Müller
FIRSTNAME=Peter
EMAIL=mueller@test.net
```

When parsing the lines, all unknown keys should be ignored since there may be new input parameters in the future. There is no guarantee that the lines appear in a defined order, except for the encoding parameter, which is always first – if present.

**Character Encoding**

The encoding of the in- and output files defaults to the ISO-8859-1 (Latin 1) encoding. If the file starts with the line "ENCODING=UTF8" it will be UTF8 encoded. Here an example printed without UTF8 decoding in order to show the difference:

```
ENCODING=UTF8
PURCHASE_ID=12345678
RUNNING_NO=1
PURCHASE_DATE=23/07/2004
PRODUCT_ID=100000
LANGUAGE_ID=2
QUANTITY=1
REG_NAME=Peter "Test" MÃ¼ller
LASTNAME=MÃ¼ller
FIRSTNAME=Peter
EMAIL=mueller@test.net
```

The file written by the key server will not include any byte order marks for UTF8.

For a description of the available input parameters see section 5.

***NOTE:*** *In order for UTF8 input to be sent to your key generator it has to be registered as a Unicode compliant SDK 3 key generator with our support team upon submission / installation of the generator. Otherwise your generator will always get ISO-8859-1 input and non-ISO-8859-1 characters will be replaced by question marks.*

***NOTE****: The encoding of emails generated by the order processing system depends on the actual content of the mail body. If your textual key has characters that do not belong to (US-ASCII or ISO-8859-1) it will use the UTF8 encoding. Japanese mails will be encoded using ISO-2022-JP, if possible. Other encodings may be added to support other languages in the future. Since not all customers use UTF8 enabled mail clients, textual keys should only include characters that are common in their language.*

## 2.4 Exit Codes

The generator must provide an exit code, which tells the key server about the success or the reason for failure. See the following table for all predefined exit code values. Please do not return other values or change the numbering scheme provided in the SDK examples!

Exceptions should be caught in the key generator main function and mapped to these error codes. Exception messages should be written to the first output file.

| | |
|---|---|
| ERC_SUCCESS | no error (a valid textual key has been generated) |
| ERC_SUCCESS_BIN | no error (a valid binary key has been generated) |
| ERC_ERROR | general error or runtime exception |
| ERC_MEMORY | memory allocation failed |
| ERC_FILE_IO | error during file i/o (map I/O exceptions to this error code) |
| ERC_BAD_ARGS | missing, incomplete or wrong command line arguments |
| ERC_BAD_INPUT | missing or incomplete input values or encoding |
| ERC_EXPIRED | key generator has expired due to a limited validity period |
| ERC_INTERNAL | other internal errors |

## 2.5 Output Files

The data that should be written to the output files depends on success or failure of the key generation and the type of the key generator. Their interpretation by the key server is determined by the exit code.

On *failure* the error message should be provided in the first result file. The second file can be omitted in this case. (e.g. exit code ERC_BAD_INPUT and output file #1 contains: "Value for REG_NAME must contain at least 8 characters!").

On *success* the output file contents depends on the type of the key generator:

**Textual Keys = ERC_SUCCESS**

The first output file is interpreted as the **User Key** which will be delivered to the customer.

The second output file is interpreted as the so-called **CCKey** (CC for “carbon copy”), and stored for your reference. You can view it in the sales section of the Control Panel, or in the export file. By creating a separate CCKey you can decide which information is reported to you. (e.g. probably your are only interested in the key itself, but not in the instruction text surrounding it.)

**IMPORTANT:** The file has to be encoded using the same encoding as the input file, using CR/LF for line breaks.

*NOTE: You may want to generate a short usage description together with the key. If you would like to provide translations of this text for different languages, you can use the LANGUAGE input value. Anyhow, you should have a look at the key template feature before using this feature as it provides a more convenient way of text customization.*

**Binary Keys = ERC_SUCCESS_BIN**

The first file is interpreted as a content description of the data that is provided in the second file. It comprises of the contents MIME type and the desired filename separated by a single colon. For example:

```
application/octet-stream:key.bin
```

The actual binary key data has to be returned in the second output.

Keep your binary keys small in order to speed up mail delivery. Many email accounts have size restrictions for incoming mail.

***NOTE:** For binary keys there is no carbon copy key feature, but the XML order notification contains a copy of the binary key file.*

# 3 CGI Key Generators

Web based key generators reside on your server and will be called by our order processing system whenever a key for one of your customers needs to be generated.

You'll have to provide a URL which can be called by our system with the input values passed as form data via a HTTP POST request. Your server is meant to return the customer's key upon such a request.

We support secure HTTP as well, so you can provide us with a URL starting with 'https://' as well. We do not require a server certificate from an independent certification authority like Verisign Inc. Therefore you are free to generate your own certificate for that purpose.

e.g. https://keygen.yourcompany.com/cgi-bin/keygen.exe

Input Values will be passed as CGI parameters. e.g.

```
https://keygen.yourcompany.com/cgi-bin/keygen.exe?PURCHASE_ID=0&RUNN
ING_NO=1&PURCHASE_DATE=16%2F08%2F2004&PRODUCT_ID=100000&QUANTITY=1&R
EG_NAME=Peter+%22Test%22+M%C3%BCller&ADDITIONAL1=&ADDITIONAL2=&RESEL
LER=&LASTNAME=M%C3%BCller&FIRSTNAME=Peter&COMPANY=&EMAIL=mueller@tes
t.net&PHONE=&FAX=&STREET=&CITY=&ZIP=&STATE=&COUNTRY=&LANGUAGE_ID=2
```

### Failure

Errors are indicated by status codes other than 200. We recommend that you use "400 Bad Request". The content of a response that leads to an error may only consist of one line that might help you and us to locate any problem.

For a status code of 200 your key generator scripts must return the key as the content of the HTTP response.

### Success

When "200 OK" is returned, the output is expected to be a valid key and is passed to the customer.

### Content Disposition and Filename

You can decide whether to display your key inline (as part of an email or web page) or as a binary file (as a mail attachment or as a download link).

In order to display your key inline you have to set the MIME Content-Type header of your response to 'text/plain' (s. examples).

Any other MIME Content-Type will result in an attachment / download with your requested mime type (s. examples).

If you choose not to use 'text/plain' you have to provide us with a file name. This file name will then be used as the file name for the download and the mail attachment. The file name is set by sending a Content-Disposition header with your response.

Please set it as follows: 'Content-Disposition: filename=<*filename.ext*>' We allow the following characters for filenames: a-z, A-Z, 0-9, !_.%$~#-:äöüÄÖÜß@^ (Please note that no blanks are allowed.)

### Security Considerations

For security reasons you should limit access to this URL to IP addresses from the element 5 network 217.65.128.x. Please note that this address space might change in the future. In this case you will be asked to adjust your web server configuration.

# 4 Input Values

The key server will collect all available data for the following fields and forward them to your key generator as input. Fields listed in **bold** always contain valid information. Other fields may be left out, if a particular piece of information is neither applicable nor available.

| *FIELD KEY (TAG)* | *SIZE* | *DESCRIPTION* |
|---|---|---|
| **PURCHASE_ID** | 10 | Our purchase id number. (1st part of the unique key)<br>*Type: decimal string representation of an unsigned 32bit integer number > 0.* |
| **RUNNING_NO** | 4 | If the customer buys multiple products in one purchase they will be enumerated by RUNNING_NO. (2nd part of the unique key. The combination of PURCHASE_ID and RUNNING_NO is guaranteed to be unique!)<br>*Type: decimal string representation in the range 1..9999* |
| **PURCHASE_DATE** | 10 | The date of purchase as string "DD/MM/YYYY", e.g. "24/12/2000" |
| **PRODUCT_ID** | 10 | ID of the product purchased. (Useful if you write a key generator that works for multiple products.)<br>*Type: decimal string representation* |
| **QUANTITY** | 4 | The number of copies ordered. *Type: decimal string representation of an unsigned 32bit integer number > 0.* |
| **REG_NAME** | 100 | The name to which the customer chose to license the product. |
| ADDITIONAL1 | 500 | 1st Customizable Field (see below). |
| ADDITIONAL2 | 500 | 2nd Customizable Field (see below). |
| RESELLER | 100 | The name of the reseller involved in this order (if applicable). |
| LASTNAME | 50 | The customer's last name. |
| FIRSTNAME | 50 | The customer's first name. |
| COMPANY | 100 | The customer's company name (if applicable). |
| EMAIL | 100 | The customer's e-mail. |
| PHONE | 50 | The customer's telephone number. |
| FAX | 50 | The customer's fax number. |
| STREET | 100 | The customer’s street address. |
| CITY | 100 | The customer's city. |
| ZIP | 20 | The customer’s ZIP or Postal Code. |
| STATE | 40 | The customer’s state or province (e.G. Florida)<br>(US / Canadian customers only) |
| COUNTRY | 50 | The customer's country (English name) |
| **LANGUAGE_ID** | 2 | The customer's preferred language (see table below) |
| ENCODING | var. | <not present> = ISO-8859-1 (Latin 1) encoding<br>“UTF8” = UTF8 Unicode |

| | |
|---|---|
| 1 | English |
| 2 | German |
| 3 | Portuguese |
| 4 | Spanish |
| 5 | Italian |
| 6 | French |

| | |
|---|---|
| 7 | Norwegian |
| 8 | Dutch |
| 9 | Swedish |
| 10 | Finnish |
| 11 | Japanese |
| 12 | Russian |

You do not, of course, have to use all of these values to generate the key. Which, and how many of these variables you use, is left to your discretion.

**Customizable Fields**

In addition to our fixed set of fields, we offer two customizable fields named “ADDITIONAL1” and “ADDITIONAL2”. These additional input values are optional and can be added to your order form by entering them in the Products/Product Delivery/Descriptions section of your Control Panel account.

Provide a short description of the type of information that you would like your customers to enter in these fields. The checkboxes on the right side allow you to make a field required/mandatory (checked) or optional (not checked). If your generator relies on these additional input values, don't forget to make the appropriate changes.

*Note: There is currently no way to validate additional input fields during the order process. We are currently working on a solution based on JavaScript.*

# 5 Delivery e-mail Key Templates

A convenient and flexible way to provide additional information with your key is the key template feature: The generated textual key will be embedded into a prepared text template for delivery. This feature enables you to change the texts without submitting a new key generator.

You can edit these templates in the Products/Descriptions section of your control panel account.

In order to use this feature you should write texts for English and optionally other languages which must contain "`<%KEY%>`" as a placeholder for the actual User Key. The key server will then substitute the key for the first occurrence of the placeholder.

The templates cannot be activated unless the proper key generator has been installed. If you would like to use this feature please notify our signup team to activate the templates upon installation of the key generator that belongs to the templates.

# 6 General Rules and Prerequisites

- Always remember that our key server is automated. Therefore you should handle all possible situations and exceptions properly. Do not use any user interaction. Return a meaningful exit code and error descriptions in case of failure.
- Some compilers display message boxes (or other dialogs) when exception or fatal errors occur. Please refer to your compilers documentation to see whether this applies to it, and how to override them.
- Your generator should be installable by copying its files into a single network folder (XCopy Installation). Please do not send installer/setup programs, as our support team has no administrator rights on the key server. Do not send COM servers that require registration.
- Access all auxiliary files via the current directory, which is set to your generators location before it gets invoked. If you need to store persistent information for your key generator you must use a file in the current directory.
- The generators are executed with a time limit of 5000 milliseconds. If the generator takes longer, it will be terminated and thus fail!

# 7 How to submit a key generator

Prior to submitting a new key generator, please run several checks on your code and make sure it is stable. You should always use the test environment provided in the SDK. Try different input values or leave them out to test your error handling.

Please include answers to the following questions with your key generator:

- Which Language / Compiler did you use for the creation of this key generator?
- Which Version of this Specification did you refer to when writing the code? (see beginning of this document)
- Which auxiliary files (DLL's, Runtime Environment, etc.) are required for your generator to work?
- Which temporary and/or persistent files are created by the generator?
- Are there any limitations regarding input values?
- Is the generator limited to work only in a predefined period of time? Does it create only a limited number of keys? Do not forget to return ERC_EXPIRED if this is the case.
- Is there anything else we need to know?

# 8 F.A.Q.

*Q: I was wondering if I could call an internet URL from within the keygen when it runs so I can send a cgi script on my web page some data with POST. Does the generator have access to the internet when running?*

A: The key server is situated behind a firewall and has no access to the internet. You should consider setting up a CGI based key server.

*Q: I know that I could use the 'Additional Field' in keygen to request the computer name, but if a customer is purchasing multiple copies, can you request the 'Additional Field' to be entered for each copy?*

A: No. In this case you'll have to use comma separated values or a similar technique.

*Q: I want to have certain fields validated, before payment is processed. If fields are incorrect, the user is asked to re-enter correct data (e.g. just like when they enter an invalid card number). I cannot see any way to do this with the Control Panel, is it possible?*

A: We’re currently planning a verification mechanism using JavaScript on the order forms.

*Q: I Would like to use product xyz for key generation. Is it supported by your ordering system?*

A: If you can wrap it in a way that meets our specifications, then the answer is yes.

*Q: What do you do if the returned value is not ERC_SUCCESS, but some kind of error (for example ERC_INTERNAL)?*

A: The purchase is added to a key generation failure list which is processed manually from time to time. Web based key generators will be retried automatically up to 3 times. (e.g. if the server was down)

# 9 Examples / Templates

**Borland Delphi**

The Borland Delphi example provides a convenient abstract base class that implements the key generator API. You can create your own generator by descending from the base class.

The example can handle ISO-8859-1 and UTF8 input transparently and provides code for textual as well as binary keys.

The `Value()` function is provided for easy access to the input values. Remember to select "Generate console application" in the Project Options / Linker dialog.

| *FILENAME* | *DESCRIPTION* |
|---|---|
| Example.dpr | project file |
| KeyIntf.pas | abstract interface classes (don't touch) |
| KeyUser.pas | key generator (start here with your implementation) |

**C#**

The C# example implements all possible key generator features. The type of key depends on the BINARY_GEN define. The supported encoding on the UNICODE_GEN define.

| *FILENAME* | *DESCRIPTION* |
|---|---|
| Main.cs | key generator class |
| AssemblyInfo.cs | meta information |
| make.bat | a simple make batch for the csc.exe command line compiler that also includes a test run using the provided examples |

**JAVA**

Since the standard JAVA main function is declared `void`, we have added a new main function `public static final int KeyMain(String args[])` which has to be implemented in every JAVA key generator. This function gets called via the java native interface by our key server.

Our examples were developed using SUN's Java SDK 1.4.

JAVA requires that the name of a class and the filename of its implementation match. Therefore you'll have to rename the example and the class to make it distinguishable from the classes provided by other programmers. Always use the naming scheme "P" followed by your program number. If you don't have a program number please contact our signup team.

Our key server currently runs JRE 1.3 but will be updated to JRE 1.4 soon.

| *FILENAME* | *DESCRIPTION* |
|---|---|
| P100000.java | key generator class |
| make.bat | a simple make batch for the javac.exe command line compiler that also includes a test run using the provided examples and javadoc extraction |

**Visual Basic .NET**

The VB.NET example implements all possible key generator features. The type of key depends on the BINARY_GEN define. The supported encoding on the UNICODE_GEN define.

| *FILENAME* | *DESCRIPTION* |
|---|---|
| KeyGen.vb | key generator class |
| make.bat | a simple make batch for the vbc.exe command line compiler that also includes a test run using the provided examples |

**Visual Basic 4 thru 6**

*Due to the limitations of these Visual Basic versions this example has some flaws. You should only use it, if you cannot provide a generator written in one of the other languages.*

Since Visual Basic does not include support for the setting of exit codes, we had to import the `ExitProcess()` function from KERNEL32.DLL to get them working. This technique is described in Microsoft's knowledge base (Q178357), you'll find a copy of this article in "vba-error-level.htm". One pitfall with this implementation is that `ExitProcess()` must be called as the last function in `Form_Terminate()` which in turn is executed when you Unload the form using `Unload Me`.

The other problem is that `Form_Terminate()` will unload the Visual Basic IDE as well. Therefore you should remove this call temporarily for development and debugging. Don't forget to put it in the final release. Test all error conditions before submitting a key generator.

The form is required for this to work properly but has no further use in key generators. You should not add any controls to it or make it visible.

| *FILENAME* | *DESCRIPTION* |
|---|---|
| KeyGenVB.vbp | project file |
| FormKey.frm | key generator (start here with your implementation) |