

Key Generator SDK

Version 2.7 / June 2001

Contents

1. Overview.....	1
2. Input Values.....	2
3. Output Values.....	3
Textual Keys.....	3
Delivery e-mail templates.....	4
Binary Keys.....	4
4. API for EXE and JAVA generators.....	4
5. API for DLL generators.....	5
6. CGI Key Generators.....	5
7. General Rules and Prerequisites.....	6
8. How to submit a key generator.....	7
9. F.A.Q.....	7
10. Examples / Templates.....	8
C/C++.....	8
Borland Delphi EXE.....	8
Borland Delphi EXE for Binary Keys.....	8
Borland Delphi DLL.....	8
Visual Basic 4 and higher.....	8
JAVA.....	9

1. Overview

This key generator SDK (Software Developers Kit) describes the interface between our order processing system and your individual key generators. These generators allow the use of customized key generation algorithms inside our automated order process. You can implement the key generator in your preferred programming language.

Upon the successful completion of a sale, our server will build a list of input values that will be fed into your key generator. Your key generator then picks the values it needs and uses them to run a key generation algorithm. The outputted key will be sent to the customer to activate your product, as well as be sent to you by e-mail and/or be displayed in the Control Panel.

This SDK defines a set of rules that you should follow closely to ensure proper operation of your key generator on a permanent, continuous basis. Please read this document carefully before writing a custom key generator.

Your key generator can be implemented as a DLL, a console application (EXE) or in a JAVA class. While JAVA is almost platform independent, the binary executables must be compiled for the Windows NT 4 API. Included in this document are complete example templates for the most popular programming languages.

We will no longer accept new key generators that do not comply with this specification (or a later version).

2. 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. (1 st part of the unique key)
RUNNING_NO	4	If the customer buys multiple products in one purchase they will be enumerated by RUNNING_NO. (2 nd part of the unique key. The combination of PURCHASE_ID and RUNNING_NO is guaranteed to be unique!)
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.)
QUANTITY	4	The number of copies ordered.
REG_NAME	100	The name to which the customer chose to license the product.
ADDITIONAL1	500	1 st Customizable Field (see below).
ADDITIONAL2	500	2 nd 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) (US / Canadian customers only)
COUNTRY	50	The customer's country (english name)
LANGUAGE_ID	2	The customer's preferred language: English = 1; German = 2; Portuguese = 3; Spanish = 4; Italian = 5; French = 6 (others may be added in the future!) (see below for usage hints)

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.

In addition to our fixed set of fields, we offer two customizable fields. These additional input values are optional and can be added to your order form by entering them in the Control Panel. On the bottom of the Products/Program Description page, you will find the settings for additional fields. 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 optional (not checked) or mandatory (checked). If your generator relies on these additional input values, don't forget to make the appropriate changes.

We offer two variable fields that will appear on your order form. You may use these fields to ask your customers questions like "Where did you first hear about our software?" or "Do you want to receive regular updates about the software?" You can edit these fields here or on the control panel. You may also designate these fields as "required" from your users, i.e. they must enter them in order to complete the order. To make them required, simply check the box at the end of the line.

Additional Field 1:

Additional Field 2:

Email Template:
 Your e-mail text, which is being sent with the registration key (only relevant if we send the key for you). Your text must contain a <%KEY%> tag, which will be replaced later by the generated key. [\[Click here for detailed help.\]](#)

```
Dear customer!

Thank you for purchasing our product!
Please open the registration dialog and
enter the following key:

<%KEY%>

Regards,
  Your developer.
```

All input values are passed as strings with ISO-8859-1 (Latin 1) encoding (code page 850). The strings may contain any character between #32 and #255 up to the maximum size specified in the above table. If your code has size restrictions for strings, you must truncate the input values yourself.

3. Output Values

Your key generator can either create a textual or a binary key. In any case 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 provided in the SDK examples!

ERC_SUCCESS	no error (a valid textual key has been generated)
ERC_SUCCESS_BIN	no error (a valid binary key has been generated)
ERC_MEMORY	memory allocation failed
ERC_FILE_IO	error during file i/o
ERC_BAD_ARGS	missing, incomplete or wrong command line arguments (EXE and JAVA generators only)
ERC_BAD_INPUT	missing or incomplete input values
ERC_EXPIRED	key generator has expired due to a limited validity period
ERC_INTERNAL	other internal errors

If the exit code denotes an error you should provide a detailed description in the first result string. (e.g. ERC_BAD_INPUT and "Value for REG_NAME must contain at least 8 characters!").

Textual Keys

A text key generator must provide three output values: an exit code and two strings (a maximum of 4000 characters will be stored). For long texts please use CR/LF for line breaks (character codes #13 and #10).

If the key was generated successfully (ERC_SUCCESS) the two strings are interpreted as the key. There are two output key strings:

1. The first one is the "User Key" which will be delivered inside an e-mail to the customer.
2. The second key is called "CC Key" (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. You can also decide which information is reported to you, for example many publishers are only interested in the key itself, but not the instruction text surrounding it.

You can optionally generate multiple keys for a single purchase whenever a customer buys more than one license depending on your policy.

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.

Delivery e-mail templates

However a much more convenient and flexible way is the Template feature where the generated User Key will be embedded into a prepared Text for delivery. You can edit these texts in your control panel account. This enables you to change the texts without submitting a new key generator.

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.

Binary Keys

A binary key generator must return an exit code, a content description string and the binary key data. Note that successful generation of a binary key is indicated with ERC_SUCCESS_BIN to distinguish binary from textual keys.

The content description returned in the first output is a string which comprises of the contents MIME type and the desired filename separated by a single colon. For example:

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

The size of binary keys is limited to 50 K.

You can use the key template for a usage description. The <%KEY%> tag will be ignored for binary keys.

This SDK release includes example code for Delphi, other Languages will follow.

4. API for EXE and JAVA generators

EXE and JAVA generators are called with three command line arguments, all of which are file names. The first is the input file, the 2nd and 3rd are output files. The input file format is 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. The values are terminated at the end of line, and may contain special characters. An example line could read:

```
REG_NAME=Peter Müller
```

When parsing the lines all unknown keys should be ignored (there may be new input values in the future). Code examples are provided below for the most common programming languages showing how to obtain the values from the input file.

The output files should simply contain the strings (or binary data) described in the section "Output values". These files do not exist when the generator gets called.

Don't forget to set the exit code properly. (See the code examples to learn how this is done in the different programming languages.)

5. API for DLL generators

The following section describes the details of the textual key generator DLL interface. Binary key generator DLL's are currently not supported please submit an EXE generator instead.

Generally, you should be able to implement your own key generator DLL just by using the templates provided. The templates show you how to pass the parameters.

DLLs must export a single function named "GenKeyEx" or "_GenKeyEx@12". The latter is supported as some compilers (like Microsoft Visual C/C++) have a feature called name decoration. Remember that function names are case sensitive. If your compiler assigns a different name to the exported function, your generator will not work with our server. (Should this happen, please try to find a compiler switch that removes the decoration. Use a dump utility to see the exported names of your DLL).

The C function declaration looks like this:

```
DLLEXPORT int __stdcall
GenKeyEx(char * args[], char *userkey, char *cckey)
```

C++ programmers should use the `extern "C"` directive to make exported functions accessible from the calling code:

```
extern "C" DLLEXPORT int __stdcall
GenKeyEx(char * args[], char *userkey, char *cckey)
```

Note that the **stdcall** calling convention is used. The parameters are:

1. `char *args[]` points to an array of pointers to null terminated strings. These strings contain the key-value pairs described above in alternating order. [`*key, *value, *key, *value, ..., NULL`].
2. `char *userkey` points to the memory address where you should store the first result string (max. 4000 bytes).
3. `char *cckey` points to the memory address where you should store second result string (max. 4000 bytes).

The function's return value is interpreted as exit code.

See the code examples for a better understanding or just use the code provided for your convenience. An example for Delphi programmers is also included.

6. 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.

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

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.

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

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.

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 Content-Type header of your response to 'text/plain' (s. examples).

Any other 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 buy 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, !_.\$~#-:äüÄÖÜß@^ (Please note that no blanks are allowed.)

7. 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. Please do not send installer/setup programs, as our support team has no administrator rights on the key server. Access all auxiliary files via the current directory, which is set to your generators location before gets invoked.
- The generators are executed with a time limit of 5000 milliseconds. If the generator takes longer, it will be terminated and thus fail!
- If you need to store persistent information for your key generator you must use a file named <PRODUCT_ID>.ini in the current directory.

8. 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?

9. 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. The CGI feature is now available as described in section 6 of this document.

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: This feature is planned for the future.

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.

10. Examples / Templates

C/C++

The example is written in Standard C, but also works with C++ compilers. We used Borland C++ 5.5 (available for free download at www.borland.com). If you use a different compiler you'll probably have to change the `DLL_EXPORT` macro or use a `.DEF` file for your linker. Also the declaration and/or name of the DLL entry point might be different. Please refer to your compiler's documentation for details.

When you're using a C++ Compiler for the key generator you can use the C examples as well since C is a subset of C++. In this case you'll have to use the `extern "C"` directive to make exported functions accessible from the calling code.

```
extern "C" DLL_EXPORT int __stdcall
GenKeyEx(char * args[], char *userkey, char *cckey)
```

If you are using Visual C++ or Borland C++ Builder you should not use class libraries like MFC or VCL since they use a lot of space and are only required for graphical user interfaces.

FILENAME	DESCRIPTION
key_intf.h	header file (don't touch)
key_bcb.c	key generator (start here with your implementation)
make.bat	make batch file (for use with borland's command line compiler)

Borland Delphi EXE

The Borland Delphi EXE example provides a convenient interface unit, so that you can concentrate on the actual key generation algorithm. 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
DelphiEXE.dpr	project file
KeyIntf.pas	interface functions (don't touch)
KeyUser.pas	key generator (start here with your implementation)

Borland Delphi EXE for Binary Keys

FILENAME	DESCRIPTION
DelphiBin.dpr	project file
KeyIntfBin.pas	interface functions (don't touch)
KeyUser.pas	key generator (start here with your implementation)

Borland Delphi DLL

The Borland Delphi DLL example comes with a parser for the array of input values that is passed by the caller. The DLL example is provided for completeness but we suggest that you use the Delphi EXE example instead.

FILENAME	DESCRIPTION
DelphiDLL.dpr	key generator (start here with your implementation)

Visual Basic 4 and higher

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)

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 instead of `main()` when you use the provided `jkey.exe` as a replacement for `java.exe`. For testing purposes you can use other runtime environments as well, because `main()` calls `KeyMain()` in turn and reports the exit code on the console. Our examples were developed using SUN's JDK 1.2.2.

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.