



***Software Development Kit***  
**E13B**

*Copyright 2020*

***Recogniform Technologies SpA***

# **HOW TO CONTACT US**

Recogniform Technologies SpA

Contrada Concistocchi

87036 Rende (CS), Italy

Phone : +39 0984 404174

Fax : +39 0984 830299

Internet : [www.recogniform.com](http://www.recogniform.com)

E-Mail : [info@recogniform.com](mailto:info@recogniform.com)

# Table of contents

<b>Introduction</b>	<b>5</b>
Copyright .....	5
License .....	5
Overview .....	6
ADVANCED and CLASSIC engine.....	6
<b>Usage</b>	<b>8</b>
Visual C++ .....	8
C# .....	8
Visual Basic .....	8
Visual Basic .NET .....	8
Java .....	8
Delphi .....	8
<b>API References</b>	<b>10</b>
E13B_Init .....	10
E13B_Done .....	11
E13B_Read .....	12
E13B_ResultLinesCount .....	13
E13B_ResultLineLength .....	15
E13B_ResultLineChars .....	16
E13B_ResultLineConfidences .....	17
E13B_ResultLineRects .....	18
E13B_LoadImage .....	19
E13B_FreeImage .....	20
E13B_ImportImage .....	21
E13B_SetParameter .....	22
<b>Sample</b>	<b>27</b>
Code Sample .....	27

# **Introduction**

I

# 1 Introduction

## 1.1 Copyright

The software and the documentation are property of:

Recogniform Technologies SpA  
Contrada Concistocchi  
87036 Rende (CS)  
Italy  
[www.recogniform.com](http://www.recogniform.com)  
[info@recogniform.com](mailto:info@recogniform.com)

## 1.2 License

It is illegal to copy or reproduce this manual, or any part thereof, in any shape or form.

The information contained in this manual is subject to change without notice and does not present a commitment on the part of Recogniform Technologies SpA.

Recogniform Technologies SpA shall not be held liable for technical or editorial errors and/or omissions made here, nor for incidental or consequential damages resulting from the furnishing, performance, or use of the software and documentation.

Recogniform Technologies SpA reserves the right to make changes to the software and documentation without notice.

Product names mentioned here are used for identification purposes only and may be tradenames and/or registered trademarks of their respective companies.

***YOU CANNOT DISTRUBUTE SOFTWARE INCLUDING THIS SDK LIBRARY UNLESS YOU HAVE A WRITTEN AGREEMENT (ROYALTIES FREE OR ROYALTIES BASED) WITH RECOGNIFORM TECHNOLOGIES SPA***

## 1.3 Overview

The library allows to recognize MICR E13B codelines from images acquired from scanner. The basic character set supported contains numeric and symbols characters:

0 1 2 3 4 5 6 7 8 9 and other four symbols called BSB (or Transit), Domestic (or On-Us), Amount, Dash; after recognition they are respectively reported like ":" "<" ";" "=" in ASCII characters.

The recognition process is fast and accurate and you can get too the confidence and the rectangle area of each recognized character...

### 1.3.1 ADVANCED and CLASSIC engine

This new version of the library implements a new advanced recognition engines working natively on low resolution (100 DPI) color images and able to recognize also partially occluded characters. By default the library uses this new advanced engine, but it's possible to switch between the ADVANCED and the CLASSIC engine in any moment (see E13B\_SetParameter function ).

**Usage**



## 2 Usage

### 2.1 Visual C++

You have to include the RECOE13BAPI.C in your program. Before to execute your application make sure the *RECOE13B.DLL* is available in your same .exe directory or in windows\system directory.

### 2.2 C#

You have to include the RECOE13BAPI.CS in your program. Before to execute your application make sure the *RECOE13B.DLL* is available in your same .exe directory or in windows\system directory.

### 2.3 Visual Basic

You have to include the RECOE13BAPI.BAS in your program. Before to execute your application make sure the *RECOE13B.DLL* is available in your same .exe directory or in windows\system directory.

### 2.4 Visual Basic .NET

You have to include the RECOE13BAPI.VB in your program. Before to execute your application make sure the *RECOE13B.DLL* is available in your same .exe directory or in windows\system directory.

### 2.5 Java

You have to use 32 bit JVM and you have to include the RECOE13BAPI.JAVA in your program. Before to execute your application make sure the *RECOE13B.DLL* is available in your same .jar directory.

### 2.6 Delphi

You have to include the RECOE13BAPI.PAS in your program. Before to execute your application make sure the *RECOE13B.DLL* is available in your same .exe directory or in windows\system directory.

## API References



### 3 API References

#### 3.1 E13B\_Init

##### C/C++ Declaration

```
__stdcall long E13B_Init(char* Name, char* Key,  
long* Session);
```

##### C# Declaration

```
int E13B_Init(string Name, string Key, ref int  
Session);
```

##### Visual Basic Declaration

```
Function E13B_Init(ByVal Name As String, ByVal Key  
As String, ByRef Session As Integer) As Integer
```

##### Visual Basic .NET Declaration

```
Function E13B_Init(ByVal Name As String, ByVal Key  
As String, ByRef Session As Integer) As Integer
```

##### Delphi Declaration

```
Function  
E13B_Init(User:PAnsiChar;Password:PAnsiChar;Var  
SessionHandle:THandle):Integer; stdcall;
```

##### Java Declaration

```
int E13B_Init(String Name, String Key, int[]  
Session);
```

##### Description

This is the first function to call: initialize the library and returns a session handle to use in next calls. When you buy the library you receive an "user" and a "password" string necessary to initialize the library in normal mode: without this value or with wrang values the library is initialized in evaluation mode. The evaluation mode works exactly as normal mode but some time, when you call the recognition function, is displayed a warning dialog box remembering the evaluation state: you can close it and continue to work with no problems.

### Parameters

*User* (in) - then user name string

*Password* (in) - then password string

*SessionHandle* (out) - the session handle to use in next library calls

### Return values

*E13B\_OK* : success, the library is initialized

*E13B\_NOMEM* : the system don't have enough memory

*E13B\_NORESOURCE* : some necessary resources don't are available

### Note

The "status" string, shown in the windows dialog, is built by 1 and 0 meaning OK or KO for this checking in the order from left to right:

- SDK initialized with valid user/key
- License file integrity
- License file matching sdk version
- Computer verification done or not required
- Dongle verification done or not required
- Date expiration done or not required
- SDK option used unlocked

## 3.2

### **E13B\_Done**

#### C/C++ Declaration

```
__stdcall long E13B_Done(long Session);
```

#### C# Declaration

```
int E13B_Done(int Session);
```

#### Visual Basic Declaration

```
Function E13B_Done(ByVal Session As Integer) As  
Integer
```

#### Visual Basic .NET Declaration

```
Function E13B_Done(ByVal Session As Integer) As  
Integer
```

### **Delphi Declaration**

```
function E13B_Done(SessionHandle:THandle):Integer;  
stdcall;
```

### **Java Declaration**

```
int E13B_Done(int Session);
```

### **Description**

This is the last function to call when you don't need more services from the library: deinitialize the library and free all used resources.

### **Parameters**

*SessionHandle* (in) - the session handle to free

### **Return values**

*E13B\_OK*: success, the library is finalized

*E13B\_INVALIDHANDLE*: the session handle is not valid

## **3.3 E13B\_Read**

### **C/C++ Declaration**

```
__stdcall long E13B_Read(long Session, long  
DIBHandle, bool SingleLine);
```

### **C# Declaration**

```
int E13B_Read(int Session, int DIBHandle, bool  
SingleLine);
```

### **Visual Basic Declaration**

```
Function E13B_Read( ByVal Session As Integer, ByVal  
DIBHandle As Integer, ByVal SingleLine As Boolean)  
As Integer
```

### **Visual Basic .NET Declaration**

```
Function E13B_Read( ByVal Session As Integer, ByVal  
DIBHandle As Integer, ByVal SingleLine As Boolean)  
As Integer
```

## Delphi Declaration

```
function E13B_Read(SessionHandle:THandle;  
DIB:THandle; SingleLine:Boolean):Integer; stdcall;
```

## Java Declaration

```
int E13B_Read(int Session, int DIBHandle, boolean  
SingleLine);
```

## Description

This is the function performing the recognition: you have to call this function after library initialization to perform the optical character recognition.

## Parameters

*SessionHandle* (in) - the session handle to use

*DIBIn* (in) - the handle of the DIB to recognize

*SingleLine* (in) - flag to help the recognition: if "true" (1) the engine search for a single codeline, if "false" (0) the engine search for multiple codelines.

## Return values

*E13B\_OK*: success, the recognition is done correctly

*E13B\_INVALIDHANDLE*: the session handle is not valid

*E13B\_INVALIDDIB* : the input DIB is not valid, ie. is not a monochrome image

## Notes

The library works with standard DIBs images. Refer to Microsoft documentation for additional info about Device Indipendent Bitmaps. If you have to recognize a gray-scale image, please consider to use our *Dynamic Thresholding Library* to get better result in conversion.

## **3.4 E13B\_ResultLinesCount**

### C/C++ Declaration

```
__stdcall long E13B_ResultLinesCount(long Session,  
long* Lines);
```

### **C# Declaration**

```
int E13B_ResultLinesCount(int Session, ref int  
Lines);
```

### **Visual Basic Declaration**

```
Function E13B_ResultLinesCount(ByVal Session As  
Integer, ByRef Lines As Integer) As LoIntegerng
```

### **Visual Basic .NET Declaration**

```
Function E13B_ResultLinesCount(ByVal Session As  
Integer, ByRef Lines As Integer) As Integer
```

### **Delphi Declaration**

```
function  
E13B_ResultLinesCount(SessionHandle:THandle; Var  
Count:Integer):Integer; stdcall;
```

### **Java Declaration**

```
int E13B_ResultLinesCount(int Session,int Lines);
```

### **Description**

This is the function to call after the recognition process to get the number of codelines found in the image.

### **Parameters**

*SessionHandle* (in) - the session handle to use  
*LinesCount* (out) - the number of codelines recognized in the image

### **Return values**

*E13B\_OK* : success, the lines count is retrieved

*E13B\_INVALIDHANDLE* : the session handle is not valid

## 3.5 E13B\_ResultLineLength

### C/C++ Declaration

```
__stdcall long E13B_ResultLineLength(long Session,
long Line, long* Length);
```

### C# Declaration

```
int E13B_ResultLineLength(int Session, int Line,
ref int Length);
```

### Visual Basic Declaration

```
Function E13B_ResultLineLength(ByVal Session As
Integer, ByVal Line As Integer, ByRef Length As
Integer) As Integer
```

### Visual Basic .NET Declaration

```
Function E13B_ResultLineLength(ByVal Session As
Integer, ByVal Line As Integer, ByRef Length As
Integer) As Integer
```

### Delphi Declaration

```
function
E13B_ResultLineLength(SessionHandle:THandle;
Line:Integer; Var Length:Integer):Integer;
stdcall;
```

### Java Declaration

```
int E13B_ResultLineLength(int Session, long Line,
int[] Length);
```

### Description

This is the function to call after the recognition process to get the length of a recognized codeline.

### Parameters

*SessionHandle* (in) - the session handle to use

*LineIndex* (ind) - It must be always 0

*Length* (out) - the number of character recognized in the selected codeline

### Return values

*E13B\_OK*: success, the line length is retrieved

*E13B\_INVALIDHANDLE*: the session handle is not valid

*E13B\_INVALIDLINE*: the line index is not valid

## 3.6 E13B\_ResultLineChars

### C/C++ Declaration

```
__stdcall E13B_ResultLineChars(long Session, long  
Line, char* Buffer);
```

### C# Declaration

```
int E13B_ResultLineChars(int Session, int Line,  
StringBuilder Buffer);
```

### Visual Basic Declaration

```
Function E13B_ResultLineChars(ByVal Session As  
Integer, ByVal Line As Integer, ByVal Buffer As  
String) As Integer
```

### Visual Basic .NET Declaration

```
Function E13B_ResultLineChars(ByVal Session As  
Integer, ByVal Line As Integer, ByVal Buffer As  
StringBuilder) As Integer
```

### Delphi Declaration

```
function  
E13B_ResultLineChars(SessionHandle:THandle;  
Line:Integer; Buffer:PAnsiChar):Integer; stdcall;
```

### Java Declaration

```
int E13B_ResultLineChars(int Session, int Line,  
Memory Buffer);
```

### Description

This is the function to call after the recognition process to get the chars recognized in a codeline.

### Parameters

*SessionHandle* (in) - the session handle to use  
*LineIndex* (ind) it must be always 0.

*CharBuffer* (out) - the buffer where will be movede the characters recognized in the selected codeline. You have to allocate enough space.

### Return values

*E13B\_OK* : success, the line length is retrieved

*E13B\_INVALIDHANDLE* : the session handle is not valid

*E13B\_INVALIDLINE* : the line index is not valid

## 3.7 E13B\_ResultLineConfidences

### C/C++ Declaration

```
__stdcall long E13B_ResultLineConfidences(long Session, long Line, void* Buffer);
```

### C# Declaration

```
int E13B_ResultLineConfidences(int Session, int Line, float[] Buffer);
```

### Visual Basic Declaration

```
Function E13B_ResultLineConfidences(ByVal Session As Integer, ByVal Line As Integer, ByRef Buffer As Single) As Integer
```

### Visual Basic .NET Declaration

```
Function E13B_ResultLineConfidences(ByVal Session As Integer, ByVal Line As Integer, ByRef Buffer As Single) As Integer
```

### Delphi Declaration

```
function
E13B_ResultLineConfidences(SessionHandle:THandle;
Line:Integer; Buffer:Pointer):Integer; stdcall;
```

### Java Declaration

```
int E13B_ResultLineConfidences(int Session, int Line, float[] Buffer);
```

#### Description

This is the function to call after the recognition process to get the confidences of recognized chars in a codeline. The confidence is the probability that a char was been recognized correctly.

#### Parameters

*SessionHandle* (in) - the session handle to use

*LineIndex* (ind) – It must be always 0.

*ConfidenceBuffer* (out) - the buffer where will be movede the confidences of characters recognized in the selected codeline. You have to allocate enough space. Each confidence value is rapresented by a single precision number in the range 0.0 - 100.0 where 0.0 is lowest confidence and 100.0 is the highest confidence level.

#### Return values

*E13B\_OK*: success, the line length is retrieved

*E13B\_INVALIDHANDLE* : the session handle is not valid

*E13B\_INVALIDLINE* : the line index is not valid

## 3.8 E13B\_ResultLineRects

#### C/C++ Declaration

```
__stdcall long E13B_ResultLineRects(long Session,  
long Line, void* Buffer);
```

#### C# Declaration

```
int E13B_ResultLineRects(int Session, int Line,  
Rect Buffer);
```

#### Visual Basic Declaration

```
Function E13B_ResultLineRects(ByVal Session As  
Integer, ByVal Line As Integer, ByRef Buffer As  
Rect) As Integer
```

## **Visual Basic .NET Declaration**

```
Function E13B_ResultLineRects(ByVal Session As Integer, ByVal Line As Integer, ByRef Buffer As Rect) As Integer
```

## **Delphi Declaration**

```
function
E13B_ResultLineRects(SessionHandle:THandle; Line:
Integer; Buffer: Pointer): Integer; stdcall;
```

## **Java Declaration**

```
int E13B_ResultLineRects(int Session, int Line,
Pointer Buffer);
```

### **Description**

This is the function to call after the recognition process to get the rectangle areas filled by each character in a codeline.

### **Parameters**

*SessionHandle* (in) - the session handle to use

*LineIndex* (ind) – It must be always zero.

*RectsBuffer* (out) - the buffer where will be move the rects of characters recognized in the selected codeline. You have to allocate enough space. Each rect value is represented by four long numbers representing left, top, right and bottom position of the rectangle.

### **Return values**

E13B\_OK : success, the line length is retrieved

E13B\_INVALIDHANDLE : the session handle is not valid

E13B\_INVALIDLINE : the line index is not valid

## **3.9**

## **E13B\_LoadImage**

### **C/C++ Declaration**

```
__stdcall long E13B_LoadImage(long Session, char*
fileName);
```

### **C# Declaration**

```
int E13B_LoadImage(int Session , string FileName);
```

### **Visual Basic Declaration**

```
Function E13B_LoadImage(ByVal Session As Integer,  
ByVal FileName As String) As Integer
```

### **Visual Basic .NET Declaration**

```
Function E13B_LoadImage(ByVal Session As Integer,  
ByVal FileName As String) As Integer
```

### **Delphi Declaration**

```
function E13B_LoadImage(SessionHandle:THandle;  
FileName:PChar): Integer; stdcall;
```

### **Java Declaration**

```
int E13B_LoadImage(int Session , String FileName  
);
```

### **Description**

This function allows to load an image from file obtaining a DIB handle. Supported files formats are TIF, JPG, PNG and BMP

### **Parameters**

*SessionHandle* (in) - the session handle to use  
*FileName* (in) - the name of the file to load

### **Return values**

The DIB handle with the image inside the file, 0 if an error occurred.

## **3.10 E13B\_FreeImage**

### **C/C++ Declaration**

```
__stdcall void E13B_FreeImage(long Session, long  
DIBHandle);
```

### **C# Declaration**

```
void E13B_FreeImage(int Session, int DIBHandle);
```

### **Visual Basic Declaration**

```
Sub E13B_FreeImage(ByVal Session As Integer, ByVal  
DIBHandle As Integer)
```

### **Visual Basic .NET Declaration**

```
Sub E13B_FreeImage(ByVal Session As Integer, ByVal  
DIBHandle As Integer)
```

### **Delphi Declaration**

```
procedure E13B_FreeImage(SessionHandle:THandle;  
DIBHandle: Integer); stdcall;
```

### **Java Declaration**

```
void E13B_FreeImage(int Session , int DIBHandle );
```

### **Description**

This function allows to remove from memory an image previously loaded from file.

### **Parameters**

*SessionHandle* (in) - the session handle to use  
*DIBHandle* (in) - the handle of the DIB to free

### **Return values**

n/a

## **3.11 E13B\_ImportImage**

### **C/C++ Declaration**

```
__stdcall long E13B_ImportImage(long Session, long  
HandleBitmap, long HandlePalette );
```

### **C# Declaration**

```
int E13B_ImportImage(int Session, int  
HandleBitmap, int HandlePalette);
```

### **Visual Basic Declaration**

```
Function E13B_ImportImage(ByVal Session As Integer, ByVal HandleBitmap As Integer, ByVal HandlePalette As Integer)As Integer;
```

### **Visual Basic .NET Declaration**

```
Function E13B_ImportImage(ByVal Session As Integer, ByVal HandleBitmap As Integer, ByVal HandlePalette As Integer)As Integer;
```

### **Delphi Declaration**

```
function E13B_ImportImage(Session:Integer;  
HandleBitmap:Integer;  
HandlePalette:Integer):Integer; stdcall;
```

### **Java Declaration**

```
int E13B_ImportImage(int SessionHandle, int  
HandleBitmap, int HandlePalette);
```

#### **Description**

This function allows to create a DIB from a DDB and its palette.

#### **Parameters**

*Session* (in) - the session handle to use

*HandleBitmap* (in) - the handle of the bitmap to import

*HandlePalette* (in) - the handle of the palette

#### **Return values**

*DIBHandle* - the handle of the DIB

## **3.12 E13B\_SetParameter**

### **C/C++ Declaration**

```
__stdcall long E13B_SetParameter(long Session,  
long ParameterID, long ParameterValue );
```

### **C# Declaration**

---

```
int E13B_SetParameter(int Session, int
ParameterID, int ParameterValue);
```

### **Visual Basic Declaration**

```
Function E13B_SetParameter(ByVal Session As
THandle, ByVal ParameterID As Integer,ByVal
ParameterValue As Integer)As Integer;
```

### **Visual Basic .NET Declaration**

```
Function E13B_SetParameter(ByVal Session As
THandle, ByVal ParameterID As Integer,ByVal
ParameterValue As Integer)As Integer;
```

### **Delphi Declaration**

```
function E13B_SetParameter(Session:THandle;
ParameterID:Integer;
ParameterValue:Integer):Integer; stdcall;
```

### **Java Declaration**

```
int E13B_SetParameter(int Session,int ParameterID,
int ParameterValue);
```

### **Description**

This function allows to set some processing parameter.

Parameters BSB (or Transit), Domestic (or On-Us), Amount, Dash

*Session* (in) - the session handle to use

*ParameterID* (in) - ID of the parameter to set:

- *E13BPARAM\_RECOGNIZESPACE* (1) allows to enable/disable the recognition of spaces in the codeline. This parameter is valid only for the CLASSIC engine.
- *E13BPARAM\_ENGINE* (2) allows to select one of the available recognitions engine available
- *E13BPARAM\_EXPECTEDCHARS* (3) allows to set the number of character to recognize. This parameter is valid only for the ADVANCED engine
- *E13BPARAM\_SYMBOL1* (4) allows to set the ascii code of the BSB symbol instead of ":" .
- *E13BPARAM\_SYMBOL2* (5) allows to set the ascii code of the

- Domestic symbol instead of “;” .
- *E13BPARAM\_SYMBOL3* (6) allows to set the ascii code of the Amount symbol instead of “<” .
  - *E13BPARAM\_SYMBOL4* (7) allows to set the ascii code of the Dash symbol instead of “=” .
  - *E13BPARAM\_MAXRESOLUTION* (8) allows to set the maximum resolution to be used internally in DPI, This parameter is valid only for the ADVANCED engine
  - *E13BPARAM\_MULTISPECTRAL* (9) allows to enable/disable the multispectral recognition. This parameter is valid only for the ADVANCED engine

*ParameterValue* (in) - value of the parameter to set:

- For *E13BPARAM\_RECOGNIZESPACE* allowed values are 0 or 1 for disable/enable the recognition of spaces. Default value is 0, so no recognition of spaces is executed.
- For *E13BPARAM\_ENGINE* allowed value are 0 to select the new recognition engine, 1 to select the classic recognition engine. Default value is 0, so the new recognition engine is used.
- For *E13BPARAM\_EXPECTEDCHARS* allowed value are 0 for automatic setting or the number of character to recognize. If the number is inserted the software avoid to take dirty characters
- For *E13BPARAM\_SYMBOL1* allowed value are all character that have a counterpart in the ascii table
- For *E13BPARAM\_SYMBOL2* allowed value are all character that have a counterpart in the ascii table
- For *E13BPARAM\_SYMBOL3* allowed value are all character that have a counterpart in the ascii table
- For *E13BPARAM\_SYMBOL4* allowed value are all character that have a counterpart in the ascii table
- For *E13BPARAM\_MAXRESOLUTION* the default value is 150. If an higher value is set the software became more slow
- For *E13BPARAM\_MULTISPECTRAL* allowed values are 0 or 1 for disable/enable the multispectral recognition. The default value is 1. If the image is a color image the recognition will be multi-spectral)

#### Return values

*E13B\_OK*: success, the line length is retrieved

*E13B\_INVALIDHANDLE* : the session handle is not valid

*E13B\_INVALIDLINE* : the line index is not valid

*E13B\_INVALIDPARAMETER* : the parameter is not valid

*E13B\_NOMEM* : there are memory problem

E13B\_NORESOURCE : there are no available resources in the system  
E13B\_INVALIDDIB : the input DIB is not valid, ie. the resolution is not set properly  
E13B\_INTERNALEXCEPTION :an internal error occurred  
E13B\_NORUNTIMELICENSE : there is no license  
E13B\_EVALUATIONLIMITREACHED: the number of recognition are finished for an evaluation version that don't show nag screen.

**Sample**

V

## 4 Sample

### 4.1 Code Sample

```

// Demo user function
long RecognizeMyCodeline(long CodelineDIB)
{
    // Declare local variables
    long Session,Error,Lines,LineLen;
    char Chars[1024]; // We expect to have codelines with less
then 1024 chars !
    single Confidences[1024];
    rect Rects[1024];

    // Init the library
    Error=E13B_Init("demo","demo", &Session);

    // Check if any error occurred
    if (Error==E13B_OK)
    {

        // Perform the recognition
        Error=E13B_Read(Session, CodelineDIB, FALSE);

        // Check if any error occurred
        if (Error==E13B_OK)
        {

            // Retrieve the codeline length
            E13B_ResultLineLength(Session,i,&LineLen);

            //

=====
=           // You could allocate here memory for Chars,
Confidences and Rects
           // using LineLen value instead to use fixed size
buffers...
           //
=====

            // Retrieve the codeline chars
            E13B_ResultLineChars(Session,i,&Chars);

            // Retrieve the codeline char confidences...if
required
            E13B_ResultLineConfidences(Session,i,&Confidences);

            // Retrieve the codeline char rects...if required
            E13B_ResultLineRects(Session,i,&Rects);

            //

=====

            // Add here your code to use the recognized data
            //
=====

    }
}

```

```
// Deinitialize the library  
E13B_Done(Session);  
}  
return (Error);  
}
```

*Annotation*