# Software License Agreement

## SFN Font Library for .NET Core

### OS Independent Font Library

**version 1**

**2024**



*SUB SYSTEMS, INC.*

**3200 Maysilee Street**

**Austin, TX 78728**

**512-733-2525**

**Software License Agreement**

The Software is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The Software is licensed, not sold. This LICENSE AGREEMENT grants you the following rights:

A. This product is licensed per developer basis only. Each developer working with this package needs to purchase a separate license.

B. The purchaser has the right to link the DLL functions into their application. The following restriction apply on its usage: the target application may not be a stand-alone font library; the target application may use this product for one operating system platform only; and the source code (or part) of the library may not be distributed in any form.

C. The SERVER LICENSE allows for the server application development. The server licenses must be purchased separately when using this product in a server application. Additionally, the product is licensed per developer basis. Only an UNLIMITED SERVER LICENSE allows for royalty-free distribution of your server applications using this product.

D. ENTERPRISE LICENSE: The large corporations with revenue more than $50 million and large government entities must purchase an Enterprise License. An Enterprise license is also applicable if any target customer of your product using the Software have revenue more than $500 million. Please contact us at info@subsystems.com for a quote for an Enterprise License.

E. Your license rights under this LICENSE AGREEMENT are non-exclusive. All rights not expressly granted herein are reserved by Licensor.

F. You may not sell, transfer or convey the software license to any third party without Licensor's prior express written consent.

G. The license remains valid for 12 months after the issue date. The subsequent year license renewal cost is discounted by 20 percent from the license acquisition cost. The license includes standard technical support, patches and new releases.

H. You may not disable, deactivate or remove any license enforcement mechanism used by the software.

This software is designed keeping the safety and the reliability concerns as the main considerations. Every effort has been made to make the product reliable and error free. However, Sub Systems, Inc. makes no warranties against any damage, direct or indirect, resulting from the use of the software or the manual and can not be held responsible for the same. The product is provided 'as is' without warranty of any kind, either expressed or implied, including but not limited to the implied warranties of suitability for a particular purpose. The buyer assumes the entire risk of any damage caused by this software. In no event shall Sub Systems, Inc. be liable for damage of any kind, loss of data, loss of profits, interruption of business or other financial losses arising directly or indirectly from the use of this product. Any liability of Sub Systems will be exclusively limited to refund of purchase price.

Sub Systems, Inc. offers a 30 day money back guarantee with the product. Must call for an RMA number before returning the product.

# Getting Started

This chapter describes the contents of the software distribution ZIP file, and provides a step by step process of incorporating the font library into your application. To begin:

1. Add the reference for sfn.dll in your project.

Create a project reference for the included product package. The package name is found as sfn.1.n.n.n.nupkg. The 'n.n.n' stands for the product minor release number. This is how your project file would apear:

```
<PackageReference Include="sfn" Version="1.0.0.0"/>
```

Also, please ensure that:

2. Add the 'using' or 'Import' namespace statement for the project dll, example:

using SubSystems.SF

or

Import SubSystems.SF

**In This Chapter**

# Files

The distribution zip filee includes a nuget package called sfn.1.n.n.n.nupkg. The 'n.n.n' stands for the product minor release number.

**DLL Demo Files:**

The following demo files are included in the c_demo.zip file.

| | |
|---|---|
| demo.cs | Source code for the demo program |
| demo.exe | Executable demo program |
| demo.csproj | The project file to compile the demo. |
| AssemblyInfo.cs | Assembly information file |

# License Key

*Your License Key and License number are e-mailed to you after your order is processed.* You would set the license information using the SfnSetLicenseInfo static function. This should be preferably done before creating the Sfn object to avoid pop-up nag screens.

int SfnSetLicnseInfo(String LicenseKey, String LicenseNumber, String CompanyName);

| | |
|---|---|
| LicenseKey: | Your license key is available in the product delivery email sent to you upon the purchase of the product. It consists of a string in the form of "xxxxx-yyyyy-zzzzz". |
| LicenseNumber: | Your license number is also available in the product delivery email. The license number string starts with a "srab" or "smo" prefix. |

CompanyName: Your company name as specified in your order.

**Return Value:** This method returns 0 when successful. A non-zero return value indicates an error condition. Here are the possible return values:

0 License application successful.

1 Invalid License Key.

2 Invalid License Number.

3 Ran out of available licenses. Please consider purchasing additional licenses.

Example:

result=Sfn.SfnSetLicenseInfo("xxxxx-yyyyy-zzzzz","srabnnnnn-n","Your Company Name")

Replace the 'xxxxx-yyyyy-zzzzz' by your license key, replace "srabnnnnn-n" with your license number, and "Your Company Name" with your company name as specified in your order.

**Note:** *SfnSetLicenseInfo method should be called only once at the beginning of your application. Calling this method for each conversion would degrade the conversion performance.*

# Sample Code

Please follow these steps to get started:

1. Include the sfn.1.n.n.n.nupkg nuget package in your project. This is how your project file entry would appear:

<PackageReference Include="sfn" Version="1.0.0.0"/>

This package is included in the distribution zip folder.

2. Call SfnSetLicenseInfo method to set your license information

```
Int  LicenseCode=Sfn.SfnSetLicenseInfo("YourLicenseKey",
"YourLicenseNumber", "YourCompanyName");
if (LicenseCode!=0) {
   int LicenseStatusCode=Sfn.SfnGetLicenseStatus();
   String ErrText=Sfn.SfnLastMsgText + "license status
```

```
code: "+LicenseStatusCode.ToString();
```

3. Call one of these constructors to create the Sfn object:

```
Sfn sfn=new Sfn("");   // This message instructs the
library to use the stock truetype fonts included in the
DLL.


Sfm sfn=new Sfn(FontFolder); // This method allows to
specify the path to a folder that contains additional
truetype fonts which you would like to include for font
selection
```

4. Create a font request object and fill it with the parameters to create a font:

```
Sfn.ClsFontReq req=new Sfn.ClsFontReq();

req.typeface=”Arial”;   // or another font typeface

int res=1440;       // font resolution. In this example we
want to create a high resolution font at 1440 dpi.

int PointSize=12;  // This example create a 12 point font

req.PointSize=(int)(PointSize*res/72);  // Specify the
point-size in resolution unit

req.MustChar=(uint)0x680; // specify a character that must
be supported by this font.  This parameter lets you
specify a unicode character block that this font must
support.  Here we are specifying an Arabic character
0x680. For English, you would set MustChar to (uint)’A’.

req.CharSet=0;  // This is windows character set value.
It is used only if the MustChar parameter is not specified


// The following three parameters lets you specify if you
want to create a bold, italic or strike font.

req.bold=false;

req.italic=false;

req.strike=false;


int SfnId=sfn.SfnCreateFont(req);  // create font id using
this font request object
```

5. Get text metrics for your text using this font id:

```
Sfn.ClsTextMetric mt=sfn.GetTextMetric(SfnId, text,
false);  // set the third parameter to true to place the
text in the right-to-left direction.


The GetTextMetric method returns an object of the class
ClsTextMetrix.  This object provides you with the
following information:
```

```
Please refer to the GetTextMetric method in the Control
Methods topic
```

# Control Methods

| In This Chapter |
| --- |
| GetTextMetric |
| SfnCreateFont |

## GetTextMetric

**Get text glyphs, width and text to glyph map**

Sfn.ClsTextMetric mt=sfn.GetTextMetric(int SfnId, String text, bool direction);

| | |
| --- | --- |
| SfnId | Font id as returned from a previous call to the SfnCreateFont method |
| text | Text string to analyze |
| direction | Set to true to specify a right-to-left text flow. Most application will specify a false value to specify left-to-right text flow. |

**Return value:** This function when successful returns a text metric object. A false value indicates an error condition.

Here is the information contained in the ClsTextMetric object:

| | |
| --- | --- |
| mt.typeface | Typeface used by the library for the specified text. This would generally be the same as the typeface used to create the font id (SfnId). However, the library can choose another font if the |

| | |
|---|---|
| | characters in your specified text are not supported by SfnId. |
| mt.SfnCharSet | Character set used by the library. The character set values are one of the Windows character set and generally not relevant to cross OS applications. |
| mt.FullName | Full name of the typeface selected by the library. |
| mt.ascent | Text ascent in the resolution used to create SfnId. The text ascent specifies the distance from the baseline of the text to the top of the character box. |
| mt.descent | Text descent in the resolution used to create SfnId. The text descent specifies the distance from the baseline of the text to the bottom of the character box. |
| mt.height | Text height in the resolution used to create SfnId. The text descent specifies the distance from the top of the text to the bottom of the character box. |
| mt.RawGlyph | An array of raw glyphs generate for the text. The raw glyphs are the glyphs before glyph substitution is applied on the glyphs. |
| mt.glyph | An array of final glyphs generate for the text. The final glyphs are obtained by applying context sensitive substitutions to raw glyphs. |
| mt.width | An array containing the width of each glyph in the mt.glyph array. The total width of the specified text is the sum of the widhts in this array. |
| mt.order | This array contains the glyph order of each character in the input ‘text’ string. |
| mt.rtl | Text direction. It is set to true if the text contains right-to-left placement characters. |
| | |

# SfnCreateFont

**Create a font  id using the font request parameters.**

int SfnId=sfn.SfnCreateFont(ClsFontReq req);

| | |
|---|---|
| Req: | An object contaning font request parameters such as typeface, point-size, etc. |

**Return value:** This fuction returns a non-negative font id value.  A return value of -2 indicates a license validation error.  A return value of -1 indicates a general error.

**Example:**

```
Sfn.ClsFontReq req=new Sfn.ClsFontReq();

req.typeface=”Arial”;   // or another font typeface

int res=1440;       // font resolution. In this example we want to create a high resolution font at 1440 dpi.

int PointSize=12;  // This example create a 12 point font

req.PointSize=(int)(PointSize*res/72);  // Specify the point-size in resolution unit

req.MustChar=(uint)0x680; // specify a character that must be supported by this font.  This parameter lets you specify a unicode character block that this font must support.  Here we are specifing an Arabic character 0x680. For English, you would set MustChar to (uint)’A’.

req.CharSet=0;  // This is windows character set value. It is used only if the MustChar parameter is not specified


// The following three parameters lets you specify if you want to create a bold, italic or strike font.

req.bold=false;

req.italic=false;

req.strike=false;


int SfnId=sfn.SfnCreateFont(req);  // create font id using this font request object
```