Quantum Enterprises

Scriptalizer DLL Integration Guide

Version History

Version	Release date	Comments
1.0	17th July 2016	Initial release
1.1	4th August 2016	Add Descriptalize method
1.2	17th May 2017	Add support for COM interop incl Word VBA
1.3	14th July 2019	Clarify registration instructions

Introduction

Scriptalizer is a tool to generate natural-looking handwriting text using a combination of custom fonts, a configuration file, and randomisation logic. With multiple glyphs for each written character, Scriptalizer takes input text and adds appropriate beginning and end strokes, randomises glyphs used for each character, deals with repeated characters and much more. The result is a string of character codes which, in a “normal” font, are entirely undecipherable; but when rendered in a Scriptalizer font results in a natural, realistic representation of handwriting.

Scriptalizer can be invoked via the Scriptalizer website www.scriptalizer.co.uk but is also available under license as a standalone DLL. This allows you to embed the Scriptalizer logic within your own application, generating “handwritten” output programmatically.

Scriptalizer is COM-compatible and can be invoked in MS Office applications and others that support COM interoperability. This document includes instructions for installing and registering Scriptalizer for COM use.

This document describes the prerequisites, options and integration of that DLL and may be used in conjunction with the fully-working example code.

Requirements

The DLL (delivered as file QuantumScriptalizer.DLL) is a Microsoft .Net assembly targeting Microsoft .Net Framework 2.0. It should also run on later versions of the Framework. Note that it has been developed and tested in a Windows environment. We are not aware of any reasons why it should not work on other platforms supporting .Net (such as .Net Core or Mono on Linux) but this has not been tested and is not currently supported by Quantum Enterprises.

No additional software is required, although to successfully use Scriptalizer you will need at least one Scriptalizer font installed on the machine where the generated output is to be rendered, plus access to the configuration file for that font (or a compatible generic configuration).

Any .Net language should be able to call the DLL directly, and examples in both VB.Net and C# are provided, for a Windows Form application and a website demo respectively.

Integration - .Net

Make sure your project includes a reference to the QuantumScriptalizer.DLL file.

For simplicity and usability, we recommend importing the Quantum namespace into your code:

VB.Net

```
Imports Quantum
```

C#

```
using Quantum;
```

Create an instance of the class “Scriptalizer”. The constructor takes no parameters. Scriptalizer implements iDisposable so you can declare it either via a Dim statement or as the scope of a Using block:

VB.Net

```
Using script As New Scriptalizer()
```

C#

```
using (Scriptalizer script = new Scriptalizer()) {
```

(If you have not imported the Quantum namespace, you will need to use `Quantum.Scriptalizer()` instead)

The Scriptalizer class exposes a method, “Scriptalize”, which takes four parameters and returns scriptalized text.

```
Function Scriptalizer.Scriptalize(InputText As String, Config As String, ConfigType As ConfigTypes, ErrorFrequency As Integer) As String
```

```
string Scriptalizer.Scriptalize(string InputText, string Config, ConfigTypes ConfigType, int ErrorFrequency)
```

The parameters are described below:

- InputText – the plain text to be converted
- Config – a string containing one of:
    - The full path and filename of the configuration file to use
    - The actual configuration content
- ConfigType – an enum from Quantum.ConfigTypes indicating what the config parameter contains. Valid values are ExternalFile and ConfigData
- ErrorFrequency – an integer indicating the frequency of generating "random" crossed-out words in the output.

InputText can contain any characters supported by the target Scriptalizer font. It can include line breaks, spaces and tab characters. Bear in mind that if the Scriptalized output is to be used in a web page, these characters will need further manipulation to give the desired layout on the web page.

The *Config* parameter allows you to pass the configuration data associated with the target font to the method. Configuration data identifies the font version, and includes information about the available character set of the font. It's important to use the correct configuration data for the target font to avoid undecipherable characters appearing in the output. When you purchase a Scriptalizer font you will also have received a configuration file. When calling the Scriptalize method, you may pass this data in by providing the full path and filename of the configuration file; or by passing the actual file contents (config files are typically around 250 bytes). If passing actual file contents, it's important that you preserve the white space and line breaks within the file content.

ConfigType indicates whether you're passing actual config data, or the path and name of a config file to use.

ErrorFrequency is an integer in the range 0 to 32768. 0 indicates that no crossed-out errors are to be generated. Other values indicate that 1 in [n] words of between 1 and 5 characters is to have a random character in the range a – z inserted somewhere in it, then to be crossed-out and the word repeated correctly.

The Scriptalize method returns a string of scriptalized text. As above, if this is to be displayed in a web page it may need further manipulation to present line breaks and tabs as required.

VB.Net

```
outputText = script.Scriptalize(inputText, configFilename, ConfigTypes.ExternalFile, errorRate)
```

C#

```
outputText = script.Scriptalize(inputText, configFilename, ConfigTypes.ExternalFile, errorRate);
```

The class also exposes a second method, "Descriptalize". This takes a single string parameter, sInput, containing "scriptalized" text, and returns a string result, containing de-scriptalized plain-text. NOTE that if the scriptalized text includes "errors" (crossed-out words), these will not be fully descriptalized.

Integration – COM

Before using Scriptalizer in your COM projects (e.g. from MS Word), you need to install and register the component.

Place the **QuantumScriptalizer.DLL** and **QuantumScriptalizer.TLB** files from the package into your Windows System folder. For 32-bit systems this is typically located at **C:\Windows\System**; for 64-bit systems, the files should be installed in **C:\Windows\SYSWOW64** if present, or **C:\Windows\System32** if not.

Next register the component with Windows: Open a command shell window ***with administrator privileges***. (Choose Programs/Accessories/Command Prompt from the start menu, but then right-click and choose "Run as Administrator"). Change the path to the folder where your QuantumScriptalizer.DLL and QuantumScriptalizer.TLB files are located.

Enter the command

[framework]**RegAsm.exe QuantumScriptalizer.dll /codebase**

where *[framework]* is the full path to your .Net framework installation. This will typically be **C:\Windows\Microsoft.NET\Framework\v4.0.30319** if you have .Net 4 installed, or **C:\Windows\Microsoft.Net\Framework\v2.0.50727** if your latest .Net version is 2.0. You should see the confirmation message "Types registered successfully".

Your COM-compliant application will then need to add a reference to Scriptalizer. For Microsoft Office applications, this will involve opening the **VBA Editor** and choosing **Tools**, **References** from the main menu bar. **QuantumScriptalizer** should be shown in the list of components; tick the checkbox next to it and click OK. If the references don't include QuantumScriptalizer, then choose **Browse**, navigate to the folder containing the .DLL and .TLB files, and select the QuantumScriptalizer.TLB file; then click **OK** and continue as above.

You can then add code to invoke the methods described in "Integration - .Net" above. For a full working example, see the separate document "**Word_VBA.doc**" included in the Scriptalizer package (in the **Examples/Word** folder). Open the VBA Editor and click on "**ThisDocument**" in the project explorer window; the main window will show the VBA code to execute the Scriptalizer. The document is also setup with a custom toolbar to invoke Scriptalizer, plus a keyboard shortcut of **Alt-Ctrl-s** to launch it. Before running it, you will need to update the path to your font's configuration file, and also update the name of the font in the last line of the subroutine.

Error handling

The DLL does not throw exceptions and should always successfully return a text string. However if the passed parameters are invalid or a problem occurs scriptalizing the input, an error message will be returned rather than the scriptalized input.

Error messages all start with the text “S!” followed by a 3-digit number, and some descriptive text, as follows:

- `S!001: Unable to load config from file [ConfigFilename]`
- `S!002: invalid, corrupt or missing config data`
- `S!003: an Error processing your configuration file. Please contact Quantum Enterprises for support, quoting error message "[MessageText]"`
- `S!004: Invalid or corrupt Config data`
- `S!005: Error occurred during scriptalization; Please contact Quantum Enterprises for support, quoting error message "[MessageText]"`
- `S!099: Unexpected program condition; Please contact Quantum Enterprises for support, quoting error message "S!099"`

S!001: When passing the path + filename of an external Config file, the DLL was unable to locate/read the file. Check the path is correct and that your application has read permissions to the file.

S!002: Can occur with either ConfgType, and indicates corrupt or empty configuration data. Check the data content, ensuring that if passing the data (rather than a reference to the file), the lines are separated by carriage-return / line-feed pairs (vbCrLf in VB.Net).

S!003: An error occurred during pre-processing of your config data. Check your config file as above; if unable to resolve, please contact Quantum Enterprises, supplying the config file/data used.

S!004: Indicates that the config data was readable but was invalid (typically indicates a truncated config file or one not correctly separated by carriage-return / line-feed pairs.

S!005: An error occurred during actual scriptalization. In most cases this will be due to a previously undetected inconsistency in the config data, but may be indicative of an issue within the Scriptalizer itself. Please contact Quantum Enterprises, supplying the config file/data used, and the input text you were trying to Scriptalize.

S!099: An unhandled exception occurred at some point within the DLL execution. If you encounter this error, please contact Quantum Enterprises, supplying the config file/data used, the input text supplied, and the full error message received.

The class also exposes a second method, Descriptalize. This takes a single string parameter, sInput, containing “scriptalized” text, and returns a string result, containing de-scriptalized plain-text. NOTE that if the scriptalized text includes “errors” (crossed-out words), these will not be fully descriptalized.