



Quickstart and Reference guide v1.2

Version	Author	Comments
V1.1	David Laing	Release version
V1.2	David Laing	Added VB.NET documentation, and updated for v1.2

Overview

The .NET startup problem

Before your Windows Forms / Visual Basic / VB.NET / C# .NET can do anything (even show a splashscreen), the .NET framework must be loaded into memory. On a cold start (i.e, just after a reboot), the .NET framework isn't loaded into memory by default, nor is it in the disk cache; so approximately 21Mb must first be loaded from disk before your application gets to execute its first instruction. Depending on hard disk speed, this can take up to 7sec on 2005 hardware, or 20sec on 2001 hardware!

The result is that the time between the user clicking on your application icon, and the first indication that something (anything!) happened can be up to 20 seconds. Users will often end up clicking on the app icon multiple times, trying to make something happen, and thinking "what a crap app!" Not the kind of first impression you want, especially if you are trying to entice your users away from a "speedy" legacy application.

The Solution

QuickSplash.NET is a native Win32 component that shows a (customised) splashscreen - in less than 0.5 seconds, whilst loading your Windows Forms / Visual Basic / VB.NET / C# .NET application in the background. Once the framework and your application has initialised, a simple file deletion tells the splashscreen to close.

This dramatically improves the perceived startup performance of your application. Rather than having to wait for up to 20 seconds wondering if anything is happening, your users watch an interactive splash screen giving them a progress update of what is happening. 20 seconds just seem to fly by when you're distracted by something.

Contents

Overview	2
The .NET startup problem	2
The Solution	2
Contents	2
Components	3
ChangeLog	3
Tutorials: Using QuickSplash.NET in your project	5
.NET 2.0 – Windows forms project.....	5
.NET 1.1 – Windows forms project.....	7
Creating a Setup project using the Splashscreen.....	9
Licensing: Removing the Unregistered notice.....	10
Further questions, comments, suggestions	10
Appendix 1: Configuration file options.....	11
Appendix 2: Command line options.....	11

Components

QuickSplash.NET is made up of the following files:

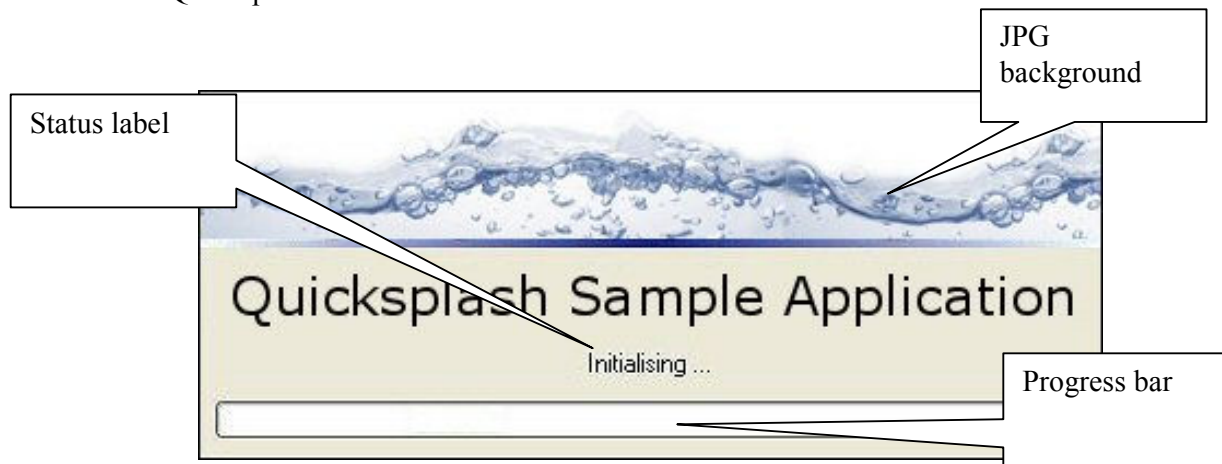
Folder	Files	Description
.	Quickstart & Reference Guide	This document!
Splashscreen	Splash.exe SplashConfig.cfg SplashScreen-Background.jpg SplashScreen-ProgressBar.bmp	This is the core of QuickSplash. Running this reads the .cfg file, loads the splashscreen & loads your .NET app in the background. Contains all the config options telling quicksplash which splashscreen to show, and which .NET app to load. Splashscreen background Individual progress bars (optional)
Samples\v1.1	SampleProjects.sln	Sample projects showing how to use the Splashscreen for .NET v1.1
Samples\v2.0	SampleProjects.sln	Sample projects showing how to use the Splashscreen for .NET v2.0

ChangeLog

v1.1 to v1.2

1. Start.exe has been renamed Splash.exe (to avoid conflict with Windows internal Start command)
2. Added “EXE Startup Check:” config option
3. Added “Lock File Path:” config option. You can now specify the folder where the lock files will be created. Lockfiles are now also named <guid>.lock, rather than data.lock, with the path & name of the lock file passed to the .NET application via a command line parameter. This is useful when the splashscreen needs to be used in multi-user scenarios, such as with Terminal Services. Calling Splash.exe –overrideunlockfileguid=mystring replaces <guid> with mystring.

4. How Quicksplash.NET works



1. Splash.exe, loads a JPG file (as specified in SplashConfig.cfg) to centre of screen. (under 0.5 sec)
2. Creates <guid>.lock file
3. A status label can be displayed anywhere on the JPG background. Text written to <guid>.lock file is displayed in the "Status" label. Overwrite <guid>.lock file with new text to change Status label.
4. A marquee style progress bar can be displayed on the splashscreen
5. In the mean time, the .NET app is loaded in background (.NET .exe location specified in config file)
6. When the .NET app has loaded, it deletes the <guid>.lock file. This causes the splash screen to close itself

Tutorials: Using QuickSplash.NET in your project

.NET 2.0 – Windows forms project

This tutorial shows you how to create the splashscreen for the sample project found at:

Samples\v2.0\CSharp-ProgressBarAndStatusLabel & Samples\v2.0\VB-ProgressBarAndStatusLabel.

1. Create a Windows Forms project.
2. Copy the files from \Splashscreen into the Windows Forms project folder.

Name	Size	Type
ConnectWait.ico	2 KB	Icon
SplashConfig.cfg	1 KB	Microsoft Office Ou...
SplashScreen-Background.jpg	19 KB	JPEG Image
SplashscreenController.cs	1 KB	Visual C# Source file
SplashscreenController.vb	1 KB	Visual Basic Source file
SplashScreen-ProgressBar.bmp	1 KB	Bitmap Image
start.exe	480 KB	Application

If coding in VB, you only need the SplashscreenController.vb file, and visa versa if coding in C#.

3. Make sure that all of these are included in the VS2005 project file. (Project > Show All Files, select files, context menu > Include in project).
4. Select ConnectWait.ico, Splashscreen.cfg, Splashscreen-ProgressBar.bmp, Splashscreen-Background.jpg & Splash.exe. Properties, Build Action = Content, Copy to Output Directory = Copy Always
5. Have a quick look at SplashscreenController.cs/vb. This contains functions that are used to control the splashscreen whilst the .NET app is loading.
6. Edit the file containing the main entry point. In C#, this will be the file containing the static Main() method, usually Program.cs. In VB, this will be the constructor of the first form that is loaded.

Add the following line as the first statement in this method.

```
Splashscreen.SplashscreenController.UpdateStatusLabel("Initialising");
```

This causes the splashscreen to display “Initialising” in the status label by overwriting the contents of <guid>.lock with “Initialising”.

7. If your application connects to a database, or loads data from some location, you might want to display several status messages using this method here.
 - a. Finally, once your application has loaded, and the main form is displayed to the user, you will want to close down the splashscreen. A good way to do this is to create an event handler in the main form for the Application.Idle event.

Within this event, Call the `SplashscreenController.CloseSplash()` method, which deletes the <guid>.lock file, and causes the splashscreen to close.

In C#:

```
public Form1()
{
    InitializeComponent();
    Application.Idle += new EventHandler(OnLoaded);
}

/// <summary>
/// Custom "OnLoaded" event. Fired after form finished
loaded & displayed
/// </summary>
/// <param name="sender"></param>
/// <param name="args"></param>
protected virtual void OnLoaded(object sender, EventArgs args)
{
    Application.Idle -= new EventHandler(OnLoaded);
}
```

Quickstart and Reference Guide – v1.2

```
        //Shut down the splashscreen
        SplashscreenController.CloseSplash();
    }
b. In VB:
Public Sub New()
    MyBase.New()
    SplashscreenController.UpdateStatusLabel("Connecting to datasource...")

    InitializeComponent()
    'Trap the idle event - this tells us when the form has finished loading
    AddHandler Application.Idle, AddressOf OnLoaded
End Sub
Protected Overridable Sub OnLoaded(ByVal sender As Object, ByVal args As
EventArgs)
    RemoveHandler Application.Idle, AddressOf OnLoaded
    SplashscreenController.CloseSplash()
End Sub
```

8. Now, edit SplashConfig.cfg. Set details for the Executable file to run, the Splash Picture file to show, whether to show a status label & progress bar. See Appendix 1 for full details.
9. Compile the application. Note how the splash screen files are copied to the output directory. To start the application using the splashscreen, run Splash.exe from the output directory.
10. Now that you have the sample splashscreen up and running, you can customise Splashscreen-Background.jpg to provide your own branding. Edit the SplashConfig.cfg settings to change the location of the status label and progress bar.

.NET 1.1 – Windows forms project

This tutorial shows you how to create the splashscreen for the sample project found at:
Samples\v1.1\CSharp-ProgressBarAndStatusLabel & Samples\v1.1\VB-ProgressBarAndStatusLabel.

1. Create a Windows Forms project.
2. Copy the files from \Splashscreen into the Windows Forms project folder.

Name	Size	Type
ConnectWait.ico	2 KB	Icon
SplashConfig.cfg	1 KB	Microsoft Office Ou...
SplashScreen-Background.jpg	19 KB	JPEG Image
SplashscreenController.cs	1 KB	Visual C# Source file
SplashscreenController.vb	1 KB	Visual Basic Source file
SplashScreen-ProgressBar.bmp	1 KB	Bitmap Image
start.exe	480 KB	Application

If coding in VB, you only need the SplashScreenController.vb file, and visa versa if coding in C#.

3. Make sure that all of these are included in the VS2003 project file. (Project > Show All Files, select files, context menu > Include in project.
4. Ensure that the relevant files are copied to the output directory. In C#, this can be done with a build event. VB does not support build events, so you need to create a .cmd file that you can run after a build.

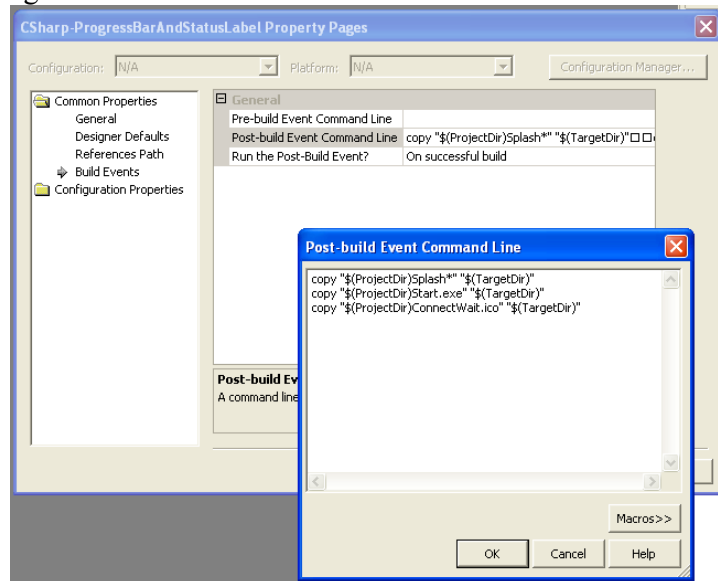
a. Creating a Build Event in C#

Project Properties > Common Properties, Build Events, Post build event command line.

Add the following:

```
copy "$(ProjectDir)Splash*" "$(TargetDir) "  
copy "$(ProjectDir)ConnectWait.ico" "$(TargetDir) "
```

eg:



b. Creating a post build .cmd file in VB.NET

Create a new text file: PostBuild.cmd with the following commands within it:

```
copy "Splash*" "bin"  
copy "ConnectWait.ico" "bin"
```

After building the project, run PostBuild.cmd from the command line to copy the required files to the output directory.

5. Have a quick look at SplashScreenController.cs/vb. This contains functions that are used to control the splashscreen whilst the .NET app is loading.

6. Edit the file containing the main entry point. In C#, this will be the file containing the static Main() method. In VB, this will be the constructor of the first form that is loaded. Add the following line as the first statement in this method.
`Splashscreen.SplashscreenController.UpdateStatusLabel("Initialising");`
 This causes the splashscreen to display “Initialising” in the status label by overwriting the contents of <guid>.lock with “Initialising”.
7. If your application connects to a database, or loads data from some location, you might want to display several status messages using this method.
8. Finally, once your application has loaded, and the main form is displayed to the user, you will want to close down the splashscreen. A good way to do this is to create an event handler in the main form for the Application.Idle event.

Within this event, Call the `SplashscreenController.CloseSplash();` method, which deletes the <guid>.lock file, and causes the splashscreen to close.

- a. In C#:

```
public Form1()
{
    InitializeComponent();
    Application.Idle += new EventHandler(OnLoaded);
}

/// <summary>
/// Custom "OnLoaded" event. Fired after form finished
loaded & displayed
/// </summary>
/// <param name="sender"></param>
/// <param name="args"></param>
protected virtual void OnLoaded(object sender, EventArgs args)
{
    Application.Idle -= new EventHandler(OnLoaded);

    //Shut down the splashscreen
    SplashscreenController.CloseSplash();
}
```

- b. In VB:

```
Public Sub New()
    MyBase.New()
    SplashscreenController.UpdateStatusLabel("Connecting to datasource...")

    InitializeComponent()
    'Trap the idle event - this tells us when the form has finished loading
    AddHandler Application.Idle, AddressOf OnLoaded
End Sub
Protected Overridable Sub OnLoaded(ByVal sender As Object, ByVal args As
EventArgs)
    RemoveHandler Application.Idle, AddressOf OnLoaded
    SplashscreenController.CloseSplash()
End Sub
```

9. Now, edit SplashConfig.cfg. Set details for the Executable file to run, the Splash picture to show, whether to show a status label & progress bar. See Appendix 1 for full details.
10. Compile the application. In C# the splash screen files are copied to the output directory automatically via the build event. In VB you need to do this manually by running the PostBuild.cmd batch file. To start the application using the splashscreen, run Splash.exe from the output directory.
11. Now that you have the sample splashscreen up and running, you can customise Splashscreen-Background.jpg to provide your own branding. Edit the SplashConfig.cfg settings to change the location of the status label and progress bar.

Creating a Setup project using the Splashscreen

A sample VS setup project (Setup-VB-ProgressBarAndStatusLabel) has been included showing how to install the splashscreen along with your application. These instructions are the same for VB / C# projects.

1. Create a standard VS Setup project
2. Ensure that you add both the Primary Output AND the Content Files from the project you are installing. ConnectWait.ico, SplashConfig.cfg, SplashScreen-ProgressBar.bmp and SplashScreen-Background.jpg should all be marked as Build Action: Content (Right mouse in Solution Explorer, Properties, Build Action)
3. Add Splash.exe as a file to the setup project. (Right mouse on setup project, Add, File...)
4. The shortcut in the users start menu must point to Splash.exe, rather than the .NET exe file. So, Right mouse on setup project > View > File System. Users Programs Menu > Create New Shortcut to Application Folder/Splash.exe
5. Compile, and install

Licensing: Removing the Unregistered notice

The trial version of QuickSplash.NET displays an “UNREGISTERED” message in the middle of the splashscreen.

To remove this message, you need to purchase a registration code from <http://www.quicksplash.net>

Registration codes are licensed on a per developer basis, and include access to all minor version upgrades.

Further questions, comments, suggestions

For any further questions, comments or suggestions, have a look at the forums on <http://www.quicksplash.net>

Alternatively, contact us by email at: david@quicksplash.net



Happy developing!

Appendix 1: Configuration file options

Most aspects of the splash screen are controlled via the SplashConfig.cfg file, including:

- The splash picture
- The location & style of the Status label and Progress bar
- The location of the .NET exe file to load
- The license key

The config file is a simple text file, stored in the same directory as the splash screen exe, and named SplashConfig.cfg. It uses simple key:value pairs, one per line. The order of the lines is significant:

Key	Value (example)	Description
Reg Code:	123456789123456	Unique 30 digit license key – removes the “Unregistered” notice
Executable:	MyDotNetApp.exe	Location of the main .NET exe to load in the background
EXE Startup Check:	ON / OFF	When set to OFF, this ignores the check for the Executable: file. When ON, QuickSplash will throw an error if the Executable: file cannot be found.
Lock File Path:	C:\Temp\lockfiles\	Specifies the location that the lockfiles should be created in. The user running Splash.exe should have write permissions in this folder. .\ --use same folder as Splash.exe .\lockfiles\ -- uses lockfiles\ folder relative to Splash.exe C:\Temp\ -- uses absolute folder C:\Temp\
Splash Picture:	SplashScreen-Background.jpg	The JPG to use as a background for the splashscreen
Progress Bar:	ON/OFF	Whether to show the progress bar
Progress Bar Picture:	ProgressBar.bmp	Graphic to use for progress bar. i.e. 
Progress Bar Bars:	5	Number of times to repeat progress bar picture. Specifying 5 would show: 
Progress Bar Picture Space:	3	Space between progress bar picture graphics.
Progress Bar Vertical:	15	Top, Left co-ordinates of progress bar.
Progress Bar Horizontal:	146	
Progress Bar Width:	391	Width of progress bar
Progress Bar Speed:	0.5	Speed progress bars should move at. Valid values: 0 – 1.
Progress Bar Colour:	\$00FFFFFF	Background colour to progress bar – in format \$00BBGGRR (Alpha Blue Green Red)
Status Bar:	ON/OFF	Whether to show status text
Status Bar Colour:	\$00D8E9EC	Background colour of status text
Status Bar Text Colour:	\$00000000	Text colour of status text
Status Bar Left:	40	Top, Left co-ordinates of status bar.
Status Bar Top:	120	
Status Bar Width:	335	Width of status bar

Appendix 2: Command line options

Arg	Value (example)	Description
-overridelockfileguid	-overridelockfileguid=myuniquestring	Forces the name of the lockfile to be this string – so, Splash.exe –overridelockfileguid=mystring calls MyNetApp.exe –lockfile=”<resolved Lock File Path>\mystring.lock