Developers's guide

Table of Contents

1 Introduction

WebFMX is a solution that allows you to remote FireMonkey applications to the Web. By just adding 1 line of code to the FireMonkey's project, the application becomes dual-platform; Windows and HTML5, thus it can be run as usual on a windows environment, or it can be installed on a WebFMX Server environment and be accessed remotely from any HTML5 compliant Web Browser.

Why WebFMX?

1. Enables you to create dual-platform Windows/HTML5 FireMonkey Apps, effortless.

2. Expands applications availability by delivering them instantly to users anywhere on any device.

3. Reduces dramatically the Total cost of ownership (TCO), by slashing IT costs and simplifying administration avoiding costly virtualization/remoting solutions such as Citrix XenApp® or Microsoft™ RemoteApp.

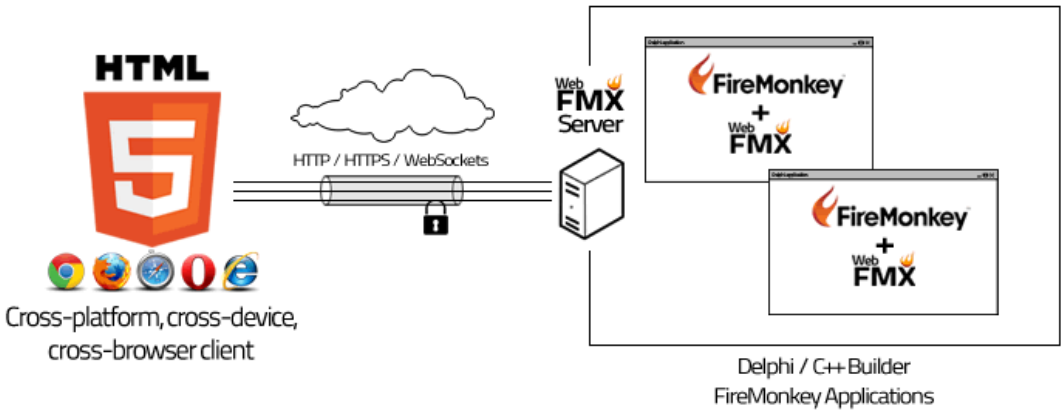
See more:

Architecture
Getting Started
Installing WebFMX
Compiling the application
Registering the application
Accessing the app from the Web
WebFMX Server Manager
Managing the SSL Certificate
Customizing the Web Interface

2 Architecture

WebFMX is composed by:

- WebFMX Client: HTML5 Web Browser
- WebFMX Server: Windows Service
- WebFMX Platform runtime: Runtime unit(s) included in the FireMonkey application



WebFMX Platform runtime is a set of Delphi units that plugs into FireMonkey's framework to redirect Windows calls and drawing commands to the remote HTML5 canvas

Requirements:

Applications

- The application project must use the FireMonkey HD framework.
- Delphi XE3 or Delphi XE4

WebFMX Server

- Windows XP 32-bit / Windows XP 64-bit
- Windows Vista 32-bit / Windows Vista 64-bit
- Windows 7 32-bit / Windows 7 64-bit
- Windows Server 2008 32-bit / Windows Server 2008 64-bit

Web Client

- HTML5-compliant Web Browser

3 Getting Started

To get started, use this section to cover the fundamental aspects of WebFMX. You will learn how to create all the needed configurations in a simple step by step guide so that you can start enjoying the benefits of WebFMX:

1. Installing WebFMX
2. Compiling and testing your application
3. Registering the application in WebFMX Server
4. Accessing an application from the Web

Find a more exhaustive reference of the available options here:

Programming Reference

WebFMX Server Manager

Managing the SSL Certificate

Appendix A - Dialogs

Appendix B - Tailoring the interface

Appendix C - JavaScript API

3.1 Installing WebFMX

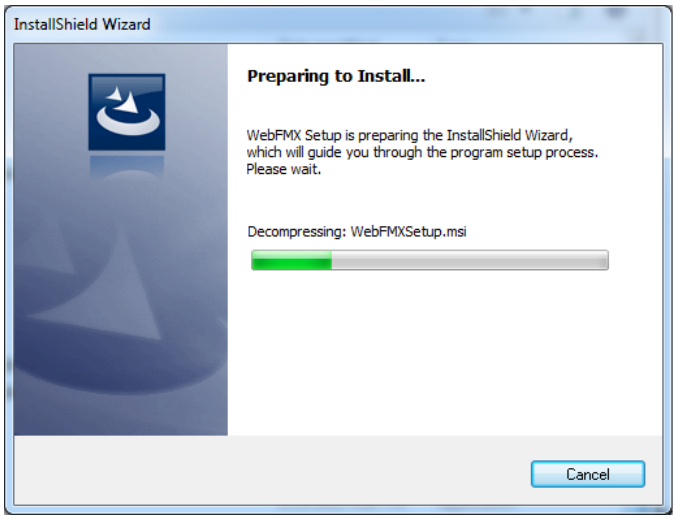
WebFMX can be installed using the setup file provided:

1. Download the installer from one of the links below:

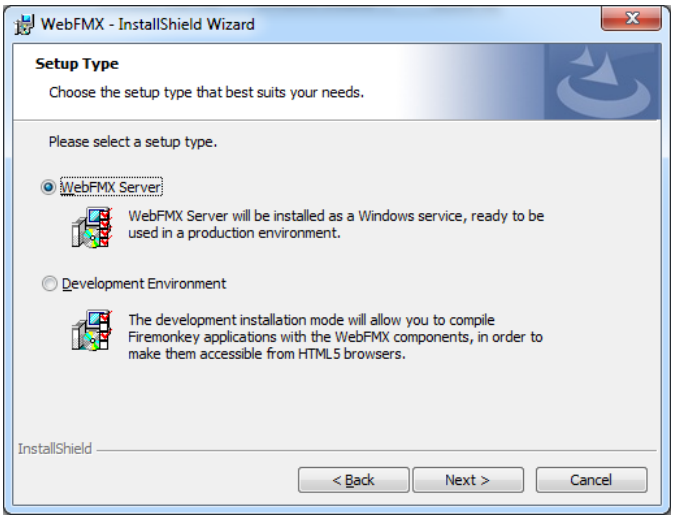
   Exe File
   http://www.cybelesoft.com/downloads/webfmx.exe

   Zip File
   http://www.cybelesoft.com/downloads/webfmx.msi

2. Execute the installer on the target machine.



3. Select type of environment to install:



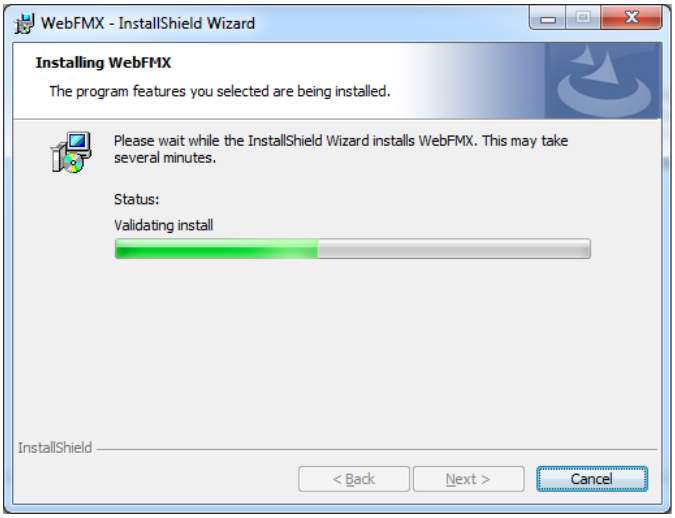
WebFMX Server

This environment is where FireMonkey apps will be run and acceded remotely. This is not needed for development purposes but, it could be installed for testing the application.
The WebFMX Server is a HTTP/WebSocket Server that maintains the communication between the Web Browser and the WebFMX Runtime inside the FireMonkey application.
On this installation mode, the WebFMX Server will be installed as a Windows Service.

Development Environment

This environment is meant to be installed on the developer machine. This mode installs the WebFMX runtime units that you need to include in your FireMonkey application's project. It includes also a WebFMX Server that will execute in a "development mode", to quickly test your application from a Web Browser.

4. Press Next and wait for the installation process to finish. When it is done, press the "Finish" button.



3.2 Compiling and testing your application

In order to create a dual Windows/HTML5 FireMonkey application, you have to compile this application including the WebFMX.Platform.dcu unit. By default, the WebFMX.Platform.dcu unit will be accessible on the installation directory, under the folder below:

```
C:\Program Files (x86)\WebFMX\dev\XE3\win32

C:\Program Files (x86)\WebFMX\dev\XE3\win64

C:\Program Files (x86)\WebFMX\dev\XE4\win32

C:\Program Files (x86)\WebFMX\dev\XE4\win64
```

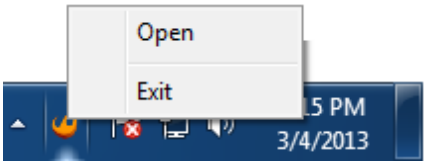
- The source file will only be available when you install using the Development installation mode.
- If you install the Development Environment these paths will be automatically added to the Delphi XE3 and Delphi XE4 Library Paths (32 bits and 64 bits paths).

Follow the next steps to compile your application:

1. Open your Rad Studio or Delphi XE3 or Delphi XE4.
3. Open your application project.
4. Add the unit WebFMX.Platform to the Uses of your project source file.

```
program MyApp;

uses
  FMX.Forms,
  WebFMX.Platform,
  MyApp.Main in MyApp.Main.pas' {Form1};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

5. Compile Your Program and Run it.
6. Observe on the Task Bar that the WebFMX Development Server icon will appear.



7. Right-click on the icon, and after that click on the "Open" menu.

A Web browser window will be opened with your application running inside.

If you don't want the debugging mode to start a new WebFMX Server or want to specify the port in which it will start, you can set change the DevServer property. You may also configure WebFMX not to open a new browser window while running your application.

3.2.1 Registering the application in WebFMX Server

Each application that needs to be accessed through the WebFMX Server, has to be added as an application profile.

We consider you have already Compiled and tested your application with the WebFMX runtime units.

To create an application profile, follow these steps:

1. Open the WebFMX Server Manager, available in the Start Menu.
2. Go to the "Applications" tab.
3. Click on the Add button.
4. Give a name to the application and inform the application path and file name.
7. Press OK and Apply on the Server Manager screen.

Now the application is ready to be accessed from the Web.

3.3 Accessing the app from the Web

Follow the next steps to access registered FireMonkey applications using the Web Browser:

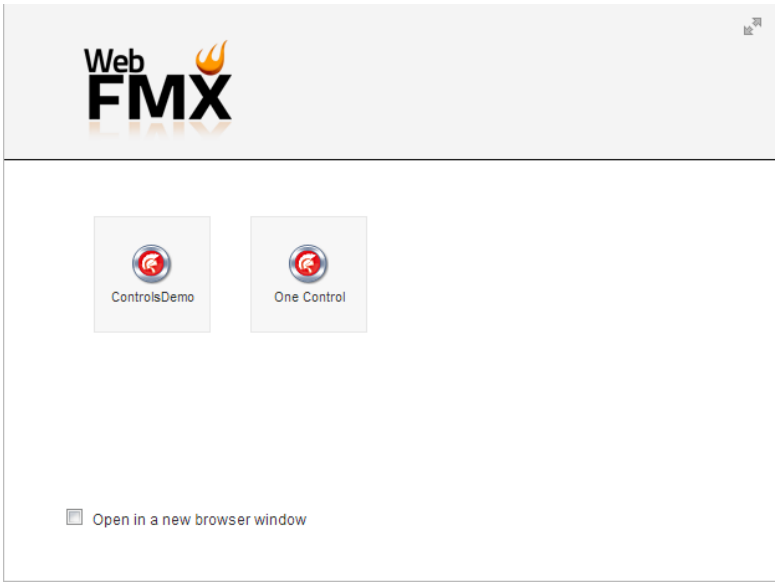
1. Open your preferred Web Browser.

2. Type in the WebFMX Server Address.

3. Inform your username and password, if required.

4. If you have created only one profile application, WebFMX Server will connect you directly to the application.

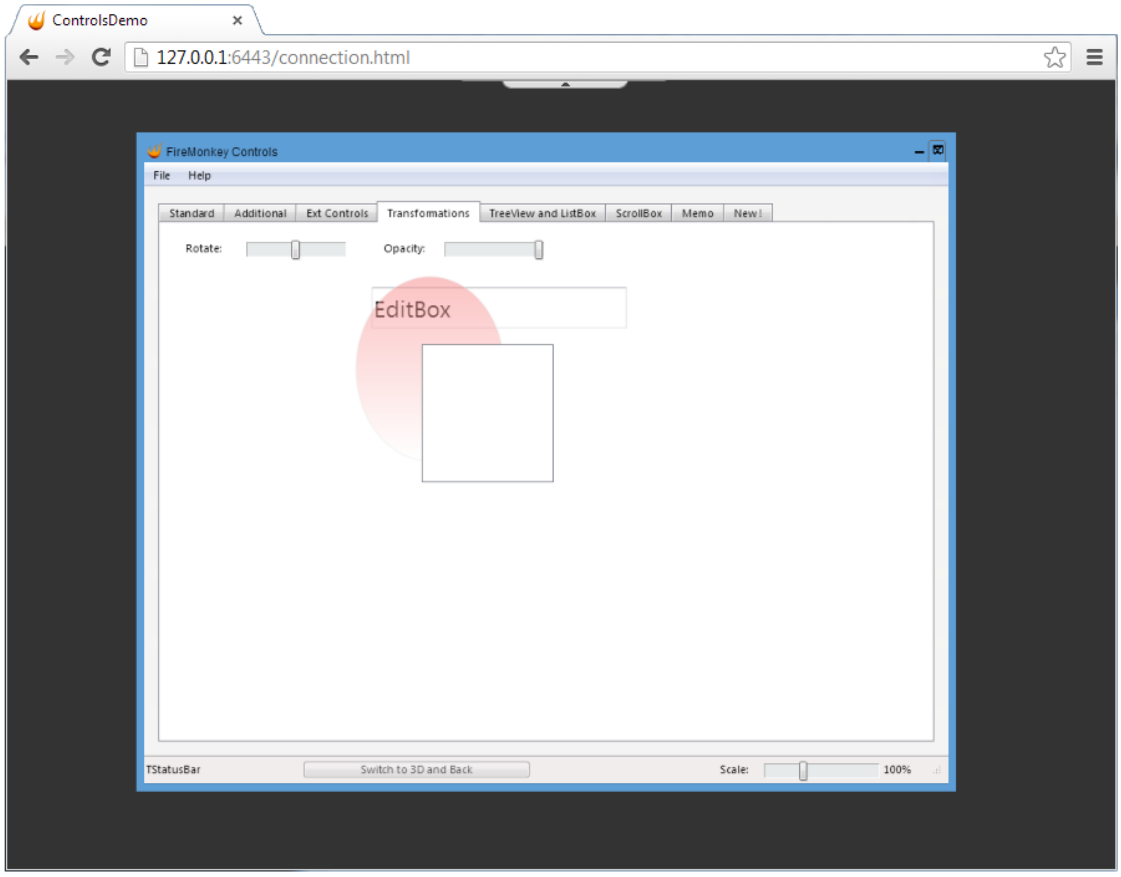
5. If you have created more than on profile application, the Start Page will be presented, so that you can pick one among of the different registered applications.



a. Check the option "Open in a new browser window" if you want the application to be opened in another tab.

b. Click on the application's icon you want to access.

Now you will be able to see and interact with your application from the browser window:



3.4 Application Execution behavior

After you compile your application with the WebFMX.Platform runtime unit, it will get to be started in two platforms: Windows and Web (HTML5).
This topic is intended to let you know the occasions in which each platform will be started as default:

Windows Shell:
When the application is started from the windows shell, it will behave as a normal Windows Application.

Delphi:
When the application is run under Delphi, a development server instance will be started and the Web Platform will be started on a new browser window.

WebFMX Service:
If the application is accessed through the WebFMX Web Server, it will also work through the Web Platform (HTML5).

If you want to disable that the HTML5 platform you can set the PlatformHTML5 Enabled property to false. That way the application will stop responding web requests.

These can be useful during a integration period, in case you want to release the application before you have finished the WebFMX integration, or if you want to debug your application using the Windows platform.

4 Programming Reference

WebFMX provides some programming resources, inside the WebFMX.Platform unit that will allow you to:

1. Retrieve information regarding the client browser and other end user environment variables such as application resolution, authentication information, IP and browser information.

2. Manipulate the application resolution.

3. Manipulate the fonts and WebFonts mapping.

4. Download files.

5. Disable the WebFMX.Platform and have only the Windows platform activated.

6. Enable/Disable the Development Server while debugging the application.

Getting Started

1. Add the WebFMX.Platform into the uses list of the unit you are working with.

2. Access the global variable PlatformHTML5 and its properties, methods and events in order to access these programming resources.

See also

Throughout the next topics, find a detailed description of all public WebFMX classes, including the main TPlatformHTML5.

4.1 TPlatformHTML5

This class provides you access to all public methods, properties and events for interacting with some of the WebFMX settings and behaviors.

Properties

RemoteInfo
DevServer
Enabled
FontMode

Methods

AddDefaultWebFonts
ClearWebFonts
DownloadFile
ResizeRemote
RegisterWebFont

Events

OnAppTerminate
OnBrowserResize
OnSupportsWebFonts

Remarks

The PlatformHTML5 global variable holds a TPlatformHTML5 class object that is automatically instantiated every time your application is executed.

See also

Find on the TRemoteInfo and TDevServer topics more information regarding the RemoteInfo and DevServer properties.

4.1.1 Properties

4.1.1.1 RemoteInfo

The RemoteInfo object allows you to retrieve information regarding the end-user environment as well as to manipulate some of its settings.

Delphi Syntax

```
var RemoteInfo : TRemoteInfo;
RemoteInfo := Object.RemoteInfo;
```

Remarks

Every time you run your application, the PlatformHTML5 object's RemoteInfo property will be loaded with the user environment settings and all these values will be kept updated by WebFMX Server.

See also

Read the DevServer topic for settings regarding the Development Server.

4.1.1.2 DevServer

The DevServer object allow you to configure the Development Server Behavior.
The Development Server is the server which is started while you debug your application with WebFMX Runtime Units.

Delphi Syntax

```
var DevServer : TDevServer;
DevServer := Object.DevServer;
```

Remarks

Every time you run your application, the PlatformHTML5 object's DevServer property will be loaded with the Server default values.

See also

You may also find useful the Application Execution behavior topic.

4.1.1.3 Enabled

Enables/disables the WebFMX Platform. By default this property is set to true, if you set it to false, the application will not be accessed from the web.

Delphi Syntax

```
Object.Enable [:= Boolean];
```

Remarks

When enabled, the applications uses WebFMX platform when:

a) The application is run under Delphi
b) The application is accessed through the WebFMX Web Server.

When the application is started from the windows shell, it will behave as a normal Windows Application.

See also

See also The Application Execution behavior topic.

4.1.1.4 FontMode

There are some Windows fonts that are not supported or do not exist in Web format. Through the FontMode property you may configure how WebFMX should solve this fonts issues:

Delphi Syntax

```
Object.FontMode [:= TFontMode ];
```

See also

Read also the TFontMode topic, in order to understand how each mode will work.

4.1.2 Methods

4.1.2.1 AddDefaultWebFonts

This method adds default font mappings for non-supported WebFonts that are known by WebFMX Server.

Delphi Syntax

```
Object.AddDefaultWebFonts;
```

Remarks

So far, the known non supported fonts configured by default on WebFMX Server along with its mappings are:

Segoe UI	Open+Sans:400italic,700italic,400,700
Segoe UI Light	Open+Sans:300italic,300

See also

Read also the TFontMode topic, the OnSupportsWebFont event topic and the ClearWebFonts and RegisterWebFont method topics.

4.1.2.2 ClearWebFonts

This method clears all default and registered WebFont mappings.

Delphi Syntax

```
Object.ClearWebFonts;
```

Remarks

Once you call this method to clear all the default and registered WebFont mappings, you should either set the TFontMode to fmBitmap or fmAuto or also Register new WebFont mappings through the RegisterWebFont method.

See also

You may find useful the TFontMode and FontMode topics, the OnSupportsWebFont event topic and the AddDefaultWebFonts and RegisterWebFont method topics.

4.1.2.3 DownloadFile

This method downloads to the remote machine an indicated file.

Delphi Syntax

```
Object.DownloadFile ( FileName: String );
```

Arguments

FileName	The complete file path to be downloaded.	String

Remarks

The File Upload will work automatically, without having to call any method.

See also

See also about the ResizeRemote method.

4.1.2.4 ResizeRemote

This method resizes the remote application to an indicated width and height value.

Delphi Syntax

```
Object.ResizeRemote ( width, height: Integer );
```

Arguments

width	The customized value to resize the application width in pixels.	Integer
height	The customized value to resize the application height in pixels.	Integer

See also

See also about the OnBrowserResize event and the RemoteInfo property.

4.1.2.5 RegisterWebFont

This method registers a new WebFont mapping.

Delphi Syntax

```
Object.RegisterWebFont ( FontName, FontEquiv, FontGoogle: String) );
```

Arguments

FontName	This is the Windows font name to be mapped with a WebFont.	String
FontEquiv	This is the name that will be used to register the new font.	String
FontGoogle	This is the Google WebFont URL queryString. For more information regarding it, read this google reference.	String

See also

You may also find useful the TFontMode and FontMode topics, the OnSupportsWebFont event topic and the AddDefaultWebFonts and ClearWebFonts method topics.

4.1.3 Events

4.1.3.1 OnAppTerminate

This event is fired whenever the application is going to be terminated.

Delphi Syntax

```
Object.OnAppTerminate := MyOnAppTerminate;

procedure MyOnAppTerminate( Sender: TObject );
begin
  // Your code here
end;
```

Arguments

Sender	The Sender object.	TObject

Remarks

When this event is fired the application termination can not be undone anymore.

4.1.3.2 OnBrowserResize

This event is fired whenever the end-user resizes the browser window.

Delphi Syntax

```
Object.OnBrowserResize := MyBrowserResizeEvent;

procedure MyBrowserResizeEvent(Sender: TObject; var AHandled: Boolean);
begin
  AHandled := True;
  // Your code here
end;
```

Arguments

Sender	The Sender object.	TObject
AHandled	Prevents WebFMX from resizing automatically the application, when the ScreenResolution property is set to "srFitToBrowser" or "srFitToScreen".	Boolean

See also

Read also the ResizeRemote and the RemoteInfo topics.

4.1.3.3 OnSupportsWebFont

The OnSupportsWebfont event is fired when a browser connects to the application. Inside this event you can describe how WebFMX should handle the fonts mapping.

Delphi Syntax

```
Object.OnSupportsWebfont := MySupportsWebFont;

procedure MySupportsWebFont(Sender: TObject; var ASupports: Boolean);
begin
  ASupports := True;
  // Your code here
end;
```

Arguments

Sender	The Sender object.	TObject
ASupports	Indicates whether the browser should support WebFonts or not. If set to false, the FontMode will be set to fmBitmap.	Boolean

Remarks

In order to find out the kind of browser is connecting to the application, you should read the RemoteInfo.UserAgent property.
The code below is used as a default behavior that allows WebFMX to disable WebFonts for versions under 6 of iPad, iPod and iPhone.
You can use it as an example to right down your own code:

```
 iOS := (Pos('iPad',RemoteInfo.UserAgent)>0) or (Pos('iPod',RemoteInfo.UserAgent)>0) or
(Pos('iPhone',RemoteInfo.UserAgent)>0);
 if iOS then
 begin
   MajorVersion := StrToInt(Copy(RemoteInfo.UserAgent,Pos('Version/',RemoteInfo.
  UserAgent)+8,1));
   FSupportsWebFont := (MajorVersion>=6);
 end;
```

See also

Read also the TFontMode, FontMode, AddDefaultWebFonts ClearWebFonts and RegisterWebFont topics.

4.2 TRemoteInfo

The TRemoteInfo class holds the properties with information regarding the end-user remote environment.

Properties

Width
Height
BrowserWidth
BrowserHeight
ScreenWidth
ScreenHeight
Username
UniqueBrowserId
PeerIP
UserAgent
ScreenResolution

Remarks

Whenever you run your application, a TRemoteInfo object is instantiated automatically and set as a property of the PlatformHTML5 object.

See also

Read also, about the TPlatformHTML5 DevServer property.

4.2.1 Properties

4.2.1.1 Width

Current width of the application.

Delphi Syntax

```
var width : Integer;
width := Object.Width;
```

Remarks

The width is a read only property. In order to modify the application width, you should use the method ResizeRemote.

See also

You may also read the topics Height, BrowserWidth, BrowserHeight, ScreenWidth, ScreenHeight and ScreenResolution regarding the application resolution.

4.2.1.2 Height

Current height of the application.

Delphi Syntax

```
var height : Integer;
height := Object.Height;
```

Remarks

The height is a read only property. In order to modify the application height, you should use the method ResizeRemote.

See also

You may also read the topics Width, BrowserWidth, BrowserHeight, ScreenWidth, ScreenHeight and ScreenResolution regarding the application resolution.

4.2.1.3 BrowserWidth

Current width of the end-user browser window.

Delphi Syntax

```
var browserWidth : Integer;
browserWidth := Object.BrowserWidth;
```

Remarks

The browserWidth is a read only property. In order to modify the application width, you should use the method ResizeRemote.

See also

You may also read the topics Width, Height, BrowserHeight, ScreenWidth, ScreenHeight and ScreenResolution regarding the application resolution.

4.2.1.4 BrowserHeight

Current height of the end-user browser window.

Delphi Syntax

```
var browserHeight : Integer;
browserHeight := Object.BrowserHeight;
```

Remarks

The browserHeight is a read only property. In order to modify the application height , you should use the method ResizeRemote.

See also

You may also read the topics Width, Height, BrowserWidth, ScreenWidth, ScreenHeight and ScreenResolution regarding the application resolution.

4.2.1.5 ScreenWidth

Width of the end-user screen.

Delphi Syntax

```
var screenWidth : Integer;
screenWidth := Object.ScreenWidth;
```

Remarks

The screenWidth is a read only property. In order to modify the application width, you should use the method ResizeRemote.

See also

You may also read the topics Width, Height, BrowserWidth, BrowserHeight, ScreenHeight and ScreenResolution regarding the application resolution.

4.2.1.6 ScreenHeight

Height of the end-user screen.

Delphi Syntax

```
var screenHeight : Integer;
screenHeight := Object.ScreenHeight;
```

Remarks

The screenHeight is a read only property. In order to modify the application height, you should use the method ResizeRemote.

See also

You may also read the topics Width, Height, BrowserWidth, BrowserHeight, ScreenWidth and ScreenResolution regarding the application resolution.

4.2.1.7 Username

WebFMX authenticated username.

Delphi Syntax

```
var userName : String;
userName := Object.UserName;
```

Remarks

The UserName is a read only property.

See also

Read the Permissions topic to learn how to give users permission to access your application.

4.2.1.8 UniqueBrowserId

UniqueBrowserID identifies an instance of a Web Browser. Each time an end-user opens the application from a different browser window, this ID will have a different value.

Delphi Syntax

```
var browserID : String;
browserID := Object.UniqueBrowserID;
```

Remarks

The UniqueBrowserID is a read only property.

See also

Read the UserAgent property topic, to learn how to retrieve more information about the end-user web browser.

4.2.1.9 PeerIP

IP address of the end-user.

Delphi Syntax

```
var peerIP : String;
peerIP := Object.PeerIP;
```

Remarks

The PeerIP is a read only property.

See also

Besides the end-user IP, you may have more information regarding the end-user browser through the UserAgent property.

4.2.1.10 UserAgent

User-Agent of the browser that have been used to open the WebFMX application.

Delphi Syntax

```
var userAgent : String;
userAgent := Object.UserAgent;
```

Remarks

The UserAgent is a read only property.

See also

Read the UniqueBrowserID property topic, to learn how to retrieve more information regarding the web browser.

4.2.1.11 ScreenResolution

This property holds the Resolution configured on the application profile.

Delphi Syntax

```
var resolution : String;
resolution := Object.ScreenResolution;
```

Remarks

WebFMX allows you to configure different resolutions for the Web interface as it is explained on the application profile settings.
The possible resolutions include "Fit to browser window" and "Fit to screen". These two options will adjust the application resolution according to the browser/screen size. The other resolution option are fixed values for the width and height of the application surroundings, such as "640X480" and "800X600". When you select one of these fixed resolution values for your application, WebFMX will not resize automatically the application when an end-user resizes its browser window.

In order to modify the application resolution, you should use the method ResizeRemote.

WebFMX also provides the OnBrowserResize event that will be fired whenever the end-user resizes its browser window.
Through this event, you will be able to customize the application resizing behavior. You may use the environment variables provided on the RemoteInfo object, such as BrowserWidth, BrowserHeight, ScreenWidth and ScreenHeight in order to find out the user environment dimensions.

See Also

You may also read the topics Width, Height, BrowserWidth, BrowserHeight, ScreenWidth and ScreenHeight regarding the application resolution.

4.3 TDevServer

The TDevServer class holds the properties that allows you to modify some of the Development Server settings.

Properties

Enabled
Port
StartBrowser

Remarks

Whenever you run your application, a TDevServer object is instantiated automatically as a property of the PlatformHTML5 object. Read the DevServer property topic, for more information.

See also

Read also, about the TPlatformHTML5 RemoteInfo property.

4.3.1 Properties

4.3.1.1 Enabled

Allows you to disable the development server that runs whenever you compile an application with the WebFMX.Platform runtime unit.

Delphi Syntax

```
Object.Enabled [:= Boolean];
```

Remarks

The Enabled is a read/write property.

See also

If you were looking to disable the whole PlatformHTML5, read this topic: Enabled.

4.3.1.2 Port

Allows you to change the port on which the Development Web Server will answer.

Delphi Syntax

```
Object.Port [:= Integer];
```

Remarks

The Port is a read/write property.

See also

Read also the Enabled and StartBrowser topics.

4.3.1.3 StartBrowser

Configures whether the server will open a new browser window when running the application from the development environment.

Delphi Syntax

```
Object.StartBrowser [:= Boolean];
```

Remarks

The StartBrowser is a read/write property.

See also

Read also the Enabled and Port topics.

4.4 TFontMode

The TFontMode enumerates the possible Font Modes that WebFMX works with.

Values

fmWebFont	This mode sets the application to use Web Fonts. WebFMX automatically maps some not supported fonts and you are able to map others, by calling the method RegisterWebFont for each one of them.
fmBitmap	This mode will send each letter as a small png image and will store them into the web browser cache. This will ensure maximum fidelity, but it can be sometimes a little slower than the WebFonts mode.
fmAuto	On this mode, WebFMX will use WebFont or Bitmap based on the result of the OnSupportsWebFont event. Inside the event you should indicate the environments where the WebFonts should be enabled and the other ones where WebFMX should work with the Bitmaps mode.

See also

Read also about the FontMode property, the OnSupportsWebFont event and the methods AddDefaultWebFonts, ClearWebFonts and RegisterWebFont.

4.5 TScreenResolution

The TScreenResolution data type enumerates all the possible application resolution modes:

Values

srCustom	The width and height of the application will be custom values that will be indicated separately.
srFitToBrowser	The width and height of the application will be the same of the end-user browser window. Whenever the end-user resizes the remote browser window WebFMX will resize automatically the application width.
srFitToScreen	The width and height of the application will correspond to the end-user screen width and height. Whenever the end-user resizes the remote browser window WebFMX will resize automatically the application height.
sr640x480	The width of the application will be set to 640 pixels and the height to 480 pixels.
sr800x600	The width of the application will be set to 800 pixels and the height to 600 pixels.
sr1024x768	The width of the application will be set to 1024 pixels and the height to 480 pixels.
sr1280x720	The width of the application will be set to 1280 pixels and the height to 720 pixels.
sr1280x768	The width of the application will be set to 1280 pixels and the height to 768 pixels.
sr1280x1024	The width of the application will be set to 1280 pixels and the height to 1024 pixels.
sr1440x900	The width of the application will be set to 1440 pixels and the height to 900 pixels.
sr1440x1050	The width of the application will be set to 1440 pixels and the height to 1050 pixels.

sr1600x1200	The width of the application will be set to 1600 pixels and the height to 1200 pixels.
sr1680x1050	The width of the application will be set to 1680 pixels and the height to 1050 pixels.
sr1920x1080	The width of the application will be set to 1920 pixels and the height to 1080 pixels.
sr1920x1200	The width of the application will be set to 1920 pixels and the height to 1200 pixels.

See also

Read also about the OnBrowserResize event, the ResizeRemote method and the ScreenResolution property.

5 WebFMX Server Manager

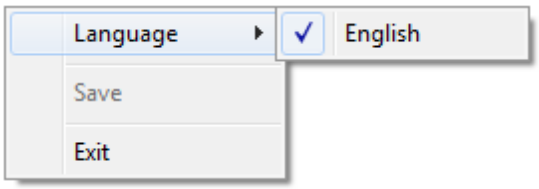
The WebFMX Server Manager is the tool to manage the WebFMX Server, from where you can manage FireMonkey's applications profiles, permissions and settings related to the WebFMX service.
To access The WebFMX Server Manager go over the Start Menu options and look for the "WebFMX *Server Manager*" item.

The WebFMX Server Manager is has the following tabs:

General
Applications
Licences

Its main menu has two sub-menus:

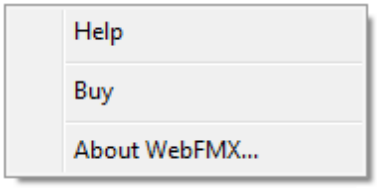
File Menu:



The File Menu is composed by the following options:

Language	Allows you to choose different languages for the application. Click on the Language that you want the application to work with. English is the default language.
Save	Click to save any change done on the system Settings.
Exit	Click on this option to exit the WebFMX Server Manager.

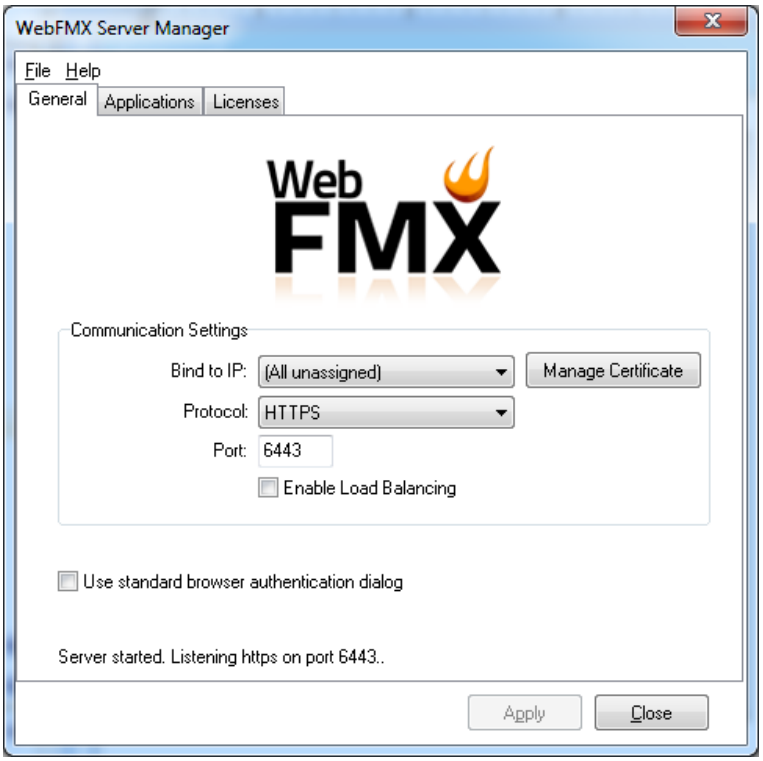
Help Menu:



The Help Menu is composed by the following options:

Help	Takes you to the application online guide.
Buy	Takes you to the Cybele Software's Buy page.
About WebFMX	Click on the About to see the application version and build number.

5.1 General



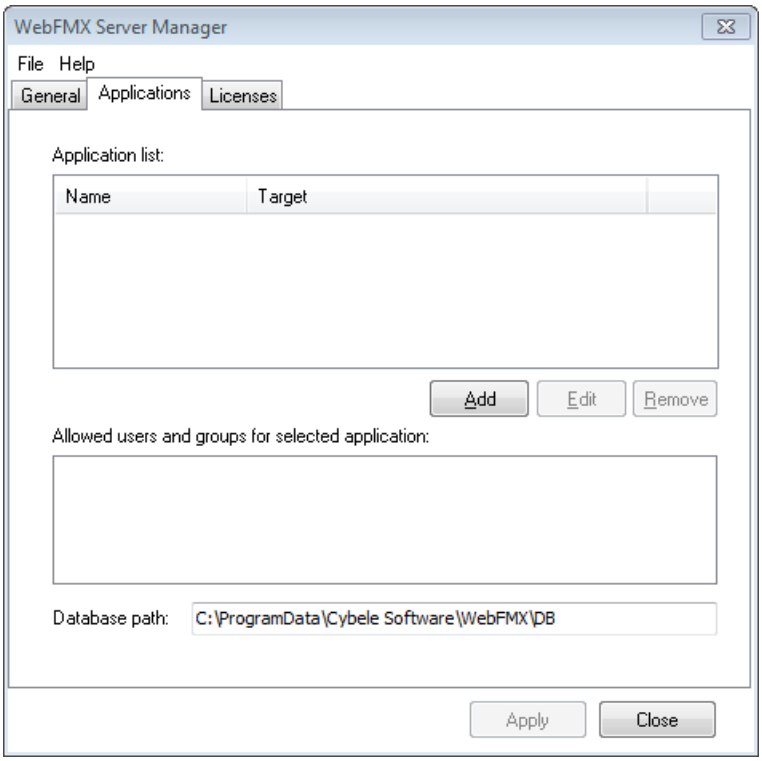
On WebFMX manager "General" tab you will find the following options:

Bind to IP	Use this option to restrict access to the service to one specific IP address. The "All unassigned" option allows access through all the available IP addresses.
Protocol	Choose between the http and https protocol.
Manage Certificate	Allows you to specify the SSL certificate. WebFMX Server already comes with a self-signed certificated, that is good enough for testing purposes.
Port	Choose which port will WebFMX Server be listening on. If the port is not available, you will see an error message on the status bar.

Always remember to press "Apply" in order to save the changes.

5.2 Applications

The "Applications" tab will allow you to configure the FireMonkey applications locations and settings as well as the user permissions to access them.



On WebFMX Server Manager's "Applications" tab you will find the following options:

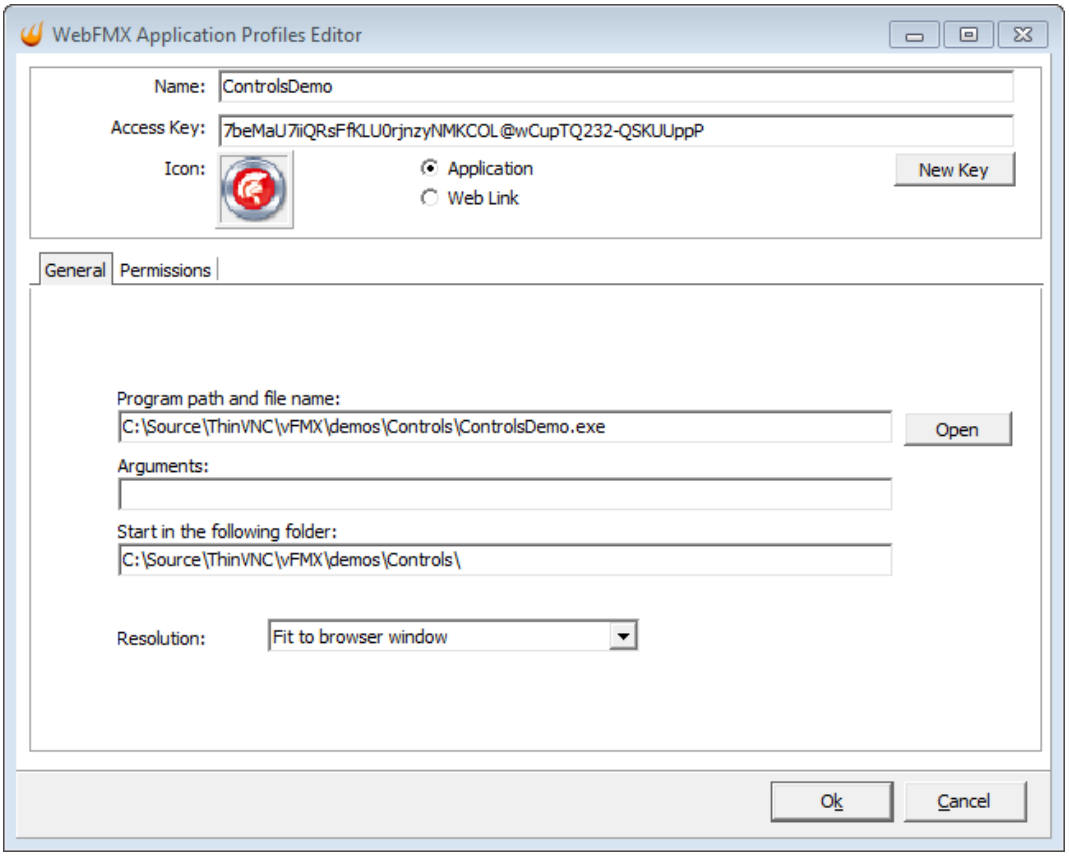
Application List	This list shows the available applications. You can enable or disable them by checking the box to the left of the name. **Name** — Name of the application. **Target** — The application path and the web address in case of the Web Link profiles.
Add	Press this button to add a new application.
Edit	Select an application and press this button to edit it.
Remove	Select an application and press this button to remove it.

Allowed users and groups for selected profile	See here the allowed users or group(s) of users for the selected application. If you want to change the permissions, edit the application.
Database path	When the application is set to work with Load Balancing, you can set a common database path to all WebFMX Brokers by informing it on this field.

Always remember to press "Apply" in order to save the changes.

5.2.1 Application profile

When you edit or add an application profile you will be presented with this screen below. The radio button "Application" must be checked.



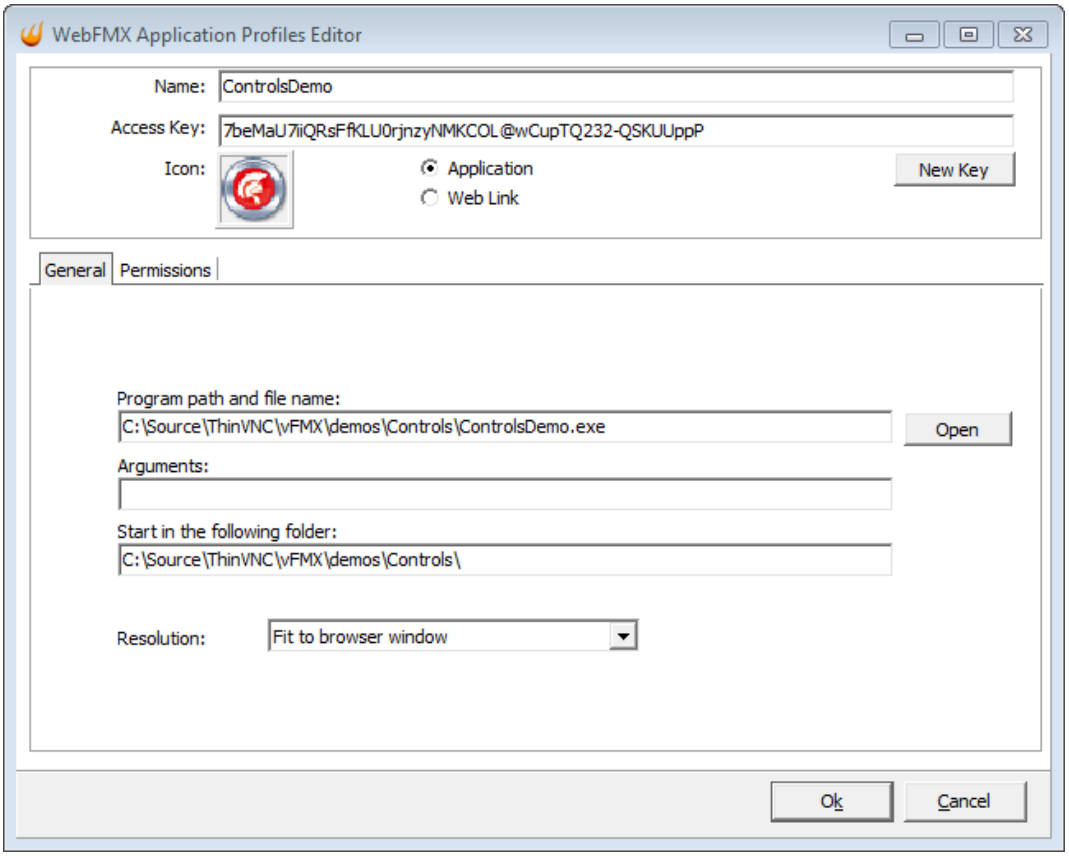
These are the profile properties you can edit:

Name	Use this field to change the application name.
Access Key	This Access Key identify the application within WebFMX Server. It is used when you want to debug your application, for example.
New Key	This button will change the Access Key and disable access through the current key. A new access key will be provided.
Icon	Click on the Icon gray box to load an image to be associated with the profile. The image will be presented along with the profile name on the web interface profiles selection.
Application /Web link	Select the Application option to have a regular profile that gives access to a FireMonkey application. If you select the Web link radio button, this profile will behaves like a Web Hyperlink.

The properties located inside the tabs will be described throughout the next subtopics.

5.2.1.1 General

On the application profile editor's "General" tab you will find these following options:

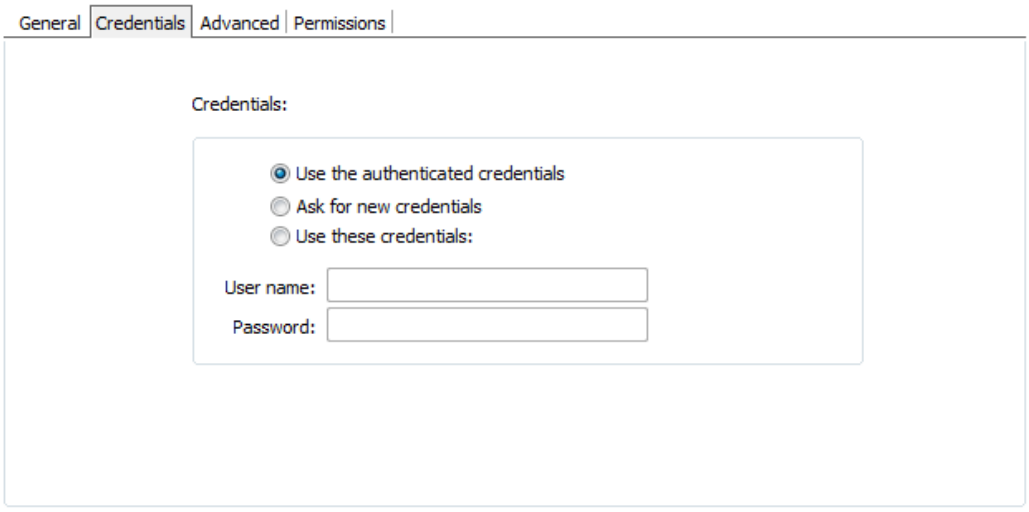


Program path and file name	Specify the complete path that gives access to the application executable file.
Arguments	Applications arguments
Start in the following folder	Inform a context directory for the application set on the field "Program path and file name"
Resolution	Choose from the available list of resolutions including "Fit to browser window" and "Fit to screen", ideal for hiding the browser and working on a full screen mode.

Idle Timeout	Set a timeout in minutes if you want WebFMX Server to wait this period before killing the application once the browser has been closed. Timeout 0 will kill the application immediately after the browser has been closed.

5.2.1.2 Credentials

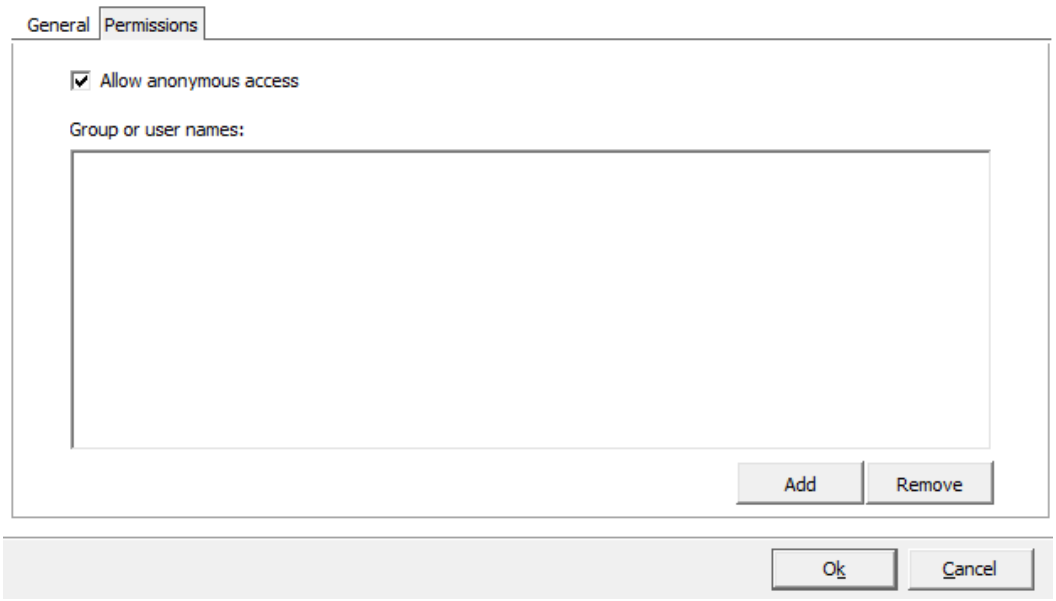
On WebFMX's application editor "Credentials" tab, you should inform the mode for logging into the specified application:



Use the authenticated credentials	Use the same credentials entered in the browser for WebFMX (specified in the "Permissions" tab). Note: If the credentials are correct, this option will connect the user automatically when selecting the application, or after authenticating for WebFMX if this is the only profile for their credentials.
Ask for new credentials	Prompt the user for new credentials to access the computer.
Use these credentials	Complete the credentials used to access the computer. Note: If the credentials are correct, this option will connect the user automatically when selecting the application, or after authenticating for WebFMX if this is the only profile for their credentials.

5.2.1.3 Permissions

Here you need to select the users that will access this application. If you don't select any user, this application will not be accessed.
These are the options you will find on the application profile editor "Permissions" tab:

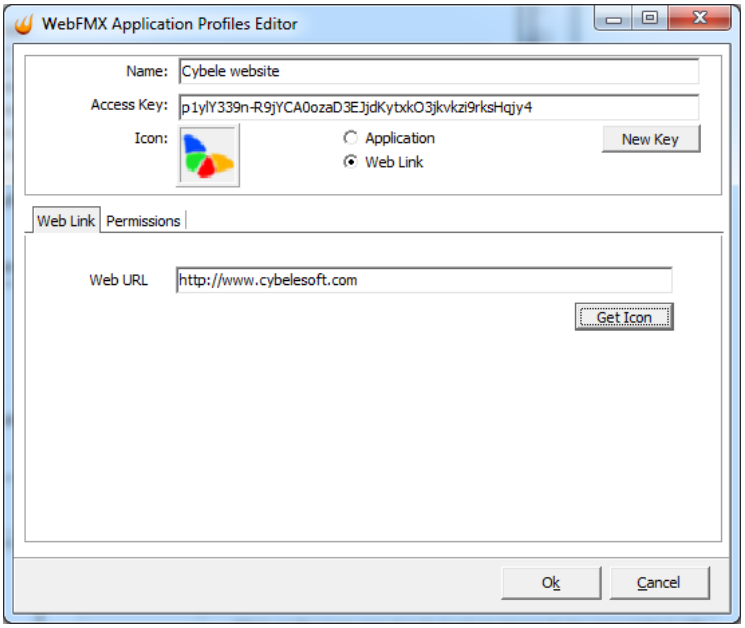


Allow anonymous access	Check this option to make this application available without any authentication. Use this option, if you want this profile to be available for everyone. This means that everybody accessing WebFMX will have access to this application. Checking this option will disable the Add and Remove buttons.
Add	Press "Add" to access the windows dialog for selecting Active Directory users.
Remove	Press "Remove" to remove a user for this profile.

If you want a user or a user group to access more than one application, you need to create more application profiles and then add this user to each profile.
The authenticated user will be able to choose from the available application profiles on the Web interface.

5.2.2 Weblink profile

When you edit or add a Web Link profile you will be presented with this screen below. The radio button "Web Link" must be checked.



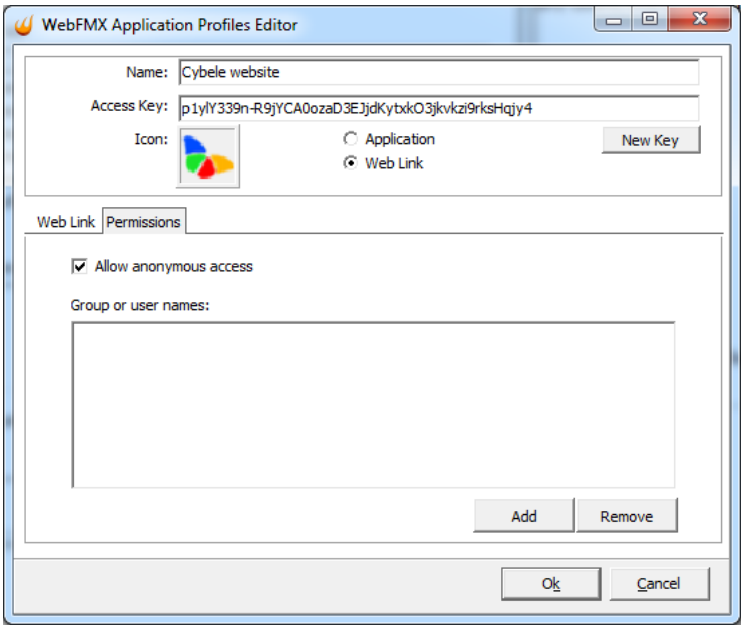
These are the profile properties you can edit:

Name	Use this field to change the profile name.
Access Key	Used in combination with WebFMX Javascript API to access this profile.
New Key	Change the Access Key to disable access through the current key and provide access through a new one.
Icon	Click on the Icon gray box to load an image to be associated with the application. The image will be presented along with the application name on the web interface.
Web link / Application profile	Select the Weblink option to have a profile that connects to a Web link. These links will be shown along with all the other applications on the WebFMX start page.
Web URL	Inform in this field the URL that you want this application profile to connect to.

The properties located inside the other tabs will be described throughout the next subtopics.

5.2.2.1 Permissions

Here you need to select the users that will access this application profile. If you don't select any users, this profile will not be available from the Web interface.
These are the options you will find under the "Permissions" tab:

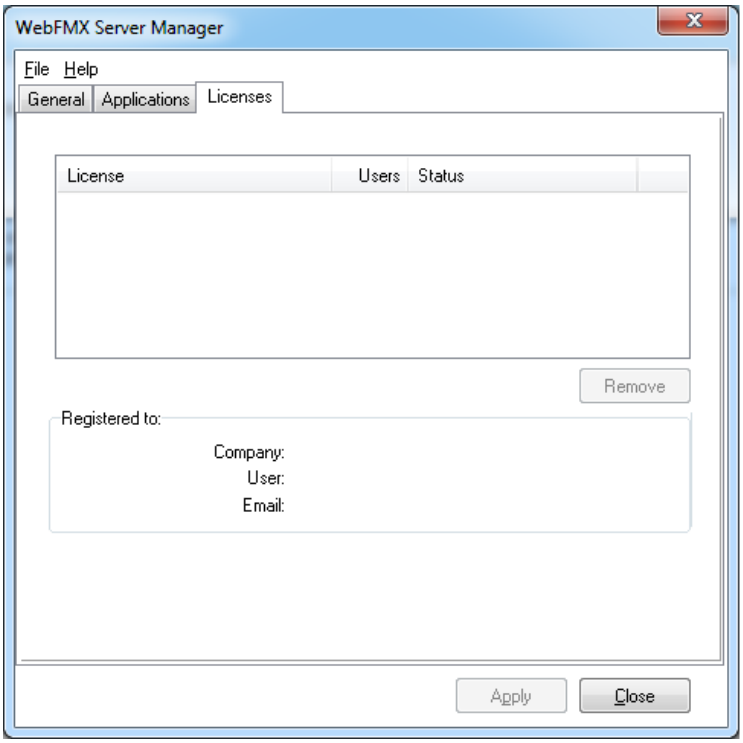


Allow anonymous access	Check this option to make this application available without any authentication. Use this option, if you want this profile to be available for everyone. This means that everybody accessing WebFMX home page will see this profile. Checking this option will disable the Add and Remove buttons.
Add	Press "Add" to access the windows dialog for selecting Active Directory users.
Remove	Press "Remove" to remove a user for this application profile.

If you want a user or a user group to access more than one application, you need to create more profiles and then add this user to each profile.
The authenticated user will be able to choose from the Web interface which application s/he will connect to.

5.3 Licenses

On the WebFMX Manager "Licenses" tab you will find the following options:



This tab always shows the licenses you have currently installed. If you don't have a license yet, you will see a message letting you know how many evaluation days you have left until the trial finishes.
Contact us regarding pricing and/or licensing questions.

6 Managing the SSL Certificate

An SSL certificate is an effective way to secure a website against unauthorized interception of data. At its simplest, an SSL Certificate is used to identify the website and encrypt all data flowing to and from the Certificate holder's Web site. This makes all exchanges between the site and its visitors 100 percent private.
A valid SSL certificate is included with the WebFMX Server installation and all communications are already encrypted with the product's default certificate. You may want to create your own certificate to identify your company better.

Managing the SSL Certificate

1. There are two forms of creating your own SSL certificate:

   a. Create A self-signed certificate

   b. Use A CA Certificate

2. Once you already have your certificate files, go to WebFMX Server Settings "General tab".

3. Click on the "Manage Certificate" option.

4. On this screen you should inform the location of the certificate files, as follows:

Certificate File
Inform the path to the certificate file.

CA File
If the certificate is issued by a unknown CA, you should fill in the pathname to the CA certificate.

Private Key
You should inform the pathname to the certificate private key file.

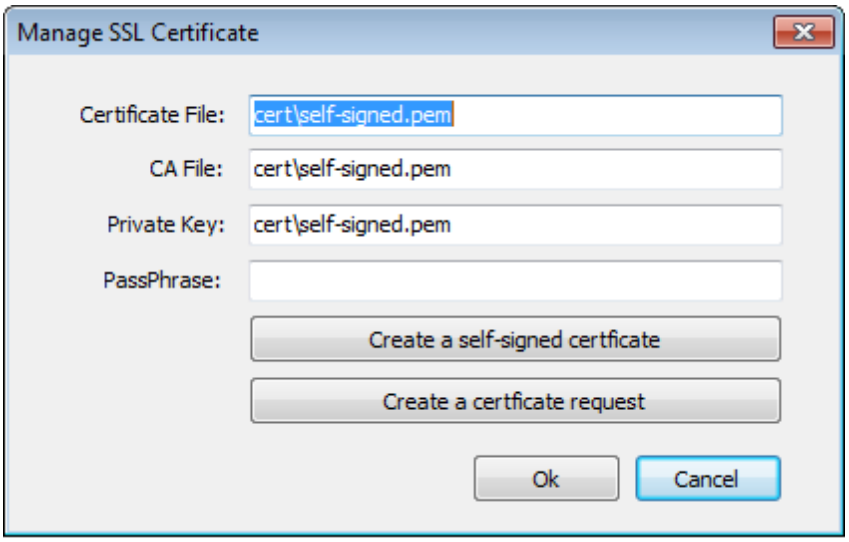
Pass Phrase
Inform the password, if there is any, used when the private key was generated.

Note: The path names can be absolute (C:\MyCertPath\UserThisCert.pem) or relative to the path where WebFMX Server is installed (\cert\UserThisCert.perm).

6.1 The Default Embedded Certificate

A certificate called "self-signed.pem" is included with the WebFMX Server installation. You will find it inside the \cert directory, located inside the WebFMX Server application path.

If you want to use this default certificate you should have the files set as the image below:



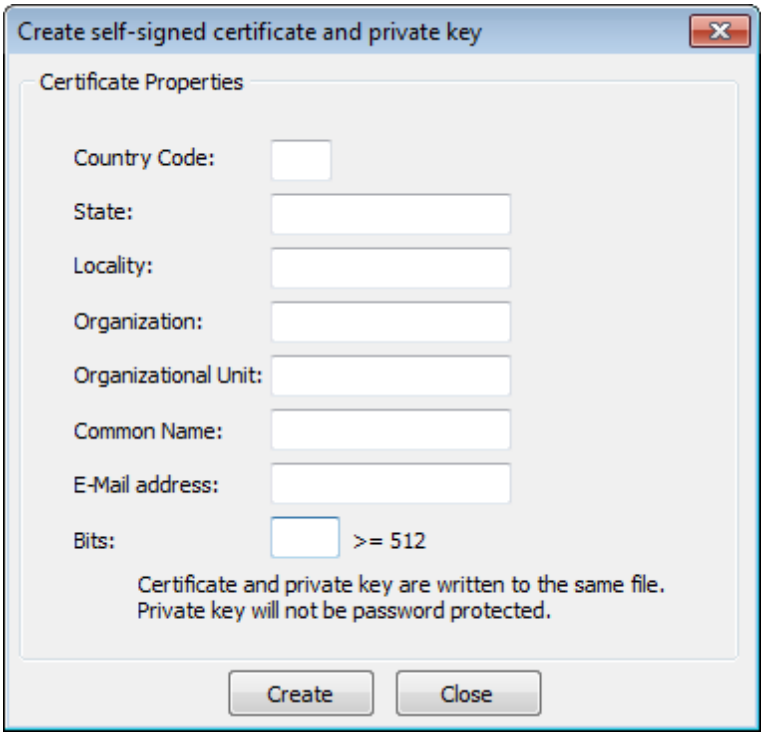
You'll find these settings inside the WebFMX Server Manager 'General' tab, by clicking on the 'Manage certificate' button.

Because this certificate is not issued by a known Certificate Authority (CA), the web browsers will warn you they can not verify its authority.

6.2 A Self-signed Certificate

This option is used to create your own self-sign certificate.

1. Go to the WebFMX Server Settings "Security tab".
2. Press the 'Manage certificate' button.
3. Press the "Create a self-signed certificate" button.
4. Fill in the form below with your organization data:



5. The "Common Name" field should be filled with the server+domain that will be used to access the WebFMX Server (webfmx.mycompany.com).
6. Press Create.
7. Select the location where you want the certificate to be stored.
8. The application will start using this self-signed certificate just created by you.

Because this certificate is not issued by a known Certificate Authority (CA), the web browsers will warn you they can not verify its authority.

6.3 A CA Certificate

In order to use this option you will have to get a certificate from a known Certificate Authority (CA). Some CA examples are GoDaddy, VeriSign, Thawte, GeoTrust and Network Solutions.

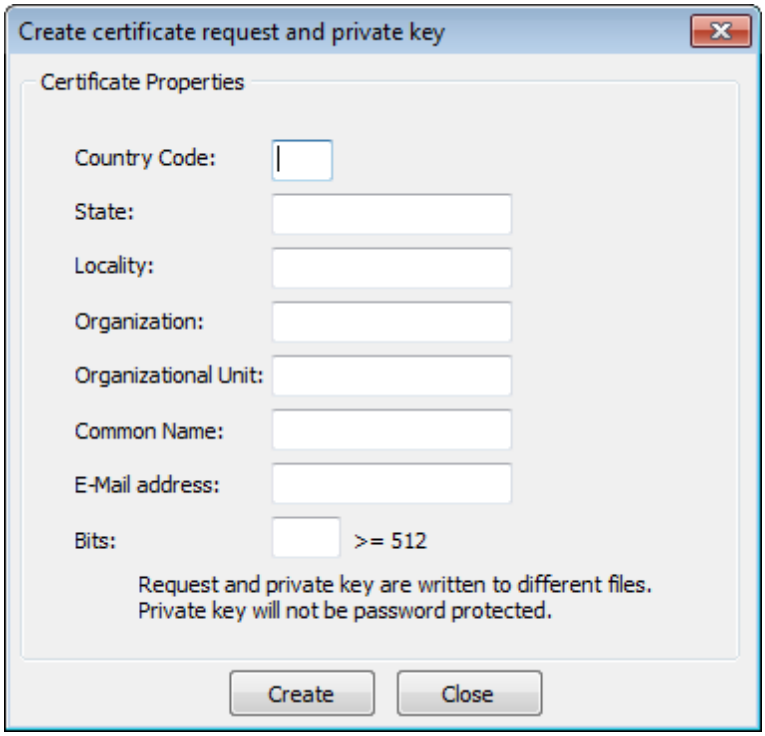
The CA will ask you for a "certificate request". Create one following the next steps:

1. Go to the WebFMX Server Settings "Security tab".

2. Press the 'Manage certificate' button.

3. Click on the "Create a certificate request" button.

4. Fill in the form below with your organization data:



5. The "Common Name" field should be filled with the server+domain that will be used to access the WebFMX server (webfmx.mycompany.com)

6. Press "Create" and the application will generate two files.

7. The first window will ask you a location to keep the private key file: "Where do you want the private key file to be stored".

   a. Inform a name for your private key.
   b. Select a place to keep it safe.
   c. Press the "Save" button.

8. The second window will ask you a location to keep the request file: "Where do you want the request file to be stored.".

a. Inform a name for the request file.
b. Select a directory where you can find the file later on to send to the CA.
c. Press the "Save" button.

9. The first file is the certificate private key. It should always be kept safe with you.

10. Send only the request file to the CA.

After the CA validation process, place the certificate they sent to you on WebFMX Server cert directory and inform the path to the files on WebFMX Server Manager, Manage Certificate option (Certificate file, CA file and Private Key).

7 Appendix A - Dialogs

In WebFMX, window frames have a "web-native" interface. The same happens with standard dialog boxes, that are translated to HTML dialogs.

Find on the next topics how each one of these dialogs will be shown on the Windows and HTML5 platforms:

7.1 Message Dialogs

The Message Dialogs implemented in WebFMX are:

MessageDlg
InputBox
Formatted Message

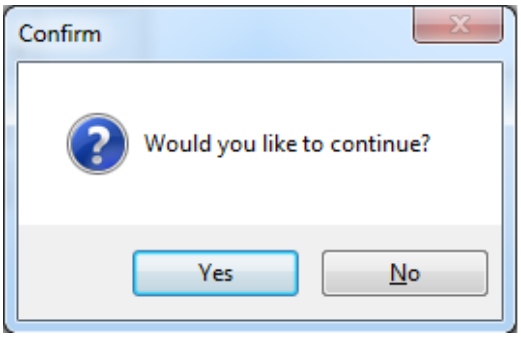
7.1.1 Message Dlg

Usage Example

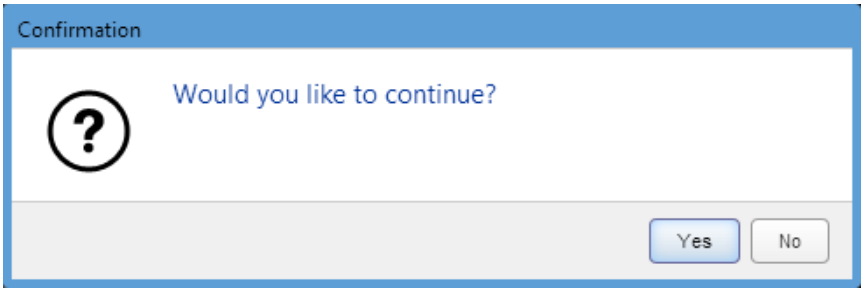
```
MessageDlg ('Would you like to continue?',mtConfirmation,[mbYes,mbNo],0);
```

This is how the dialog is shown on each platform:

Windows Platform



HTML5 Platform



Find on the table below how the Windows MessageDlg Symbols will be presented on the HTML5 Platform:

DialogType	Windows Platform	HTML5 Platform
mtWarning		
mtError		
mtInformation		
mtConfirmation		

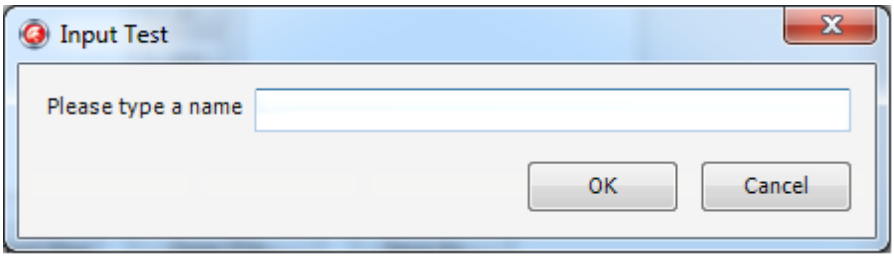
7.1.2 Input Box

Usage Example

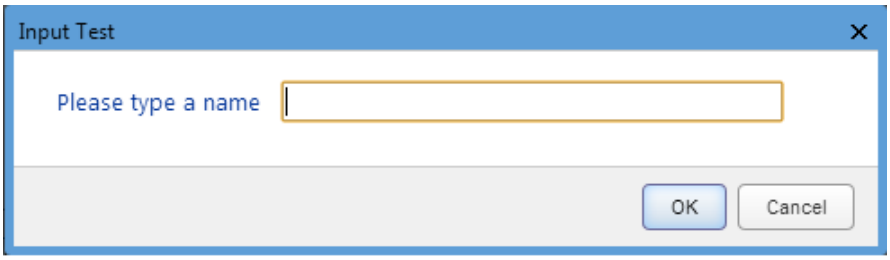
```
value := InputBox( 'Input Test' , 'Please type a name' , '' );
```

This is how the dialog is shown on each platform:

Windows Platform



HTML5 Platform



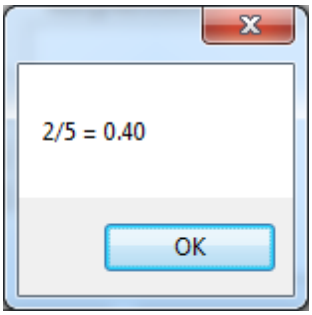
7.1.3 Formatted Message

Usage Example

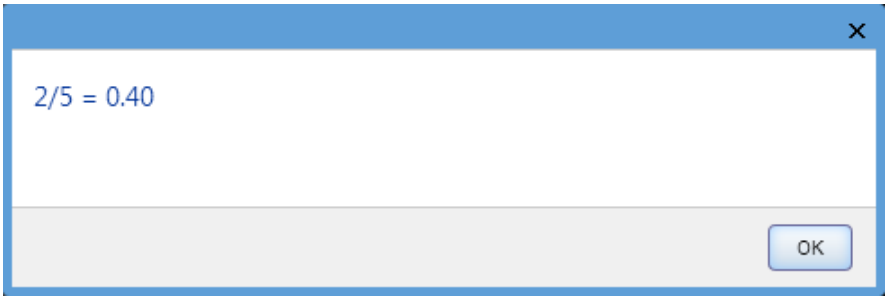
```
ShowMessageFmt( '%d/%d = %1.2f' , [2, 5, 2/5] );
```

This is how the dialog is shown on each platform:

Windows Platform



HTML5 Platform



7.2 Printing Dialogs

The Printing Dialogs implemented in WebFMX are:

PageSetup
Print

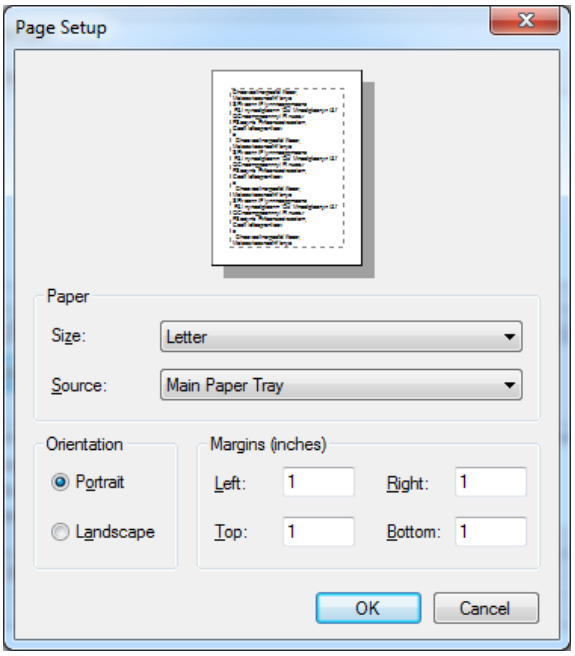
7.2.1 Page Setup

Usage Example

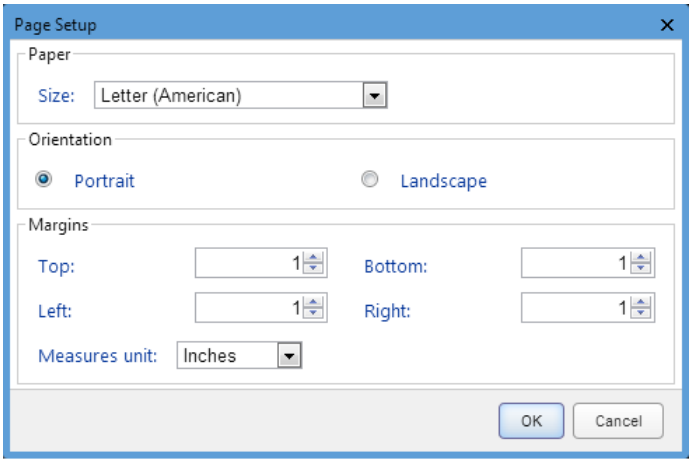
```
type
   PageSetupDialog1: TPageSetupDialog;
...

PageSetupDialog1.Execute;
```

This is how the dialog is shown on each platform:

Windows Platform



HTML5 Platform



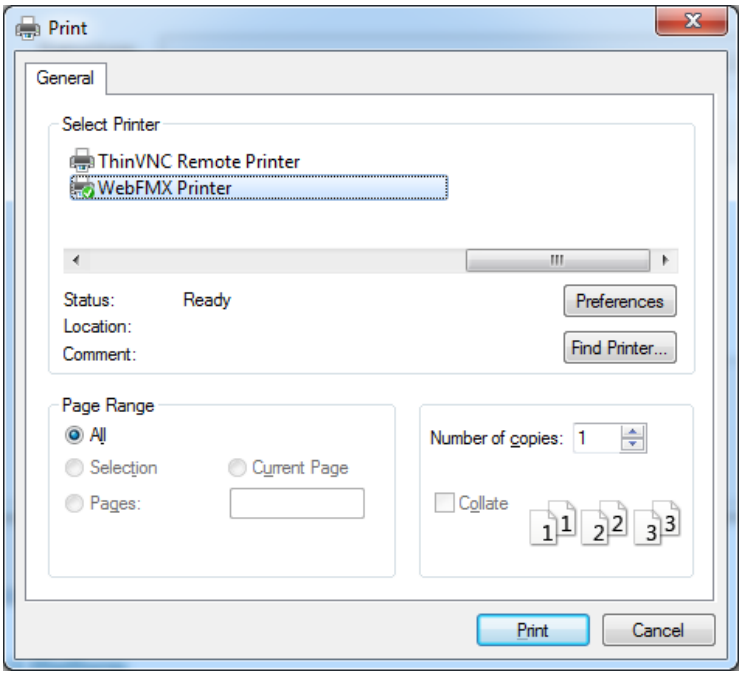
7.2.2 Print

Usage Example

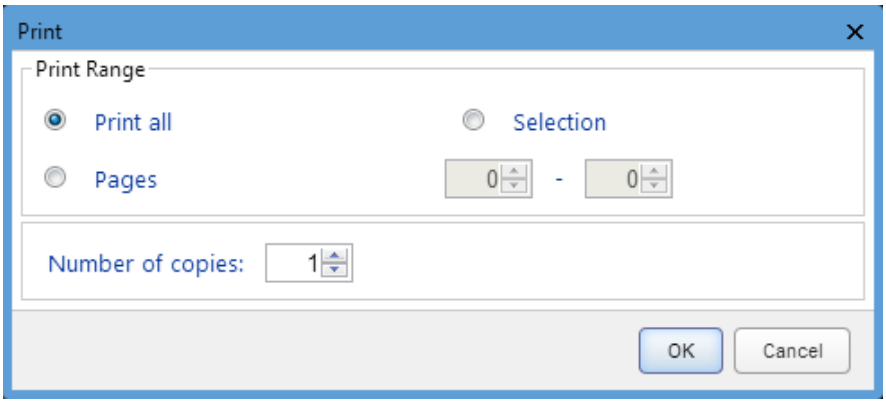
```
type
   PrintDialog1: TPrintDialog;
...

PrintDialog1.Execute;
```

This is how the dialog is shown on each platform:

Windows Platform



HTML5 Platform



7.3 File Dialogs

The File Dialogs implemented in WebFMX are:

Open File
Save As

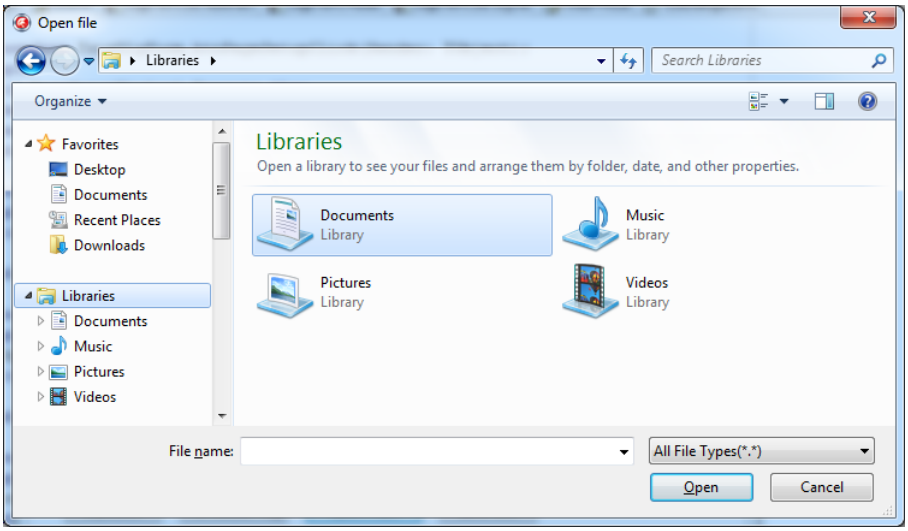
7.3.1 Open File

Usage Example

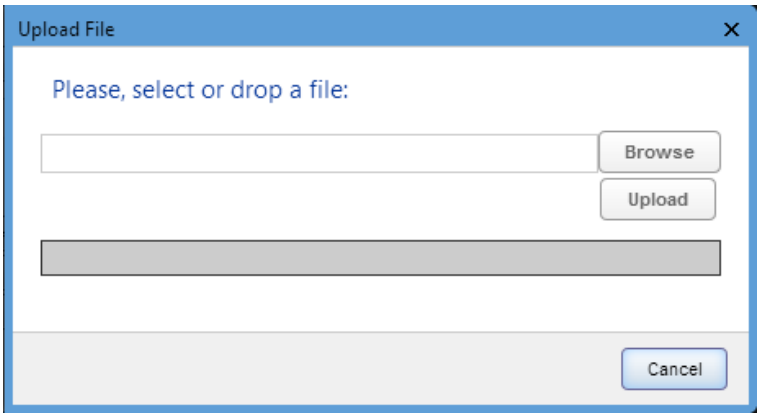
```
type
    OpenDialog1: TOpenDialog;
...

OpenDialog1.Title := 'Open file';
OpenDialog1.Execute;
```

This is how the dialog is shown on each platform:

Windows Platform



HTML5 Platform



In order to open a File from the remote machine the user will be able to make an upload of one local file into the server.

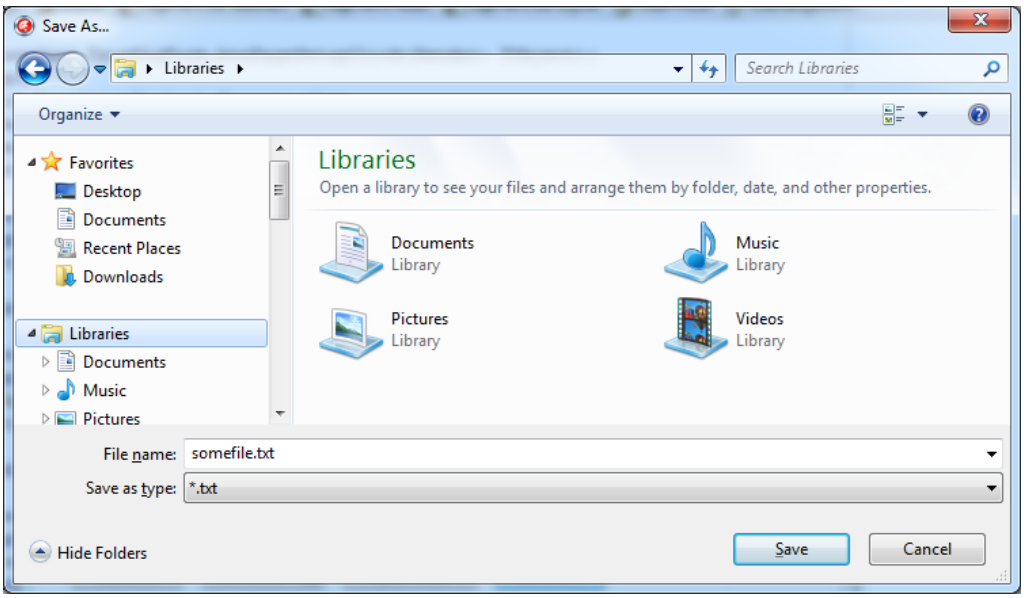
7.3.2 Save As

Usage Example

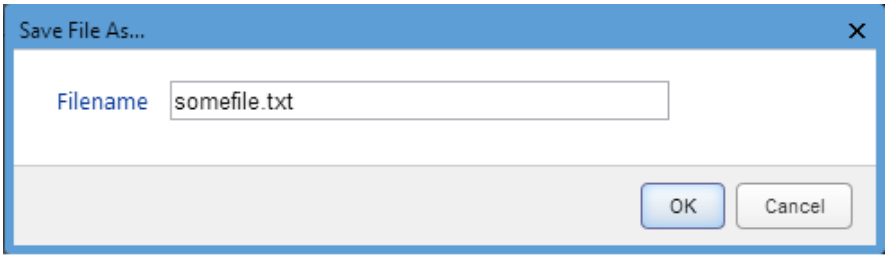
```
type
   SaveDialog1: TSaveDialog;
...

SaveDialog1.Title := 'Save As...';
SaveDialog1.Filter := '*.txt';
SaveDialog1.FileName := 'somefile.txt';
SaveDialog1.Execute;
```

This is how the dialog is shown on each platform:

Windows Platform



HTML5 Platform



In order to save a File from the remote machine WebFMX will do a download into the LocalMachine.

8 Appendix B - Tailoring the interface

8.1 Customizing the Web Interface

WebFMX allows you to modify the web interface and tailor it to your branding scheme.

Customizing the application logo and other image files can be very simple, once it only requires you to have the new image file and tell the application where it is located.

Customizing the structure and style of the application may be a little bit more complex. These kind of customizations have to be done at a programming level (HTML and CSS).

Read also how to protect the customized web files in the Files Location topic.

8.1.1 Changing the logo

Modifying the application logo can be as simple as copying the new logo image and telling WebFMX application where it is located:

1. Create a folder called "BrandingFiles", if it doesn't exist yet, under the folder webfmx located inside the WebFMX installation directory.
   (e.g.: C:/Program Files/WebFMX/webfmx)

2. Copy your own logo image file to the "BrandingFiles" folder.

3. Create the WebAliases.ini file and configure it:

   a. Create a file called "WebAliases.ini" in the installation directory (e.g.: C:/Program Files/WebFMX/WebAliases.ini). If the file already exists, only append the lines to it.

   b. Configure the redirection of the logo files you want to substitute, following the two examples below (webFMX.png.png and favicon.ico):

```
[Alias]

;=================
;Main logo
;=================
/images/webFMX.png.png=BrandingFiles\MyLogo.png

;=================
;Favicon
;=================
/favicon.ico=BrandingFiles\MyFavicon.ico
```

   c. Save it.

4. Open the application to see the changes.

Take into account:

a. Any line in the "WebAliases.ini" file starting with a semicolon will not be considered by the application. It can be used to leave comments in the file.

b. You can substitute any interface image or file, by following the same steps described above.

c. Sometimes the favicon is not shown right the way, because the browser keeps history of the images. In that case, you should clean the browser cache before trying out the changes.

8.1.2 Customizing the web files

To customize the web files, you should:

1. Create a folder called "BrandingFiles", if it doesn't exist yet, under the folder webfmx located inside the WebFMX installation directory. (e.g.: C:/Program Files/WebFMX/webfmx)

2. Make copies of the original web files that you want to modify to the "BrandingFiles" folder. Copy only the files to be modified without their associated folder structure.

3. Customize the files (html, css, etc) as you prefer.

4. Create the WebAliases.ini file and configure it:

   a. Create a file called "WebAliases.ini" in the installation directory (e.g.: C:/Program Files/WebFMX/WebAliases.ini). If the file already exists, only append the lines to it.

   b. Configure the redirection to the files you have modified, by adding a line similar to the examples below for each modified file:

```
[Alias]

/index.html=BrandingFiles\my_index.html
/css/index.css=BrandingFiles\my_index.css
```

   c. Save it.

5. Open the application and check out the changes.

Take into account:

a. Any line in the "WebAliases.ini" file that starts with a semicolon will not be considered by the application. It can be used to leave comments.

b. The paths located in the HTML, CSS, and other contents will be kept relative to the original file location. This means that you won't have to change the content paths when customizing this files.

8.1.3 Files Location

We recommend that you to create a new folder in order to keep the customized files instead of leaving it all together with the original ones. On doing so, you will:

a) Have the possibility to get back to the original interface configuration, at anytime
b) Make sure that your files will be safe after a version upgrade.

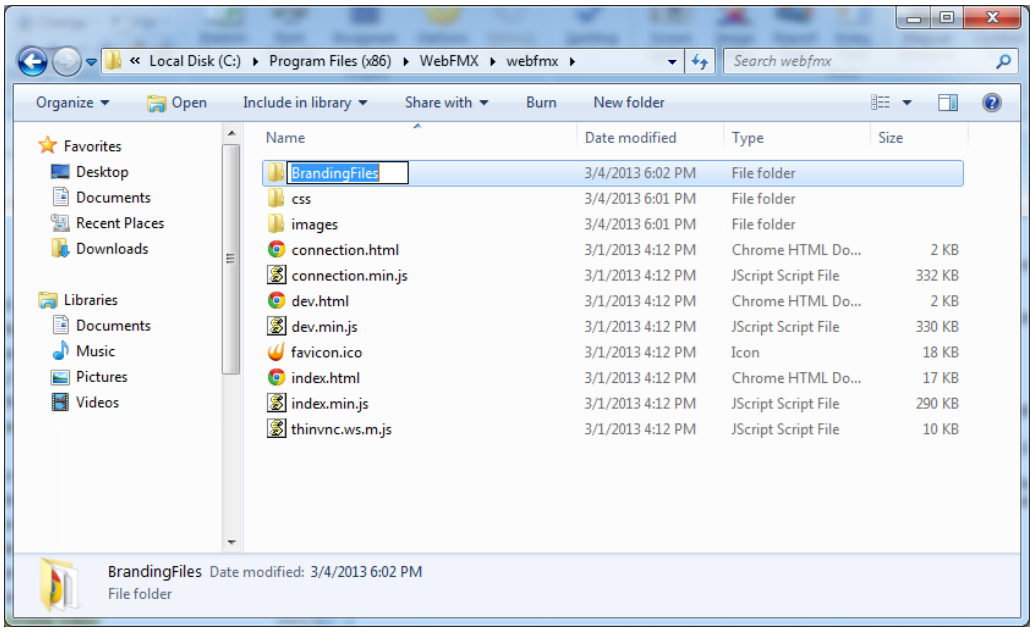
You can also choose whether to place the files inside or outside the webroot structure. Read next, how each option will behave differently.

Inside the webroot :

When the directory that will keep the customized files is created inside the webroot directory:

1) The files will be accessible externally from a URL similar to: https://127.0.0.1/BrandingFiles/customizedFile.html

2) The paths to the files, indicated in the "WebAliases.ini", can be relative to the webroot directory. (e.g. "/img/webFMX.png.png=BrandingFiles\MyLogo.png"). You will find other relative path examples on the topics Changing the logo and Customizing the web files.



Outside the webroot :

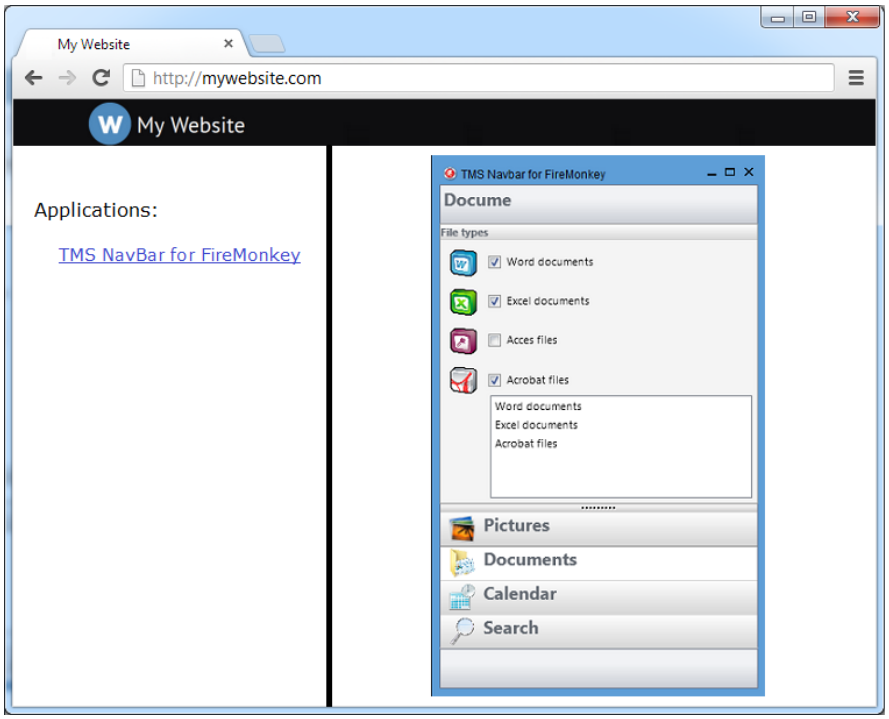
The customized files, can also be placed in any other disk location. In that case:

1) The files will be protected, because it won't be possible to access the customized files from an URL.
2) The paths to the files, indicated in the "WebAliases.ini", have to be absolute, as the example below:

```
[Alias]

/index.html=c:/BrandingFiles/my_index.html
/images/webFMX.png.png=c:/BrandingFiles/MyLogo.png
```

9 Appendix C - JavaScript API

WebFMX's JavaScript API enables you to embed FireMonkey applications into pre-existing Web environments.



In order to integrate WebFMX into your web page you will be required to modify the HTML page by adding some Javascript code.
From this point on, we consider you have already installed and configured WebFMX. If not, go back to the Getting Started topic.

To learn how to use the SDK library, read the following topics:

Deploying

Modifying the HTML file

Connect method

Authentication Scheme

SSL Certificate

Take a look also on the sdk.html file available in the WebFMX server installation directory, under the folder WebFMX. After configuring the profileKey parameter on the connect method, you can try it out from the browser through the address http(s)://server IP:port/sdk.html.

9.1 Deploying

These are the files that should be deployed within your application/website, when using the WebFMX SDK.
All the files can be found inside the application directory under the WebFMX folder.

1. The *sdk.min.js* library
2. The *webfmx.ui.css* and the *popups.css*
3. The whole *css.m* directory. These are the styles for mobile devices.
4. The whole *images/core* directory.

The images and the mobile stylesheets (items 3 and 4 above) must be placed in the same subdirectory tree relative to sdk.min.js location.

9.2 Modifying the HTML file

You need a remote application container in your web page. This can be either:

- A div element
- An iframe element
- A new browser window

You will be able to configure this through the connection mode (step 6b explained further below).

Modifying your html file, step-by-step:

1. Open the HTML page for editing.

2. Add these meta tags into the <head> tag:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="X-UA-Compatible" content="chrome=1"/>
```

3. If you want the WebFMX integration to work with iOS, add the following <meta> tags into the <head> tag.

```
<link rel="apple-touch-icon" href="images/icon.png"/>
<meta name="apple-mobile-web-app-capable" content="yes" />
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0, minimum-scale=1.0, user-scalable=no, target-
densityDpi=device-dpi"/>
```

4. Add a reference to the css stylesheet files into the <head> tag. These files must be deployed with your website/application.

```
<link rel="stylesheet" type="text/css" href="css/popups.css"/>
<link rel="stylesheet" type="text/css" href="css/webfmx.ui.css"/>
```

5. Add the following libraries inside the <head> tag:

a. The jQuery library (jquery.min.js):

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.1/
jquery.min.js" type="text/javascript"></script>
```

b. Point a script tag to the WebFMX SDK client library (sdk.min.js): this file will have to be deployed with your website/application. The tag below should be added to the <head> section, too.

```
<script src="sdk.min.js" type="text/javascript"></script>
```

6. Also inside the <head> tag, add one more <script> tag. The *GetWebFMX* method creates the object that handles the WebFMX SDK functionality. It has two arguments: the WebFMX server URL and the connection mode in which WebFMX SDK will work. The *connect* method is the method that creates the application and positions it on the structure you have configured (div, iFrame, Window).

```
<script type="text/javascript">

      function connect() {

          window.scroll(0, 1);
          var webfmx = GetWebFMX("WebFMX server URL", connection mode);
          webfmx.connect({
              divId: 'webfmx',
              profileKey: 'Substitute with profileKey if using Access
      Profiles'
              }
          });

      };

</script>
```

a. Substitute the "WebFMX server URL" argument, that goes inside the *GetWebFMX* method, with the WebFMX protocol + IP + Port following this format example https://127.0.0.1:8443.

b. Substitute the second argument with the mode you want the connection to be established:

Mode	How it works
Local (remote =false)	The application is embedded in the same page inside a div structure.
Remote (remote=true)	The sdk.min.js posts into WebFMX Server. With this mode you can place the application inside an iFrame or a new window tab.

c. Find out the next sub-topic ("Connect method") how you should complete the arguments that go within the connect method.

6. Code special behaviours on the available WebFMX SDK events.

7. Use the Authentication Scheme to ensure the usernames and passwords security.

9.3 Connect method

The "connect" method creates the application and places it on the specified html structure. In order to do so, it expects a JSON object as argument in which you can inform all the desired settings. If you want to understand exactly how each setting will influence on the connection, read the following topics:

Placement parameters

Application parameters

Settings parameters

Event parameters

Right bellow, you will find the connect method with all the possible parameters set. They should not be sent all together, because each mode and environment will require a different JSON object setup.

- The Placement parameters will be required depending on the connection mode (remote or local).
- The Application parameters will configure which application and the authentication information generated previously.
- The other parameters (Settings and Events) are optional and should be sent whenever you need to change a determined WebFMX behavior.

```
webfmx.connect({

        // Placement
        targetWindow: "substitue with the iframe id or window name",
        exitURL :     "about:blank",
        postpage:     "connection.html",
        divId :       "webfmx",

        // Application
        profileKey:  "substitute with the application Key",
        profilePass: "substitute with the generated password",

        // Settings
        resolution:      "fittobrowser",
        width:           $(window).width(),
        height:          $(window).height(),
        maxWidth:        0,
        maxHeight:       0,
        showToolbar:     false,
        scaled:          false,
        clipboard:       true,
        showToolbar:     true,
        fitToBrowser:    true,

        // Events
        events: {
                onServerConnecting           : function (reconnecting) { },
                onQueryDisconnect            : function () { },
                onServerConnect              : function () { },
                onSessionStart               : function () { },
                onServerConnectionError      : function (errMessage) { },
                onServerDisconnect           : function () { }
        }
});
```

If you are using the SDK Authentication Scheme, you should call the method IsOneTimeKeyValid, before connecting. This method will validate the authentication key and password and open a new client session.

9.3.1 Placement

Find below all the parameters related to the application placement.
Some of the parameters should be sent only when the connection mode is set to Remote and some of them should be sent only when the connection mode is Local.

Parameter	What it means	Type/format	Default	send when mode	
				remote	local
targetWindow	Inform "_self" to have the connection opened over the current window. The "*" value will open a new window with a name assigned by WebFMX. If you inform an existing window name or iframe id, WebFMX will position the connection on this target and if the target does not exist, a new window will be created with the that name.	string "*" , "_self" , target window (iframe id or window name)	"_self"	yes	no
exitURL	Assign an URL to redirect after the connection has closed.	string URL	"about:blank"	yes	no
postpage	This parameter configures the server HTML file. The embedded file name is 'connection.html'. You only have to change this value in case you have customized this file.	string html file name	"connection.html"	yes	no
divId	div id where the remote desktop will be placed, when using local mode.	string div id	"webfmx"	no	yes

9.3.2 Application

These are the parameters related to the application.

Parameter	What it means	Type/format
profileKey	Key that identifies the application to be started. You will find the key information by editing the application on the WebFMX Server Manager Application tab.	string profile key
profilePass	Generated one-time password, used to validate the user from the client side.	string password

9.3.3 Settings

Find below all the settings that can be configured through WebFMX SDK connect method.

Parameter	What it means	Type/ format	Default
resolution	"fittobrowser", "fittoscreen", "fixed", when fixed, the parameters width and height will be considered.	string resolution	"fittobrowser"
width	Application environment width. It will only be considered when the resolution parameter is set to "fixed".	integer pixels	$("#deskdiv").width()
height	Application environment height. It will only be considered when the resolution parameter is set to "fixed".	integer pixels	$("#deskdiv").height()
maxWidth	Maximum width for the application environment. Only available when the setting fitToBrowser below is enabled.	integer pixels	0
maxHeight	Maximum height for the application environment. Only available when the setting fitToBrowser below is enabled.	integer pixels	0
showToolbar	Set false to hide WebFMX toolbar.	boolean true, false	true
scaled	By setting this option, you will have the connection image scaled. The original desktop size will be the maximum limit size applied to the connection. It works only when the fitToBrowser property is enabled.	boolean true, false	false
clipboard	Enables/disables the application clipboard.	boolean true, false	true
fitToBrowser	Set this property to true in order to have the application div adjusted to the web browser size. If this property is set to false, you will have to manage manually the div location and size.	boolean true, false	true

9.3.4 Events

These are the events that can be handled from the WebFMX SDK.

Event	Parameters	When it is triggered
onServerConnecting	reconnecting	This event is fired during the server connection establishment. The reconnecting argument informs whether this is a reconnection or a first-time connection.
onQueryDisconnect	-	Anytime the Web client is about to be disconnected, the "onQueryDisconnect" will be triggered. It is intended to validate with the user if the disconnection is desired.
onServerConnect	-	The "onServerConnect" event is fired every time a "connect" command is exchanged between the browser and the WebFMX Server. It is a way of making sure the server received a sent "connect" command.
onSessionStart	-	This event will be fired when the client session has been started on WebFMX Server.
onServerConnectionError	errMessage	If an error prevents the client connection to be established, this event will be fired. The errMessage argument brings the error message.
onServerDisconnect	-	Anytime the Web client gets disconnected from the WebFMX server, the "onServerDisconnect" will be fired. It could be triggered because the connection was lost incidentally or also because the user disconnected from the server on purpose.

9.4 Authentication Scheme

This SDK is intended to embed a FireMonkey application within an existing website/ application.
In this multi-application context, the authentication process involves two steps that were designed to ensure the security of the exchanged data among the different applications components:

1. Generating an authentication key from the website/application server side
2. Validating this key from the application client side, before starting the FireMonkey application (connect method).

9.4.1 Generating the key

The one-time authentication key is a temporary key generated by WebFMX Server, intended to protect the actual username and password. Both key and password are temporary data and they will be only valid for a single connection and limited period of time.

How it works:

1. First you need to ask WebFMX Server to generate the key and password for you. Call the server following this URL format:

```
http(s)://WebFMXServer:Port/ws/oturl/get?<queryString>
```

2. The queryString should be built with all parameters listed below:

```
username= <username> &password= <password> &plen= <passlen>
&expires= <expires>
```

Find on the table below a description for each required parameter.

Parameter	Description
username	This is the username to be authenticated on WebFMX Server.
password	The password of the user to be authenticated.
plen	This parameter carries the returned unique password length.
expires	Through this parameter you can set an expiration(in minutes) for the key-password pair. Expires = 30 means that the pair won't work anymore after 30 minutes have passed from the pair generation.

3. If WebFMX gets to authenticate with the sent parameters, it will return a JSON object containing the One-Time key/password pair.

Parameter	Description
valid	Use this result to know whether the authentication was successful (username and password were valid).
key	This is the one-time key to be used for the client side authentication.
pass	This is the password to be used for the client side authentication.

4. If the authentication was successful, write on the end HTML file the generated key and password information, so that they can be used for the client-side key/password Validation.

JavaScript example:

This is a JavaScript example. The actual *GetOneTimeKey* method should be translated to the Server side programming language and only invoked from there to avoid username/password exposure on the client side.

```
function GetOneTimeKey(username,password) {
    var otkey =  getOneTimeKey(username, password);
    if (!otkey.valid) {
      alert('Invalid username/password');
    }
    return otkey;
}

function getOneTimeKey (username, password, passlen, expires) {
    var returl = "";
    if (!passlen) passlen = 8;
    if (!expires) expires = 30;
    $.ajax({
        url: "http(s)://WebFMXServer:port" + "/ws/oturl/get?
username=" + username + "&password=" + password + "&plen=" +
passlen + "&expires=" + expires,
        async: false,
        dataType: "html",
        statusCode: {
            200: function (data) {
                returl = eval("("+data+")");
            }
        }
    });
    return returl;
}
```

9.4.2 Validating the key

After you have generated the one-time key and password on the server side and have written them on the HTML file, you should call the IsOnTimeKeyValid method from the client-side. This method checks whether this authentication credentials are valid and opens a new Web client session, using Ajax.

After invoking this method, you will be able to call the Connect method, responsible to create and place the FireMonkey application on the HTML container.

JavaScript example:

```
function ValidateOneTimeKey(key, pass) {
    webfmx.isOneTimeKeyValid(
        key,
        pass,
        function () {
            connect();
        },
        function () {
            alert('error');
        }
    );
}
```

Parameters:

Parameter	Description	Type
key	one-time generated key	String
pass	one-time generated password.	String
success	function to be executed in case the authentication info is validated.	function
error	function to be executed in case the authentication info is not valid.	function

Read also:

Read also the connect method, called in case of the validation success.

9.5 SSL Certificate

When you embed WebFMX into a website and the browser can not verify the configured certificate authenticity, your integration will not work.
There are two ways of handling this issue:

1. Using your own certificate

If you already have your own certificate or if you will get one from a Certificate Authority (CA) , the only thing you will have to do is configure the certificate as described on "A CA Certificate" section.

2. Using the HTTP protocol

Otherwise, you should configure the WebFMX protocol to HTTP. This setting is available on the WebFMX Manager General tab.

Keep in mind you will have to change the protocol on the GetWebFMX server parameter as well.