TAGS REPUBLIC

Table of Contents

1. Introduction

Thank you for purchasing our Data Matrix Reader Software Solution!

The "**Data Matrix Web Package**" : Reader is a software module that will empower your site or rich web application with the possibility of decoding 2D Data Matrix codes inside an online, "in-browser" environment. Our software component suite allows for fast and efficient integration of Data Matrix barcode reading technology into online applications. Expand the way you manage information in your web applications with this innovative approach.

This manual is a setup guide that can be used as a reference when integrating our component into your, or your customers' third-party multiplatform applications / websites.

Yours truly,
The TagsRepublic / Ufo Media Design Team.

Important Notices

Before using our Software, please read the instructions carefully and follow the software package examples in order to obtain the best results.

Please read the conditions from "Rights and Warranties" section : using this Software is conditioned by the reading and accepting of aforementioned terms.

Information presented in this manual refers strictly to the software component / module / package delivered along with the reference document. This information might not be necessary applicable to older / updated versions of our products.

The purpose of this manual is helping developers integrate our components seamlessly by describing the embedding process, component parameters and configuration options. However, it is not dedicated to (focused on) the final Data Matrix Reader application users / consumers.

Ufo Media Design SRL reserves the right to make any changes in new revisions of this manual without any notice.

Requirements

Target System Recommendations

These system specifications are recommended in order to build up a strong end-user experience for applications embedding "DM Web Package Reader" component. They do not represent a minimal requirements specification:

- Webcam capable of 640 x 480 @ 30 FPS resolution with corresponding camera permissions;
- 1.6 GHz “Core” Class CPU / 1GHz Cortex A8 Class CPU for Android deployment.
- 1GB System Memory (RAM) / 512MB RAM for Android deployment;
- Adobe Flash Player 10+ enabled browser or Adobe AIR running on Windows, Linux or OS X / AIR for Android runtime;

Depending on camera stream resolution and Reader configuration, these specifications can be lowered: e.g. 512MB ram, Atom CPU class. The DM reader module should work on any webcam type. It was designed to work mainly with notebook / netbook cameras, but it should also work with extremely cheap webcams. A camera capable of 640x480 capture resolution is recommended for better detection rates. An auto-focus / manual focus camera could be required in order to read smaller barcode prints. On Android devices, the Data Matrix Reader component should be embedded into an AIR application since camera support is not included in Flash Player for Android at the time of writing.

Development System – Additional Software Requirements

- for the JavaScript Package : a local / testing web-server software solution
- for the ActionScript Package : Flex SDK, Flash Professional or Flash / Flex Builder; optionally : Google Android SDK, Adobe AIR SDK

Features and Specifications

- reader core version : 1.9.8
- detects 2D Data Matrix square and rectangular markers;
- reads 1x1, 1x2, 2x2, 4x4 Data Grid ECC200 Data Matrix symbols with / without interleaving;
- decodes error corrected information from : ASCII, C40, Text, Base 256, Edifact, ANSI x12 encoding schemes;
- light on dark contrast code (e.g. white codes on black background) support can be enabled through a dedicated parameter;
- accompanied by usage examples or public method and property specification documents;
- accepts size, camera, timing, smoothing and many other public parameters;

JavaScript Package Component:

- simple and fast deployment in a HTML / JavaScript browser environment;
- JavaScript communication interface; suited for Ajax applications;
- offers methods and properties for starting and stopping the decoding stream;
- returns the decoded string through a JavaScript function call;
- provides an interface for Javascript calls / call-backs;
- comes as a stand-alone, SWF compliant file;
- focused on webpage integration;

ActionScript Package Component:

- targets Flash Professional / Flash Builder / Flex SDK / AIR applications;
- more customization possibilities if embedded properly;
- delivered as a SWC component;
- mobile devices support through Air for Android;

Javascript package

integration

Overview

Our Data Matrix Reader can be easily integrated into any HTML page, similar to any other *.swf / Flash file by using the *<object> <embed>* tags. In order to take full advantage of the Reader's functionality you must also define a few basic JavaScript functions that communicate with the Reader's JavaScript Interface.

Customization of the Reader to your desired look, feel and functionality is accomplished by setting a number of parameters from the basic types. One can start by changing the visualization part and than move on to the more advanced parameters which allow tweaking the decoding process / performance or even the types of codes which can be read.

Please read the following “Setting up” paragraph and try out the examples from the “example” folder of the delivered package to get a better idea on how to use our Reader.

The Flash Component runs under remote Security SandBox type. Testing and using the Data Matrix JavaScript Package : Reader requires a server type environment. Running the Reader locally will not allow access to the component's JavaScript interface due to browser security policies.

Setting up

Defining the Javascript functions. Examples.

Working with the JavaScript interface implies the existence of a *barcodeRead(code)* callback function which is called by the Flash component. If this function is not

defined or doesn't return the “ok” string JavaScript interface calls shall be stopped for the duration of the decoding session. The other two functions [*StopDecoder()* and *StartDecoder()*] are calls from JS to the *.swf module which can be used to obtain the desired behavior from the Reader component.

These function can be defined in a JavaScript file which is imported in the HTML page [*<script src= "./scripts/dmreader.js" type = "text/javascript"></script>*] or directly inside the *<script>* tag.

- *barcodeRead(code)* - this is a JS call function from Flash. The function is called whenever a code is detected. Once a code is detected, the Flash decoder waits for cbi seconds (default *2000ms*) before making another call (only when a code is continuously found). This function must return "ok" in order to have a call confirmation. Optionally, in this time interval the *StopDecoder* function could be called or the decoder can be unloaded / hidden / destroyed / etc. Example:

```
function barcodeRead(code)
{
   document.getElementById("codearea").innerHTML = code;
   // just display the code inside a div with "codearea" id
   return "ok";
   // must return “ok” or else no further calls will be made
}
```

- *StopFlashDeccoder()* - this is a JS callback function JS » Flash. It puts the decoder on hold without destroying it. It only works after the user allows the camera to capture from Flash (Adobe Flash Player settings » Allow). The decoder is started automatically once webcam rights are granted. Example:

```
function StopFlashDecoder()
{
   GetSwfDecoder("dmreader").StopDecoder();
   // pauses the decoder;
   // uses the GetSwfDecoder() function defined bellow;
}
```

- *StartFlashDecoder()* - this is a JS callback function JS » Flash. Starts the decoder if it was put on hold by *StopDecoder*. It only works after the user allows the camera to capture from Flash (Flash Player settings » Allow). Example:

```
function StartFlashDecoder(code)
{
   GetSwfDecoder("dmreader").StartDecoder();
   // resumes the decoding process;
}
```

- *GetSwfDecoder(swfName)* - gets the embedded element object. Simple and very useful function to get a reference to swf object by it's name (the swfName function parameter).

```
function GetSwfDecoder(swfName)
{
   if (window.document[swfName])
      return window.document[swfName];
   // other browsers
   if (navigator.appVersion.indexOf("MSIE")!=-1)
      return document.getElementById(swfName);
   // IE browsers

   return document[swfName];
}
```

Setting the parameters.

All parameters can be set when you embed the Reader inside a HTML page. They are passed through the string that sets the *value* of the *<param>* tag with *name="movie"* and the *src* property of the *<embed>* tag.

The string is formatted in the following way:
"DMReaderSwfURL?parameter01=value01&parameter02=value02 ..."

- *DMReaderSwfURL* refers to the address where the Reader is located relative to the page / app root. All available parameters, alongside their default values (when they are not explicitly set) are defined and described in the Basic and Advanced Parameters chapters.

- Besides these parameters you can also set the actual (visual) *width* and *height* of the Reader inside the web page via the *width*/*height* properties of the *<object>* and *<embed>* tags. The Reader will scale and crop automatically according to these properties.

HTML Embedding Example.

- Here is a simple HTML embedding example setting a few simple parameters (*cw* – camera width, *ch* – camera height, *deti* – detection interval).

- The object's *classid* property must always be set in order to maintain JavaScript interface functionality (as in our example).

```
<object classid = "clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
          width = "640"
         height = "360"
          align = "middle">

   <param name = "quality"           value = "high" />
   <param name = "scale"             value ="noscale" />
   <param name = "salign"            value ="lt" />
   <param name = "bgcolor"           value ="#000000" />
   <param name = "allowScriptAccess" value ="always" />
   <param name = "movie"
         value = "../reader/dmreader.swf?cw=320&ch=240&capi=40"/>

   <embed src     = "../reader/dmreader.swf?cw=320&ch=240&capi=40"
          bgcolor = "#000000"
          width   = "640"
          height  = "360"
          quality = "high"
          align   = "middle"
          type    = "application/x-shockwave-flash"
          allowScriptAccess = "always"
          pluginspage="http://www.macromedia.com/go/getflashplayer"
   />
</object>
```

actionscript

package

integration

Overview

Our The "Data Matrix ActionScript Package" : Reader is dedicated to ActionScript developers requiring a robust DM barcode reading solution. As opposed to our JavaScript Package, component embedding and customization is done directly by using a development and deployment environment native to our software module.

Whether you work with Flash Professional CS4+, Flash / Flex builder, Haxe or with the Flex SDK command line compilers, you can easily integrate the DMReader module just like any other *.swc component. Our module targets different enviroments : Flash , AIR Applications, AIR for Android and can be deployed across multiple platforms.

Please read the following “Setting up” paragraph and take a look over the sample source code from the delivered package in order to get a better idea on how to use our Reader.

Setting up

Development Environment.

Working with the The first step when integrating the Data Matrix Reader component into any ActionScript project is represented by the library embedding process.

The examples delivered along the ActionScript package address two different environments : Adobe Flash Professional for design oriented developers and the Flex SDK framework for open source oriented development.

- When integrating the Reader into a Flash project, you should add the module to your project by specifying the path to the library:

1. Open your **.fla* project
2. Go to the *File Menu* > *Publish Settings*
3. *Publish Settings* > select the *Flash* tab
4. *Flash* tab > click on the *Settings* button
5. *Settings* > select the *Library Path* tab
6. Click on the "*plus button*" (*add new path*) and add the path to the "*DMReaderLib.swc*" library

- When integrating the Reader into a Flex SDK application (compiled through the command line mxmlc compiler):

1. either specify the library path directly to the compiler with the "library-path" option (*mxmlc -library-path+=path_to_swc ...*)
2. or add it to a compiler cofiguration XML file in the compiler section as:

```
<library-path append="true">
   <path-element>path_to_swc</path-element>
</library-path>
```

Please refer to the *howto.txt* file (located in the examples folder) when compiling the examples for specific / trageted building process information and requirements. Direct building of the Flex example can be accomplished by running the build scripts (.bat for Windows ; .sh for Unix – included in the example).

14 Public Methods and Properties.

After the working environment was set up and the library was embedded properly, the first lines that should be added to your project / application are classes imports.

Two main Data Matrix Reader classes must be imported alongside the default Flash class / package imports:

- *import ufo.DMReader;*
- *import ufo.DMRParameters;*

The *DMReader* class represents the main Data Matrix Reader object class and the *DMRParameters* class the parameter set definition class. When defining a call-back function, the *flash.geom.Point* class must also be imported.

Instantiation of the Reader object is done through a simple constructor. For e.g.

var DMScanner : DMReader = new DMReader();

After object creation, we can start configuring the Reader's appearance and behaviour:

- *width, height* – DMReader object properties referring to the Data Matrix reader canvas size (implemented through internal setter and getter methods). Cropping and scalling is done automatically by the reader component. For e.g.

DMScanner.width = 640;
DMScanner.height = 360;

- *SetParameter* – DMReader method used when setting / configuring the Reader. This method is also used when registering the call-back function or changing the Video source. It accepts static parameters defined in the DMRParameters class. These parameters are presented in the Parameter chapters.

DMReader.SetParameter(DMRParameters.PARMETER, parameter_value) : void;
For e.g.
DMScanner.SetParameter(DMRParameters.MARKER_COLOR , 0xCA0641);

- *Start* – Method that starts / resumes the decoder if it was put on hold (created "on hold") or if it was just created. (Must call the *Initialize()* method before)

  DMReader.Start() : void;

  For e.g.

  DMScanner.Start();

- *Stop* – Mehtod that puts the decoder on hold / pauses without destroying the Reader object. (Must call the *Initialize()* method before)

  DMReader.Stop() : void;

  For e.g.

  DMScanner.Stop();

- *Initialize* – Mehtod that initializes the Data Matrix Reader object : no internal reader structures are initialized / allocated before this method is called. It prepares the beginning of the decoding process. All parameters must be set with *SetParameter(...)* before this method is called.

  DMReader.Initialize() : void;

  For e.g.

  DMScanner.Initialize();

- *Registering the call-back function* : In order to take full advantage of the Data Matrix reading process, we must make use of the decoded string and optionally of the marker's quadrilateral coordinates. The Reader object communicates with the master application by triggering a registered call-back function. The call-back function must be structured in a fixed way:

function DMRCallBack(CodeText:String, P1:Point, P2:Point, P3:Point, P4:Point) : void;

where *CodeText* represents the decoded string and *P1*, *P2*, *P3*, *P4* represent the marker quad coordinates relative to the processed frame size.

Registering the callback function through the SetParameter method:
DMSCanner.SetParameter(DMRParameters.REGISTER_CALLBACK, DMRCallBack);

Please refer to the package examples for : setting a custom Video / camera stream , modifying the marker overlay appearance / behaviour and multiple parameter usage.

Writting a Simple Application.

```
package
{
   import flash.display.MovieClip;
   import flash.geom.Point;
   import ufo.DMReader;
   import ufo.DMRParameters;

   public class DMRExampleApplication extends MovieClip
   {
      private var DataMatrixReader : DMReader;

      public function DMRExampleApplication()
      {
         DataMatrixReader = new DMReader();
         this.addChild(DataMatrixReader);

         DataMatrixReader.SetParameter(DMRParameters.MARKER_COLOR,0xCA0641);
         DataMatrixReader.SetParameter(DMRParameters.REGISTER_CALLBACK,CodeFound);

         DataMatrixReader.Initialize(); // call before starting the decoder
         DataMatrixReader.Start();
      }

      private function CodeFound(Code:String,P1:Point,P2:Point,P3:Point,P4:Point):void
      {
         trace(Code); // do nothing, just trace the code
      }
   }
}
```

4. Parameter Tables

‹ Table 1 › **Full Parameter List**

	Parameter Equivalent	JavaScript Package	ActionScript Package
1	Camera Width	cw	CAMERA_WIDTH
2	Camera Height	ch	CAMERA_HEIGHT
3	Camera Frames Per Second	cfps	CAMERA_FPS
4	Marker Overlay Color	color	MARKER_COLOR
5	Displayed Video Stream Smoothing	smooth	SMOOTH_VIDEO
6	Show Detected Code Value	show	SHOW_CODE
7	View / Show Marker Interval	vmi	SHOW_MARKER_INTERVAL
8	Video Stream Capture Interval	capi	CAPTURE_INTERVAL
9	Callback-Function Interval	cbi	CALLBACK_INTERVAL
10	Detection Level	level	DETECTION_LEVEL
11	Detected Code Filter	filter	CODE_FILTER
12	Fixed Frame Rate	ffps	FIXED_FRAME_RATE
13	Threshold Box Blur Kernel Size	tbs	THRESHOLD_BOX_SIZE
14	Detection Mode	mode	DETECTION_MODE
15	External Video Stream Source	(NotAvailable)	SOURCE_VIDEO
16	Callback Function Registration	(NotAvailable)	REGISTER_CALLBACK
17	Hide Default Marker Overlay	(NotAvailable)	HIDE_MARKER

‹ Table 2 › **Default Parameter Values**

	Parameter	Value	Default	Comment
1	Camera Width	Integer	640	(pixels) depending on cam
2	Camera Height	Integer	360	(pixels) depending on cam
3	Camera Frames Per Second	Integer	30	(fps) depending on cam
4	Marker Overlay Color	Integer	13434624	(24bit RGB)
5	Displayed Video Smoothing	Boolean	false	(no smoothing)
6	Show Detected Code Value	Boolean	true	(overlay text visible)
7	View / Show Marker Interval	Integer	1000	(milliseconds)
8	Video Stream Capture Interval	Integer	20	(milliseconds)
9	Callback-Function Interval	Integer	2000	(milliseconds)
10	Detection Level	1-7,11,12,21,22	5	(square codes only)
11	Detected Code Filter	String	none	(no filter defined)
12	Fixed Frame Rate	Boolean	true	(fps locking)
13	Threshold Box Blur Kernel Size	Integer	0	(auto) (depending on frame resolution)
14	Detection Mode	0, 1, 2	0	(0 = black on white)
15	External Video Stream Source	Video	null	(default cam used)
16	Callback Function Registration	Function	null	(no callbacks)
17	Hide Default Marker Overlay	Boolean	false	(visible)

‹ Table 3 › Parameter Types

Basic Parameters	1 2 3 4 5 6
Advanced Parameters	7 8 9 10 11 12 13 14
ActionScript Package only	15 16 17

5. Basic Parameters

1 2 Camera Width and Height have a major impact in decoding accuracy (the bigger the resolution, the better the detection) but also affects the overall performance of the application, as a larger image has to be processed. By setting the camera resolution you set the actual resolution used in the detection process; the cam resolution can differ from the one displayed in the actual Reader.

3 Camera Frames Per Second is usually set to 30 fps, but this property is highly dependent on the actual hardware properties of the webcam (most webcams have support for 30fps at 640x480 resolution). If the camera doesn't have support for the set fps at the used resolution, the value for the fps will automatically be reduced.

4 Marker Overlay Color can be set as a 24 bit integer value (the default 13434624 is the equivalent of 0xCCFF00 in hex). The Marker refers to the color quadrilateral drawn over the found code.

5 Displayed Video Stream Smoothing parameter affects the quality of the displayed video stream and has no effect on the quality of the detection. Setting this parameter to true will also affect the overall performance of the application (it uses quite a lot of the CPU's capabilities). We recommend setting it to true only when a lower camera resolution is displayed at higher one (e. g. a camera resolution at

320x240 is displayed in browser at 640x480) or when the target machine disposes of enough hardware capabilities.

6 Show Detected Code Value over the Marker and on the transparent bar at the bottom of the Reader. If you do no wish for the user to see the detected code, set this parameter to false.

6. Advanced Parameters

7 View / Show Marker Interval refers to how long will the marker be shown after a successful detection, this offers a continuous visual experience (if one or two frames are dropped at a 25fps detection, the end user shall not see them, as the marker is drawn even during these frames).

8 Video Stream Capture Interval is an important parameter in the overall performance of the application (lower means more fps and smoother detection, but only on capable hardware) and is given in milliseconds (at 40ms the reader tries to detect at 25FPS = 1000 / 40). For machines with lower or average specs it is recommended to increase the value. When setting this parameter you should keep in mind the target user's hardware; usually, by decreasing the camera resolution you can process/detect at higher FPS.

9 Callback-Function Interval refers to the minimum interval (in milliseconds) between two consecutive Callback Function calls for the same detected code. This means that if the same code is found, let's say 60 times in 3 seconds (at 20 fps detection and a capture interval of 50), the Reader executes only 2 Callback Function calls (at a Callback Interval set 2000ms). For the JS version this is an useful parameter as it provides a simple way to limit the navigateToURL calls, which are slow.

10 Detection Level defines the method used by the Reader's detection algorithm to process the captured frame. It controls which type of codes can be detected (square only: 1-7; rectangular only: 11,12; mixed/both: 21,22), the Recognition Rate (of small, damaged, skewed markers under different angles) and False Positives Safety (1-3 levels are prone to false positives but can decode even badly damaged codes). If you only plan to use use square or rectangular codes, we recommend using the appropriate level to reduce method complexity / CPU load (or leave it default).

‹ Table 4 › **Detection Level**

Detection Level	Detected Codes	Recognition Rate / False Positives Safety	Method Notes
1	square*	Very High / Low	Brute Force
2	square	High / Medium Low	Single Sync
3	square	Medium High / Medium Low	Single Sync + MidPoints
4	square	Medium / Medium	Double Sync
5	square	Medium / Medium High	Double Sync + MidPoints
6	square	Decent / High	Double Sync
7	square	Decent / Very High	MidPoints
11	rectangular	Decent / High	Double Sync
12	rectangular	Decent / Very High	Double Sync + MidPoints
21	both	Decent / High	Double Sync / MidPoints
22	both	Decent / Very High	Double Sync + MidPoints

*only square codes of 1x1, 2x2 DataGrid size

11 Detected Code Filter defines a string_prefix for the decoded string. The Reader processes, shows and sends only the codes starting with the string_prefix. This is a good feature for the reduction of False Positives Probability. You can also use this

feature in order to select only codes matching your specification (e.g. : codes that start with "codes" can be set and displayed in the JS version with: *"../reader/dmreader.swf?filter=codes"* or in the AS version by calling the method: *SetParameter (DMRParameters.CODE_FILTER, "codes");*).

⑫ Fixed Frame Rate adds a frame processing rate limitation if set to "true": The frame processing rate doesn't exceed the current Camera FPS. Useful in order to keep the CPU usage bounded to the maximum available Camera stream frame rate. In some cases it sacrifices displayed stream "smoothness" for lower CPU resources. If set to "false", all available stream frames are processed.

⑬ Threshold Box Blur Kernel Size controls frame / image pre-processing (the threshold / binarization process). It can be set with a integer value (e.g. 40). The default value is *0* : Automatic Box Blur Kernel Size depending on camera resolution. Manual setting is recommended only for controlled usage environment.

⑭ Detection Mode refers to what type of barcodes can be detected (normal: black on white background or inverted). The default value is 0 = normal black or dark-colored codes on a white background. If you only plan to use normal or inverted codes, we recommend setting only the appropriate mode (or leave it default).

‹ Table 5 › Detection Mode

Detection Mode	Detected codes
0	black on white (normal)
1	white on black (inverted)
2	both (normal + inverted)

7. ActionScript-Only Parameters

15 External Video Stream Source allows you to change the default video source (from the default camera) used by the Reader. E.g.
DMDecoder.SetParameter(DMRParameters.SOURCE_VIDEO, AlternativeVideo);

Once you set a custom input stream, you cannot alter the FIXED_FRAME_RATE parameter. In this case it will be automatically defaulted to "false".

16 Callback Function Registration sets the Callback Function which is triggered / called when a valid code is detected by the Data Matrix Reader component. E.g.
DMDecoder.SetParameter (DMRParameters.REGISTER_CALLBACK, CodeFound);

The Callback function must be defined with a specific set of parameters: Decoded_Text (String) and 4 (four) points, which define the four corners of the detected Marker quadrilateral. E.g..
function CodeFound(Decoded_Text:String,P1:Point,P2:Point,P3:Point,P4:Point):void

17 Hide Default Marker Overlay can be set to *true* to hide the default Marker quadrilateral which is drawn over the detected code. E.g.
DMDecoder.SetParameter (DMRParameters.HIDE_MARKER, true);

This parameter can be used in conjunction with the Callback function to display a custom marker (using the 4 points as reference).

8. Notes

- The actual intervals may vary due to machine, browser and run – time environment issues.

- By adjusting the parameters you can get a smooth user experience even on slower machines

- Decoding accuracy is highly dependent: on image or video quality under specific lighting conditions, on marker printing quality, positioning and on camera capabilities.

- If you encounter any issues in using our software, feel free to contact us.

9. Package Contents

The following sections outline the basic folder structure and contents of the corresponding delivered packages.

JavaScript Package

▸ documentation	• Data Matrix Reader instructions manual
▸ example	• a simple web page which loads a few examples of the reader with different parameters
▸ lib	• Data Matrix Reader SWF component

ActionScript Package

▸ documentation	• Data Matrix Reader instructions manual
▾ examples	• examples folder
▾ for_flash	• examples for Adobe Flash Professional CS4+
▸ 1 – basic	• importing the library and setting a callback function
▸ 2 – customInterface	• creating a custom interface and setting parameters
▸ 3 – customVideo	• using a custom video and creating a custom marker
▸ for_flex_sdk	• simple example for Adobe Flex SDK
▸ lib	• Data Matrix Reader SWC library

10. Rights and Warranties

This is an agreement between you (either as an individual or as a commercial entity) and Ufo Media Design SRL. Please read the following terms and conditions: using this Software is conditioned by reading and accepting of all aforementioned terms. For the following, Provider shall mean "Ufo Media Design", Software shall refer to the "TagsRepublic Data Matrix Web Package Reader" and Licensee shall represent the Person or Entity which acquired the right to use this Software. Reader shall refer to parts of the Software delivered as Complied Code.

I The Provider grants the Licensee a non-exclusive license for using and embedding the Software according to the following terms.

II The Software is protected by international legislation regarding intellectual property. The rights granted to you constitute an embedding license and not a transfer of title (the Software is not sold or otherwise transferred).

III The Provider warrants to be the creator of the Software, and to be the sole owner of all intellectual and/or industrial property according to the Software. The Provider guarantees the Software does not infringe any rights of third parties.

IV The Licensee may not assign / attribute any rights hereunder without the prior written approval of the Provider. Any attempt to assign any rights, duties, or obligations hereunder without the Provider's written consent will be void.

V In no event will the Provider be liable to the Licensee or any other individual or entity connected with the Licensee for any claim, loss, or damage of any kind or nature, arising out of or in connection with the performance of this Software (or other related components). Any interruption or loss of service or use of the Software, or any files, data or other computer systems shall in no way cause liability to the Provider.

VI The Licensee is prohibited, except as expressly authorized under the present terms, from:

a) using the Software on behalf of third parties, except when specified otherwise;

b) selling, renting, leasing, lending or granting other rights (on) Software including rights on a membership or subscription basis.

VII The Licensee shall not reverse engineer, decompile, disassemble or otherwise attempt to discover the Source Code of the Software or it's components delivered as Compiled Code.

VIII The Licensee may use / embedded the Software in only 1 (one) commercial or public application (e.g. site, web app) instance at a time. By completely removing the Software from a application in which it was used / embedded, the Licensee has the right to reuse the Software in another application with the mentioning that the Software shall be used / embedded in only one application at a time. In this case, the Licensee shall guarantee the complete removal of all files which are the intellectual property of the Provider from any program or application in which the Software was used / embedded before.

IX The Licensee has the right to sell 1 (one) application which uses / embeds the Software to a third party. In this case, the Licensee shall take all appropriate measures to guarantee that the aforementioned third party shall comply with the present terms; also, by selling an application which uses the Software, the Licensee shall forfeit the right to use / embed the Software in other applications. The Provider grants the usage right to such a third party only in an application developed by the Licensee and which become the property of such third party, but doesn't grant (to aforementioned third party) any redistribution, adaptation, copying, embedding rights or the right to use in demos or tests. Any other use requires the written consent of the Provider.

X The Licensee can modify or alter portions of the Software, except the components and files supplied as Compiled Code. The Licensee is prohibited to distribute or make public any part of the Software delivered as Source Code; the Source Code can only be used in order to build a compiled, customized version of the Software.

XI The Reader and all of its copies are the intellectual property of the Provider. The structure, organization and code of the Reader are valuable and confidential information of the Provider. These terms do not grant the Licensee any intellectual property rights on the Reader and all rights not expressly granted are reserved by the Provider.

XII The Provider guarantees that the Software components delivered as Compiled Code do not contain any malware, malicious code, or any other types of virus, trojan, spying, unwanted software that violates the security, confidentiality and integrity of any target system. Any security holes that could be generated in conjunction with the use of the Software or its target system shall not make the Provider liable for any resulted loss or damages.

XIII Any malfunction or inconsistency of the Software caused by changes or modifications (new versions, upgrades, updates, etc.) of other third party supporting platforms (e.g. “Adobe Flash Player Run-time”), will not derive any responsibility to the Provider.

XIV The Licensee undertakes not to embed the Reader in sole proprietary service applications that could provide to other third party applications, the same functionality as the Reader.

XV The Licensee may use the Software for as long as the Licensee elects to do so. The Provider does not enforce any limit to the license period of the Software, except when the Licensee is in breach of the present terms.

XVI The Licensee has the right to use / embed the Reader in as many noncommercial, nonpublic applications (property of the Licensee) as he / it wishes (e.g. test, demos). The Licensee undertakes not to provide access to such applications (it is forbidden to upload the Software on publicly accessible websites). The publication of such applications requires the written consent of the Provider.

XVII Should any provision of this Agreement become invalid or unenforceable or should the License contain an omission, the remaining provisions shall be valid and enforceable and not affect the validity of the remainder.

XVIII Any dispute, controversy or claim arising out or in relation to this licensing terms and conditions shall be settled through negotiations between Parties. If the Parties fail to settle the dispute amicably, any outstanding dispute or difference between them may be referred or submitted to the Provider's territory courts and in accordance with the jurisdiction of the Provider's local law.

Tags Republic
Ufo Media Design SRL
contact@tagsrepublic.com
www.tagsrepublic.com