



# TITLEBOX USER MANUAL

*(Version: 2.3, March 2020)*

Current software versions:

**TitleBox Mega:** 12184.0

This guide explains in detail all functionalities of the **TitleBox** modules.

# Legal notice

The information in this manual is furnished for informational use only. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the prior written permission of PlayBox Technology UK Ltd.

The software, described in this manual, is owned by PlayBox Technology UK Ltd. It is protected by England and Wales Copyright Law, as well as by international copyright treaties, and may be used or copied only in accordance with the license agreement.

PlayBox Technology UK Ltd. provides this manual "as is" without any warranty, either express, or implied.

This publication may contain typographical errors or technical inaccuracies. While every precaution has been taken in the preparation of this document PlayBox Technology UK Ltd. assumes no responsibility for errors or omissions. Nor is any liability assumed for damages, resulting from the use of the information, contained herein. Changes are periodically made to the information herein. They will be incorporated in new versions of the manual. Please, check the **PlayBox website** regularly for User Manual updates.

PlayBox Technology UK Ltd. may introduce changes or improvements in the products, described in this manual at any time, without any special notice.

Please, address your comments or questions to:

PlayBox Technology UK Ltd  
Brookmans Park Teleport  
Great North Road  
AL96NE Hatfield  
United Kingdom

[info@playboxtechnology.com](mailto:info@playboxtechnology.com)

[www.playboxtechnology.com](http://www.playboxtechnology.com)



## Contents

<b>TITLEBOX USER MANUAL</b> .....	<b>1</b>
<b>LEGAL NOTICE</b> .....	<b>2</b>
<b>CONTENTS</b> .....	<b>3</b>
<b>PREFACE</b> .....	<b>5</b>
<b>STYLE CONVENTIONS</b> .....	<b>6</b>
<b>INTRODUCTION</b> .....	<b>7</b>
GETTING STARTED .....	7
<i>Quick Start</i> .....	7
<i>Manage TitleBox Instances</i> .....	7
USER INTERFACE.....	8
.....	8
<i>Work Area</i> .....	8
<i>Toolbars</i> .....	9
<i>Object Palette</i> .....	13
<i>System Bar</i> .....	14
<i>Status Bar</i> .....	14
MENU BAR.....	15
<i>File Menu</i> .....	15
<i>Edit Menu</i> .....	15
<i>Network Menu</i> .....	15
<i>Object Menu</i> .....	16
<i>Project Menu</i> .....	17
<i>Help Menu</i> .....	26
CREATING OBJECTS .....	26
<i>Step By Step</i> .....	26
<i>Editing Objects</i> .....	26
<i>Deleting Objects</i> .....	27
<i>Object List</i> .....	27
<i>Object Properties</i> .....	27
SCHEDULER.....	51
<i>Scheduler Commands</i> .....	51
<i>Object Settings</i> .....	51
<i>Objects List</i> .....	52
SLIDE MANAGER.....	53
<i>Slide control buttons</i> .....	53
<i>Project control buttons</i> .....	53
<i>Project Preview area</i> .....	53
<i>Slide Controller</i> .....	59
DATA SOURCE MANAGER.....	60
<i>Weather Data Provider</i> .....	63
<i>FileLink Data Provider</i> .....	63
<i>ODBC Data Provider</i> .....	65
<i>XML Data Provider</i> .....	66
<i>RSS Data Providers</i> .....	67
<i>HTML Table Data Provider</i> .....	68
<i>EAS (Emergency Alert System) Data Provider</i> .....	69



TASK MANAGER.....	70
<i>Tasks</i> .....	70
<i>Input Events</i> .....	86
<i>Assigning a task to an event</i> .....	89
AS-RUN LOG.....	90
<i>User Interface</i> .....	90
<i>Menu Bar</i> .....	91
<b>APPENDIX 1 - PLAYBOX GPI.....</b>	<b>93</b>
GPI IN AIRBOX.....	93
GPI PINOUT.....	93
GPI-IN IMPLEMENTATION.....	93
GPI-OUT IMPLEMENTATION.....	93
AIRBOX AS A GPI SLAVE.....	93
AIRBOX AS A GPI MASTER.....	94
AIRBOX GPI SETTINGS PANEL.....	94
RS232 9-PIN D-SUB PINOUT REFERENCE.....	94
GPI INPUT REFERENCE.....	94
GPI OUTPUT REFERENCE.....	95
<b>APPENDIX 2 – GRAPHIC RULES’ COMMANDS, USED FOR COMMUNICATION BETWEEN AIRBOX AND TITLEBOX.....</b>	<b>96</b>
<b>APPENDIX 3 – INTEGRATION OF AIRBOX WITH TITLEBOX.....</b>	<b>98</b>
<b>APPENDIX 4 – PLAYBOX GPI BOARD AND BYPASS RELAY BOARD.....</b>	<b>99</b>
<b>APPENDIX 5 – TITLEBOX SPECIFIC CLASS PROPERTIES, FUNCTIONS, AND PROCEDURES EXPLAINED.....</b>	<b>101</b>
TTBOBJECT (INHERITOR OF TOBJECT).....	101
TTBSLIDE.....	108
TTBSLIDESMANAGER.....	109
TTBDATAPROVIDER.....	110
TTBDATADISTRIBUTOR.....	112
TTBPROJECT.....	113
TITLEBOX FUNCTIONS.....	114
TBITMAP32 (INHERITOR OF TOBJECT).....	115
TTBSTORAGE.....	116
<b>APPENDIX 6 – TITLEBOX PROGRAM SCRIPT EXAMPLES.....</b>	<b>117</b>
<b>APPENDIX 7 – WINDOWS CONFIGURATIONS FOR USING NON-ENGLISH OBJECT NAMES IN AIRBOX – TITLEBOX.....</b>	<b>148</b>
<b>NET CONTROL MODE.....</b>	<b>148</b>
<b>APPENDIX 8 – TITLEBOX KEYBOARD SHORTCUTS.....</b>	<b>150</b>
<i>PROPERTY TOOL</i> .....	151
<b>INDEX.....</b>	<b>152</b>



## Preface

Dear **PlayBox** customer,

Thank you for purchasing our product! We would like to assure you that you have chosen the most cost-effective and versatile TV automation system on the market. As always, we are trying to stay close to our customers' needs, making sure they all receive adequate support and satisfaction. Your opinion about our product is an exceptionally valuable source of information for us. The ease of working with the **PlayBox** products results mainly from the suggestions and comments of our current respected customers.

This manual is structured into several sequential chapters, each aiming to ease the installation, fine tuning, and use of our products. We hope that you will enjoy working with it, and we are anxiously looking forward to receiving your feedback.

Please, send your questions, suggestions, and assistance requests to:

[support@playboxtechnology.com](mailto:support@playboxtechnology.com)



## Style Conventions

- File names, software, documents or terms are written in *italics*
  - The data is written in the *settings.ini* file.
  - The file is located in *C:\Program Files\DMT\AirBox*
  - For further information read *Shortcuts* reference book.
  - The VTR is controlled via *RS-422*.
- Direct quotations from the computer screen are presented as follows:
  - **Menu Items and commands**
  - **Tab/Page names**
  - **Column names (i.e. in a playlist or another grid)**
  - *Field names, check boxes*
  - **Buttons**
  - Screen readings are written in [square brackets]
  - **The keyboard keys are enclosed in <> signs**
  - *Terms are defined in the Glossary at the end of the manual*
- The arrows, used in the setting procedures mean as follows:
  - → A menu item follows;
  - ⇒ A page(tab) name follows;
  - → A field name, a check box name, or a value name follows.

Except for arrows, you can distinguish between the relevant menu categories also by the styles, listed above.



## Introduction

### GETTING STARTED

The **TitleBox (TB)** module is an on-air graphics manager. You can create different static or dynamic objects in **TitleBox**, such as rolls, crawls, still pictures, clocks, etc., and save them in projects. The projects could be used directly for broadcasting or as customized templates.

In **TitleBox** you can also start objects from previously created project(s) at different times, thanks to its **Scheduler**.

**TitleBox** works together with the **AirBox** playlist. Separate objects in **TitleBox** can be started or stopped via **AirBox**, by inserting *TitleBox Net control events* in an **AirBox** playlist.

For more information about an **AirBoxTBNetCtrl** event, please see [AirBox Settings->Modules->RemoteControl->TitleBox NetControl](#) chapter.

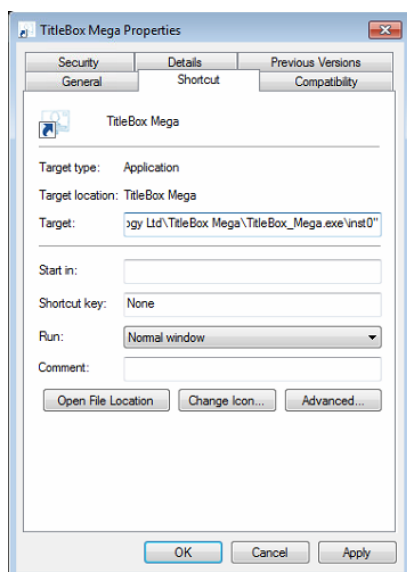
### Quick Start

1. Launch **TitleBox**;
2. Click the **Crawl** button in the object palette in the left;
3. Draw a rectangle in the **Preview Area**;
4. Type a text in the **Properties** dialog box and press **OK**;
5. Press the two **Play** buttons – one in the bottom of the window and one in the third row of the taskbar.
6. Congratulations! You have just created your first **TitleBox** crawl!

### Manage TitleBox Instances

You can use more than one **TitleBox** instance if you are licensed to. The number of **TitleBox** instances you can use depends on the number of licenses you have. You can create shortcuts for the different instance numbers. This is done in the following way:

Assume that you are allowed to use three different **TitleBox** instances – **TB instance #1**, **TB instance #2**, and **TB instance #3** and that you want to add a desktop shortcut for each separate instance. You have to do the following:



1. Create three **TitleBox** shortcuts on your desktop.

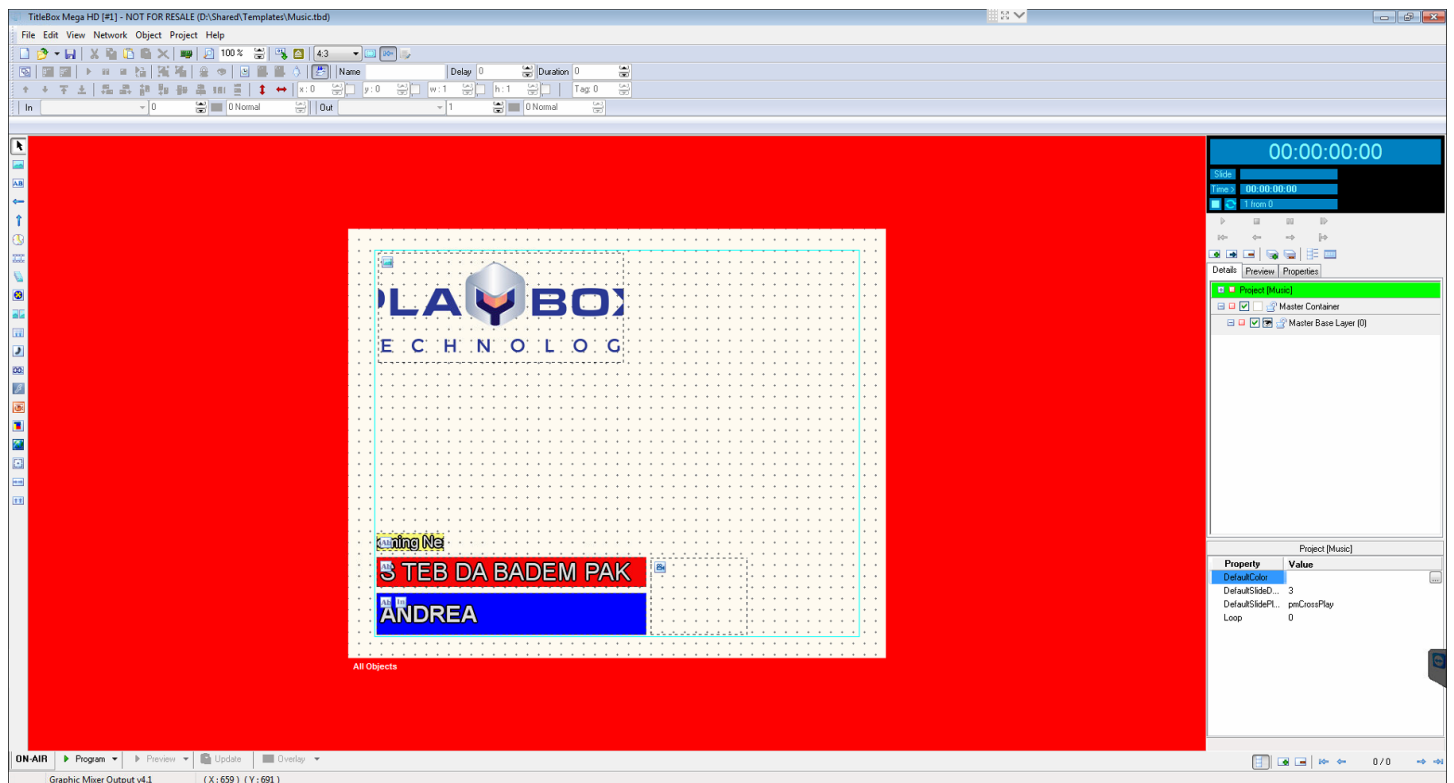
# TitleBox User Manual



2. Right-click on the first shortcut and in the *Target* field of the shortcut add “\inst0” at the end of the line, as shown in the image to the right. This will associate this **TitleBox** shortcut to **TB Instance#1**.
3. Click OK and repeat the above step for the other two shortcuts. Enter “\inst1” and “\inst2” respectively for **TB Instance #2** and **TB Instance #3**.


For **TB Instance #[n]**, you need to enter “\inst[n-1]” in the *Target* field of the Shortcut Properties.

## USER INTERFACE



## Work Area

**TitleBox** interface is very user friendly and it can be used easily in a live environment. The main part of the user interface is the *Work area*. It shows the objects, their positions, types, and status.

Objects can be moved and resized in this area. If an object is too complex and is not rendered yet, a yellow/black sign, saying **[Rendering]** will appear on the screen. This sign  will automatically disappear when the rendering is complete.

The little icons in the upper-left corners of the objects show their type, lock status, and transitions. You can enable/disable the icons from [Project menu](#) → [Options](#) → [General](#) →  [Show Objects Icons](#).

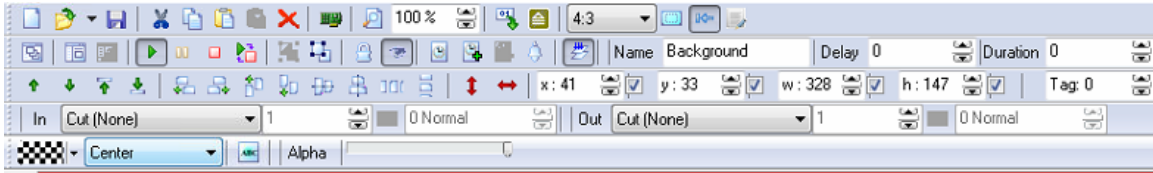
The color bar at the bottom of the work area displays the current mode of operation. By default, the preview mode is outlined in blue. You can change this color in the [Project menu](#) → [Options](#) → [General](#) tab.





## Toolbars

The **Toolbar** is designed to facilitate the project management, the individual object control, as well as the object ordering and alignment:



The **New**, **Open**, and **Save** buttons correspond to the relevant commands in the **File Menu**. The **Cut**, **Copy**, **Paste**, and **Delete** buttons correspond to the relevant commands in the **Edit Menu**.

**Paste as Copy/ In Slide** will create a new instance of the object that is copied in the clipboard. Unlike the simple **Paste** that creates a new object, **Paste as Copy** does not create objects. All changes applied to the originally copied object will be applied to all its instances.

The **Select Display device** button opens the list of available hardware drivers, from which you can select the one to work with. See also the corresponding section in the [Project Menu](#) description.

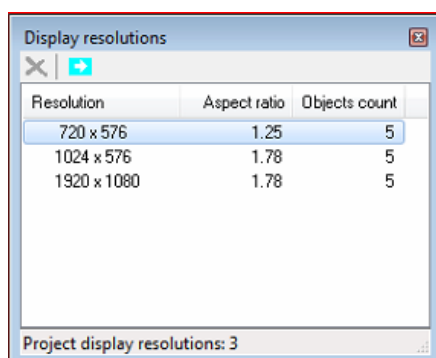
**TIP (!)** If you want to change the driver before its initialization, hold down **<Ctrl + Shift>** during **TitleBox** startup to open the **Driver select** dialog.

The **Zoom** spin-box allows resizing the work area, so you could see the whole project at a glance. It is very useful for viewing HD projects.

This button opens the **Data Source Manager** window. It contains all supported data providing plug-ins. You can use them to “translate” different types of data and display the information they contain in some of the **TitleBox** objects. Please, check the Data Source Manager [section further below](#).

This button opens the **Task Manager** window. Please, check the Task Manager description in the related [section further below](#).

When the **Output aspect ratio** is changed, **TitleBox** will automatically change the size of the objects according to the selected aspect. All of the aspect changes concerning one particular project are listed as presets in the **Display resolutions** dialog. To open it, press the **display resolution** button. A dialog will appear. It provides possibilities for fast switching between the different presets. Pressing the button or double-clicking one of the rows will initiate an aspect change. To delete a preset, use the button.




The **External aspect control** button is pressed by default. It is intended for automated switching of the aspect ratio according to an external source. Such a source can be WSS present in the incoming video, for example.

The **As-run log ON/OFF** button allows you to enable/disable the **TitleBox As-Run Log** application. The **As-Run Log** is configured in [Project menu](#) → [Options](#) ⇒ [As-Run Log](#).


The **Objects list** button shows a window with a list of all objects, available in the current **TitleBox** project.





The **Properties buttons**  are object-related. They allow for viewing or changing the properties of the currently selected object.


There are two types of properties: **Standard Properties**, that provide the standard object options, and **Property Tools**, which provide an additional set of properties. The **Property Tools** are active only for texts, rolls, and crawls.


**IMPORTANT:** The properties can be interactively changed at any moment, even when the object is running on-air.

The **Play, Pause, Stop**, and **Toggle Play/Stop** buttons  control the play status of any particular object. They correspond to the relevant commands in the **Object Menu**.

The **Group** and **Ungroup** buttons  are intended for grouping and ungrouping a selection of objects.


The **Lock** and **Visible** buttons  define the object status. It could be locked for moving and resizing or not. It could also be visible in the preview area or not.


The **Schedule** buttons  concern project scheduling. The scheduling function allows you to start each object in a project at a specific time/day of week, with specific parameters. You can read more about scheduling in the [Scheduler](#) section.

The **Schedule window** button  opens the scheduler.

The **Add to Schedule** button  adds the selected object to the schedule.


The **Remove from Schedule** button  removes the selected object from the schedule list.


The **Show Events List** button  will open a list of all events, assigned to the currently selected object. See also the [Task Manager description](#).


The **Mix Draw Mode** button  defines the draw mode of the object. If the button is pushed, two or more overlapping objects will be blended.


**(!) TIP:** When in **Mix object** mode, you can switch between the overlapping objects using the **<Alt> + click** combination. The name of the currently selected object will appear in the *Name* field and will be highlighted in the [Object list](#) window. You can activate this window from within the **Object** menu.


The *Name* field contains the name of the object. The default names are [Type]\_[#], i.e. [Crawl 1], [Roll 3], etc. You can change them at your will. The names are most important when you use the **Objects list** window, where only names and properties are displayed. In order to change the object name, just select the object, and then enter a new name into the *Name* field.

In the *Delay* field you can specify a delay for each individual object. A three-second delay means that the object would start three seconds after you have pressed the **Show object**  button.

In the *Duration* field you can specify the duration for an object, if needed. The duration determines how long the object will be displayed after pressing the **Show object**  button.

The **Order** buttons  allow defining the objects' order in case they overlap. Overlapping dynamic objects is not desirable.

The **Alignment** buttons  - allow for aligning objects to each other. These buttons work when more than one object is selected.


The **Lock buttons**  are used for locking the horizontal or vertical (or both) sizes of the object.

The **X** and **Y** numeric fields stand for X and Y positions of the selected object in the 2D space.

The **W** and **H** numeric fields stand for Width and Height of the selected object.



**NOTE:** You cannot change some objects' size during their running on air.

The *Tag*  spin-box is used for additional object control. By default, an object's tag is set to [0]. However, you can enter another value here. The idea of the tag is that you can simultaneously operate with many objects that share the same tag number. This could be done via the [script task](#) and the [program script task](#).

The *3D Offset* spin-box allows you to set an offset to the selected object in terms of depth. If you set a positive number, the object will appear deeper in the screen and vice versa.

The transitions toolbar will help you in setting your objects' *In* and *Out* transition effects:




First, select the transition effect from the *In* and *Out* drop-down list. The available types are [Cut(None)], [Cross Fade], [Fly] and [Wipe]. They are not active for **Analog clock** objects.

In the spin-boxes next to the *In/Out* fields set the duration of this transition (by default it is 1.00 second).

**NOTE:** Increasing the figures in these fields will decrease the speed of the transition.

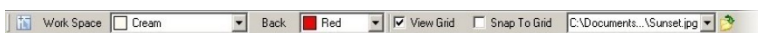
Finally, define the type of motion during the transition (ascending, descending, or constant). By default, the motion type is set to [0 Normal]. This means that the transition will proceed at constant speed.


For example, if you have set a [Fly From Left] effect and duration of 2 seconds, your object will move at a constant speed from left to right till it reaches the desired position within two seconds. If you set the motion type to [(1 to 5) Ascending], the object will start in a slower motion and its speed will accelerate during the 2 seconds of transition. In descending mode, the speed of the transition will decelerate.

The **Full screen transition** buttons  are situated next to the transition duration spin-boxes. You might need this functionality in case there are several grouped objects on the screen and these objects have the same transition effect and duration. If you trigger or stop such objects simultaneously, they will move in relation to each-other, because they have to cover different distances on the screen for the same period. Using this button will make objects move together, but not in relation to each-other, during the transitions. The transition of all grouped objects will look as if the entire screen is moving, not the separate objects.

The lowest toolbar contains object-specific settings that change depending on the currently selected object.

If no object is selected, this toolbar allows for changing the work area and the grid.



Push the **Token** button  if you need to be more precise while drawing the objects in the work area.


You can change the colour of the drawing canvas from the *Work Space* drop-down palette. To change the colour of the surrounding area, use the *Back* palette.

The  *View grid* box is checked by default. If you do not want to view the grid dots, uncheck it.

**TIP (!)** You can adjust the distance between the dots in the [Project menu](#) → [Options](#) ⇒ [General](#) tab.

If the  *Snap to Grid* box is checked, all objects in the project will be aligned to the nearest grid point.



If you want to load a picture as a background in the work area, you can browse for it after pressing the  button.

**NOTE:** Currently, it is not possible to unload the background file, unless you close **TitleBox**. You can just change it.

**TIP (!)** You can take a snapshot of the current output (fill and key) from **TitleBox** by pressing **<Ctrl+F12>** keys on your keyboard. Then, you can paste the shot into a picture editor and save it to a file.



## Object Palette

The **Object palette** contains buttons for all supported graphics objects. Click on the object you need and draw a rectangle in the work area to create it. Press this button to select object(s) by dragging a rectangle around them.



**Create a Still picture object.**

**Create a Text Template object** (with background).

**Create a Roll object** (vertically running text).

**Create a Crawl object** (horizontally running text).

**Create an Analogue Clock object** (with custom background and clock hands).

**Create an Animation object** (a sequence of 32-bit .tga files, or animated .gif files).

**Create an Animation File.**

**Insert Direct Show Media Source.**

**Insert a Banner object.**

**Create a Chat note object.**

**Create a sound object** (any DirectShow-supported sound formats – .mp3, .wav).

**Create a Digital Clock object** (with custom background and font).

**Insert a flash object.**

**Insert a Power Point presentation.**

**Create a Primary shape** – squares, ovals, triangles with outline offsets.

**Create a browser object.**

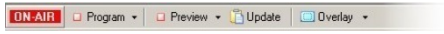
**Create a Screen capture object.**

**Create a chat line**

**Create a chat roll**



## System Bar



The system bar is situated at the bottom of the interface window. It contains buttons for hardware control:

**[ON AIR]** – playout notification filed. It blinks red when **TitleBox** is on air (i.e., the Program driver is playing).

**Program** – this is the button that controls the program output driver. Pressing it will start/stop the graphics frame buffer. If it is not running, no graphics will appear in the output. Pressing the arrow to the right of it opens a drop-down list that contains several items:

- Stop/Play– stop or play the program output driver.

- Clear will erase the graphics frame buffer, so there will be no remnants from any previously loaded projects.

- Program Driver SetUp – opens the settings options of the currently selected Program driver.

- Driver manager opens the list of available drivers, so you could choose the one to use for outputting the graphics or a driver to preview it.

**Preview** – this is the button for controlling the selected preview driver. While the preview driver is playing, the bar under the work area will turn blue. A text message in it will notify you that the Preview driver is running.

The preview mode allows you to make changes in your project, and view them in a preview window without actually applying them on the output.

When the preview driver is stopped, all changes you make in the project will be applied in real time to the output.

Pressing the arrow to the right of the **Preview** button opens a drop-down menu. It contains the same items as the ones of the program driver.

**NOTE:** In preview mode you cannot create a Flash object! You can just edit already existing objects!

**NOTE:** Some objects might not appear properly in the simple preview window!

**Update** - While in preview mode, press this button to apply the changes to the program output.

**Overlay** – It is not in use for the current **TitleBox** version.

## Status Bar

The status bar displays information about the currently selected driver and the X/Y position of the pointer in the work area:

\\\\192.168.60.218\\Soft\\mp2\\CaptureBox\\CB\_help.chm



## MENU BAR

### File Menu

This menu contains commands related to the project's file:

- New** – allows creating a new project.
- Open** – opens an existing project (\*.tbd).
- Reopen** – shows a list of the ten recently open projects.
- Merge** – merges the current project with another one.

**NOTE:** Global layer links might not operate properly because the Master container of the second project will appear as a simple slide in the merged project. You will have to drag its layers to the merged Master Container and set the layer links again.

- Save** – saves the current project to the open file (\*.tbd).
- Save as** – saves the current project to a specified file (\*.tbd).

### Edit Menu

The Edit menu contains object-related commands:

- Undo** – use it to undo **Move**, **Size**, and **Create**. Please, note that there is no undo for deleting objects.
- Cut** – removes the selected object(s) and keeps them in the buffer-memory.
- Copy** – saves a copy of the selected object in the buffer-memory.
- Paste** – pastes the buffer content in the project.
- Delete** – deletes the selected object.
- Select All** – selects all objects in the *Preview* area.

### Network Menu

This menu concerns the connection between **AirBox** and **TitleBox**.

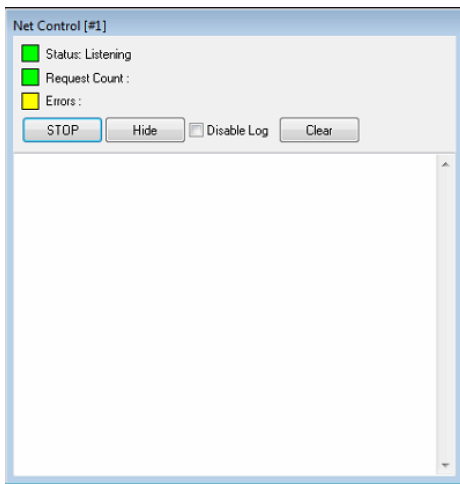
**NOTE:** This feature is available for the **TitleBox** full version only. It is not available in the **TitleBox** light version.

For detailed step-by-step guide on how to integrate **TitleBox** with **AirBox** look up in [APPENDIX 3 – Integration of AirBox with TitleBox](#)

**Export Project as Template** – exports the currently open project as a template. It will be saved in the specified Template folder. You have to specify this folder during **TitleBox** Net Control activation (see *Appendix 3 for details*).

**IMPORTANT:** The **project must be exported as a template**, so that it could be controlled through **AirBox**.

**Net control** – shows the connection status between **AirBox** and **TitleBox**.



**IMPORTANT:** In order to be able to control **TitleBox** through **Net Control** output events from **AirBox**, you have to switch **TitleBox** in **Net Control** mode (**TitleBox Network** menu → **Net Control**).

If you want to control a **TitleBox** project (and its separate objects) from **AirBox**, you have to export it as a template (**TitleBox Network** menu → **Export project as template**). The template folder will open automatically, so you do not have to browse for it. Just type a name and press **Save**.

## Object Menu

This menu contains object-related commands:

**Play**- shows and runs the selected object on the graphics frame buffer.

**Pause**- freezes the object but it remains displayed on the graphics frame buffer.

**Stop** – hides the object from the graphics frame buffer.

**Add to Scheduler** – Adds the selected object to the Schedule.


**Remove From Scheduler** – Removes the selected object from the Schedule.

**Scheduler Properties** – Shows the properties of the schedule

**Order** – This function is active when there are overlapped objects. You can move the selected object under or over the others.

**Alignments** – This function is active when more than one object is selected. It allows for aligning the objects to each other.

**Assign Task** - this function invokes the **Task Manager** for the selected object. More detailed information you can read in section [Task Manager](#), further in that manual.

**View Events** – select it to view all tasks, assigned for the currently selected object. This menu item corresponds to the **View Events List** button in the toolbar .

**Object list** – shows a window with a list of all objects available in the current **TitleBox** project (see its description [further in this chapter](#)).

**Property** – opens the **Properties** window of the currently selected object.

**Property Tools** – opens additional properties windows for objects (text, rolls, and crawls).

The commands from this menu (except for Assign Task) are also displayed in the second uppermost row of the toolbar.





## Project Menu

This menu contains project-related commands. Some of them (**Play**, **Pause**, etc.) are also operated through buttons, situated along the bottom of the window.

### Project playout commands

**Play**– starts the driver of the graphics frame buffer. If this button or command is not triggered, you will not be able to show any graphics on the screen.

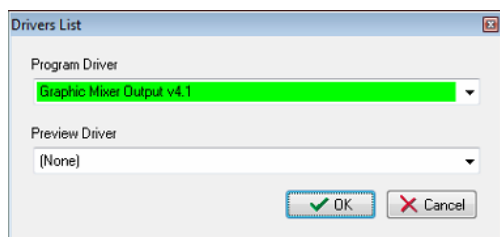
**Stop** – hides all objects from the screen but they remain loaded in the graphics frame buffer.

**Clear Buffer**– clears the graphics frame buffer. This command helps avoiding accidental showing of remnants of an old project when loading a new one.

**Scheduler** – Opens the scheduler window for setting up the scheduled graphics insertion.

**Mix Objects** – This command is a duplicate of the **Mix Draw Mode** button. It will blend the overlapping objects in the project.

### Driver Select



Click on it to view the list of available hardware drivers and select the one to work with. If there are no hardware devices installed, the list will contain the following lines:

[Preview Output Driver]- It is used for previewing the **TitleBox** project in a specially designed software preview window.

[Mapped Memory Driver] – This driver is used, when **AirBox** and **TitleBox** are being used in the same playout system. It allows keying the graphics over the video played in **AirBox**.

[TB DirectShow Driver] – This driver is used when there is a **Blackmagic DeckLink** card installed on the machine.

You have to choose the *Program Driver* and the *Preview Driver* separately.

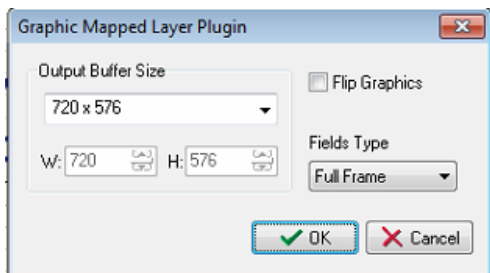
Please, check the **PlayBox** web site (<https://www.playboxtechnology.com>) or contact **PlayBox** support team ([support@playboxtechnology.com](mailto:support@playboxtechnology.com)) for currently supported boards.

### Driver Setup

Press it to change the output settings. The available options depend on the currently used driver.



## Mapped Memory Driver settings



*Output Buffer Size* – select the output resolution from the drop-down list. If you select a Custom resolution, then the next *Width* and *Height* fields will become available for entering your resolution values.

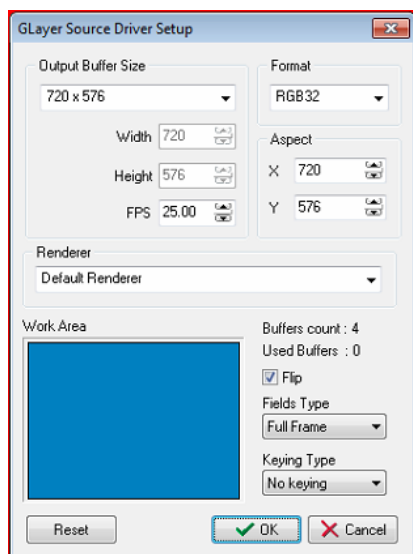
*Flip* – if you check this box, the output graphics will be flipped at 90 degrees.

*Fields type* – select the field order from the drop-down list.

Press the **OK** button to save the changes.

## TB DirectShow Driver settings

If **TitleBox** works as a standalone application, you have to select [TB DirectShow] driver. The following windows will appear:



*Output Buffer Size* – this is the output **TitleBox** resolution.

If you select a *Custom* resolution in *Output Buffer Size*, then the next **Width** and **Height** fields became available for entering your resolution values.

In *Aspect X/Y* fields enter the aspect for the display resolution.

*FPS* stands for *Frames per second* setting.

*Format* is a video formats - RGB, UYVY, etc.

*Renderer* – this drop-down list shows all available direct show renderers on this machine. Select the **DeckLink video renderer** here.

*Work Area* – it is not in use in the current version.

*Flip* – check the check-box, in order to flip the output graphics at 90 degrees.



*Fields Type* – select the field order here from the drop-down list.

*Keying Type*—here you can select the type of the keying (or no keying). Some support either "Internal Key" or "cc" only. Other DeckLink cards support both keying types.

**Internal keying** means that the video input, keying, and resulting video output, are all performed internally within the one DeckLink workstation.

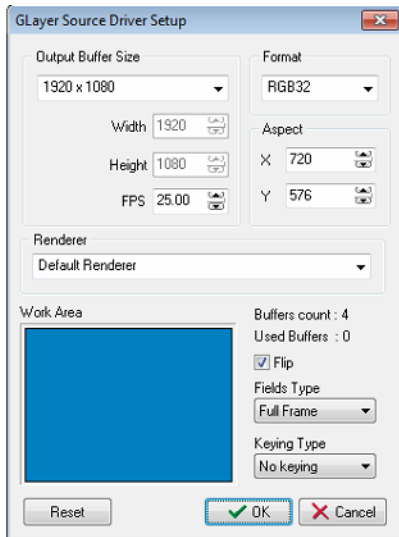
**External keying** is used when you want your alpha channel key, and the video graphic fill, to output through separate DeckLink SDI outputs for use with an external video mixer.

Please, refer to the DeckLink documentation for more information about supported keying.



## Preview Output Driver settings

This driver is used only for preview. It is not meant to be used as an output driver.



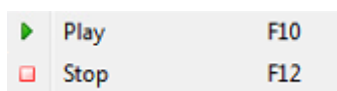
In the *Video Source* field you can select the background, on which **TitleBox** graphics will be overlaid. Just press on the radio button that corresponds to your needs. For **Image** and **Media File** you have to use the browse button and enter the file-path of your desired file, and for **Capture Device** you should select the device you are using from the drop-down menu.

From the *Filters List* field select the appropriate filter.

The *Output Monitor* field becomes available if you have more than one monitors, connected to your machine. Then you can select which monitor to show the preview window.

The *Options* area allows you to flip your video or graphics by 90 degrees by checking the relevant *Flip Video* and *Flip Graphics* boxes. If you check the  *Loop* box your Project will be looped.

The *Fields Type* drop-down list allows you to select whether to preview the **Full Frame**, **Field 1 First**, or **Field 2 First**.



Finally, the **Play** and **Stop** buttons at the bottom allow you to initiate or terminate the preview of your Project.

## Plug-ins

This item shows the list of available external plug-ins:

### File link

Opens a list of all objects in the project and their links, if any, to external files. Any changes in those files are applied in real time. The file links apply to the text and picture objects only!

### Task Manager

Select this menu item to create a list of tasks, related to objects in the current project. Check the detailed description in the [Task Manager](#) section.



## GPI Manager

In the GPI manager, you can view all GPI events and the objects, to which they have been assigned. To specify GPI events, you have to run [Task Manager](#) and make the necessary definitions there.

## Active Event (Tally)

This function keeps you informed whether there are any graphics showing on the screen or not. If there is at least one object that is being played, a high pulse is sent to the specified *COM* port. When there are no objects played, the pulse goes low. Thus, any external GPI device can be activated when there are no objects played out in **TitleBox** (i.e., a sound alarm to let the operator know that there are graphics displayed). After choosing this menu item, you can select the desired *COM* port or [None] if you do not want to send out any pulses. Furthermore, you can specify the signal type (RTS or DTR). RTS will send a pulse to pin 7, and DTR – to pin 4. Pin 5 is the ground. For more details, please, refer to the GPI pin out description in [Appendix 1](#).

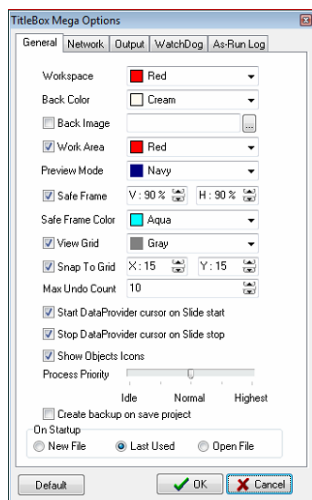
## DataSource Manager

This plug-in opens the **Data Source Manager**. It contains all supported data providing plug-ins. You can use them to “translate” different types of data and display the information that they contain in some of the **TitleBox** objects. Please, check the **Data Source Manager** description [further below](#).

## Options

Press it to open the **Project Options** window. It contains the following pages:


### General Options:



These options allow for defining the colors, the safe area, and the grid of the *Work area*:

*Workspace* – set a color for the area that surrounds the workspace.

*Back Color* – set a background color for the workspace.

*Back image* – you can select an image for background via the browser  button.

*Work Area* – set a color for the borderline of the work area.

*Preview Mode* - set a background color for the workspace, when it is in preview mode.

*Safe Frame* – check it to view the safe area in the workspace. You can adjust its size by using the [H] and [V] percentages to the right.

*Safe Frame color* – choose the color for the safe area border.

*View Grid/ Grid color* – check the box to see the grid and select its color. The grid is very useful when you resize your objects.



*Snap to Grid* – check this box to align the objects in the project to the grid. Use the [X] and [Y] spin-boxes to define the distance between two neighboring points in the grid.

*Max Undo Count* – specify the number of latest actions to be saved in the undo history. Please, note that the higher number here means higher memory usage.

**NOTE:** You cannot undo removing objects!

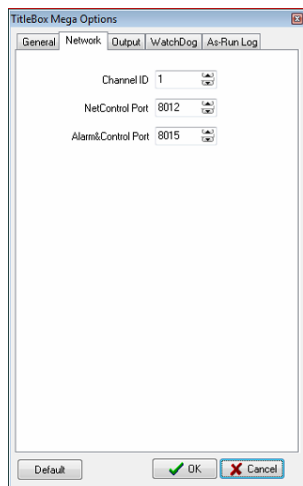
*Show Objects Icons* – check this box to enable showing the objects icons in the upper left corner of each object in the work area.

*Process Priority* – this slider will affect all other programs running on the system. Please, do not change the default setting, unless advised so.

*Create backup on save project* – check this box to if you want **TitleBox** to create a backup file of your project every time you save it. The backup file will be saved in the same folder, in which your current project is saved, and it will be with extension *\*.backup*

*On Startup* – here you can select the start mode of **TitleBox**.

## Network Options



These options are used for interaction between **TitleBox** and **AirBox**.

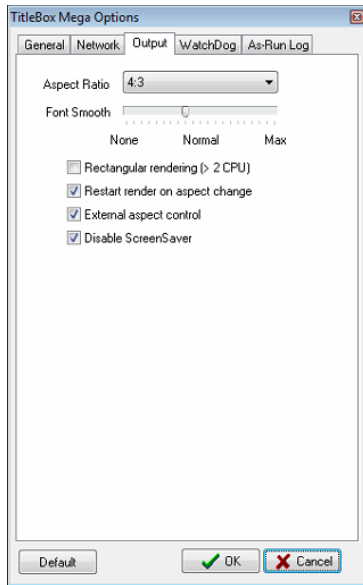
*TitleBox Channel ID* – **TitleBox**'s channel ID; it corresponds to [AirBox -> Settings -> Modules -> Remote control \[TitleBox Net Control\] -> Configure\[Channels\]](#).

*NetControl Port* – select a network port for receiving commands. This is the port, at which **TitleBox** receives commands from third-party applications (like **AirBox Net Control** commands)

*Alarm&Control Port* – select a network port for communication with **PlayBox** applications (like **AlarmBox** or **TitleBox Text Client**).



## Output Options



Here you can set the aspect ratio of the output. The default setting is [4:3]. The slider below determines the smoothness of the font characters. The greater the value, the smoother the font is.

- Rectangle rendering* is recommended for more powerful systems. It changes the algorithm of graphics rendering for optimizing the performance. Do not use it on single CPU machines, as it increases the CPU load!

**IMPORTANT!** This option with is NOT recommended with animation objects and Direct Show Media objects!

- Restart render on aspect change* – check this box to restart the rendering when the aspect ratio is changed.
- External aspect control* - it is intended for automated switching of the aspect ratio, according to an external source. Such a source can be WSS present in incoming video, for example.

**NOTE:** The  *External aspect control* check-box is related to the **External aspect control**  button on the **TitleBox toolbar**.

- Disable ScreenSaver* – disables the screensaver on the machine, on which **TitleBox** is in use.

*Stereoscopic 3D Output (SBS)* – enables the **TitleBox** stereo 3D output. Use the *Master Stereoscopic Offset* spin-box to assign the overall offset of the 3D output. The offset here measures the depth of the objects. If you set a positive number, the objects will appear deeper in the screen and vice versa.

**NOTE:** You can assign object-specific 3D offset values to each object by using the [corresponding option](#) in the main **TitleBox** toolbar.



## WatchDog

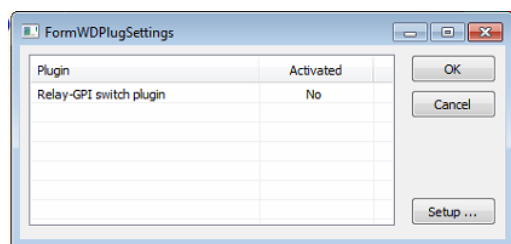


The **WatchDog** plug-in is used in **TitleBox** systems working on pass-through mode. Its purpose is to switch the by-pass relay card in case of **TitleBox** crash, in order to bypass a video signal.

The WatchDog works with a dedicated **PlayBox Bypass relay** card. You can read more about this card in [Appendix 4](#) in this manual.

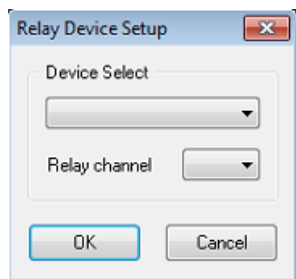
You need to have an installed **PlayBox Bypass** relay card in order to configure the WatchDog plug-in.

When you press the **Configure WatchDog** button, the following window opens.



Activate the plug-in first by double-clicking on the entry under the **Activated** column next to the corresponding **Plugin**, so that the [No] text in it turns to [Yes]. Then press the **Setup** button to open the set-up window:

In the *Device Select* field you will see the list of the installed **Bypass Relay** cards. Select the card you need and press **OK**.



**IMPORTANT:** Even if there is only one Bypass Relay card installed, select it from the *Device Select* field and press the **OK** button, for the setting to take effect.



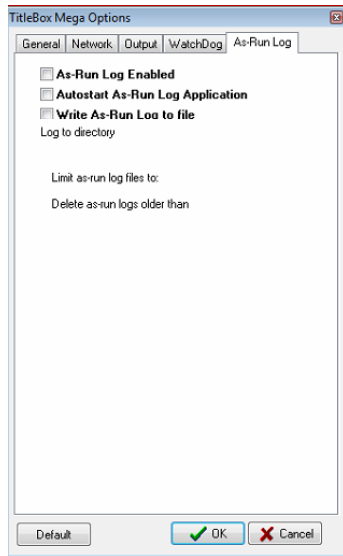


## **As-Run Log**

Use this tab to enable and configure **TitleBox** logging. Be aware that an as run log file is configured and saved per **TitleBox** instance and it is not larger than 2 GB.

The first three check-boxes allow you to do the following:

*As-Run Log Enabled* – check this box to enable logging. You can also enable/disable the [As-Run Log](#) via the corresponding [button](#) in the **TitleBox** toolbar.



*Autostart As-Run Log Application*– check this box if you want the As-Run Log to start automatically when **TitleBox** is initiated.

*Write As-Run Log to file*– check this box if you want to save your As-Run Log to file. Notice that once you enable this option, some additional options appear in the dialog.

In the *Log to directory* field enter the folder, where you want your log file to be saved.

The name of the logged file will be in the following format:

TBRL\_YYYY-MM-DD\_HH-MM-SS\_#n, where:

YYYY-MM-DD\_HH-MM-SS is the date and time when the first command in the file has been sent to the **As-Run Log** and #n is the respective **TitleBox** instance number.

For example, TBRL\_2012-06-19\_15-23-01\_#1.log is a log file, in which the first command has been sent on June 19<sup>th</sup>, 2012 at 15:23:01 o'clock from **TitleBox** Instance #1 to the **As-Run Log**.

The *Limit as-run log files to* spin-box allows you to set a limitation for the log file size in Megabytes. The minimum value that you can enter here is 1 MB, and the maximum is 2000 MB.

Be aware that the log files are saved per instance and per day, meaning that when the current date changes a new log file, for the next date, will be opened by the **As-Run Log** for writing. New log files are created also if the limit, set in the abovementioned spin-box is reached. The last case when the **As-Run Log** will create a new log file is if it is manually shut down by the user and reactivated by **TitleBox**.

Use the *Delete as-run logs older than* spin-box to automatically delete log files that are older than the configured number of days. The minimum number of days here is 1 day and the maximum is 180 days.



In the *Log items to file* area place a check next to the events, which you want to appear in your log file. The options here are self-explanatory. If you press the **Default** button only the options, which are checked in the picture above will be selected.

**NOTE:** These configurations refer only to the log file that is saved. The actual **As-Run Log** application displays all types of events.

## Help Menu

This module gives the opportunity to generate easy-to-complete problem reports. It is integrated in each **PlayBox** module. It can gather almost all the information, needed for the **PlayBox** support team in order to provide you with the prompt answers, without too many questions about your system configuration.

The Basic User's manual contains a detailed description of the PlayBox Doctor Report and other functionalities. If you do not have the Basic manual, you can download it from our website.

Displays the **About** box of the **AirBox** module. It contains useful information, such as: module version, WIBU Box number, mode, registration, etc. The name of the currently selected platform is displayed at the bottom.

## Help Index



Opens the **TitleBox** context-sensitive help. You can also invoke it by pressing <Shift + F1> key combination.

## CREATING OBJECTS

### Step By Step

All objects are created in a similar way:

1. Select the appropriate object button from the *Object Palette*.
2. Draw a rectangle in the *Preview Area*, where you wish to place it.
3. A **Properties** dialog box, allowing fine-tuning of non-text objects (Still picture, Analog clock, Animation, Digital Clock, DirectShow media, Banner, etc.), will appear.
4. Pressing **OK** inserts the created object in the project.

You can edit the properties of text-containing objects either by using the special toolbars that appear when double-clicking on the relevant object, or when clicking on the **Properties** buttons  . The unique properties of all types of objects in **TitleBox** are described in detail in their relevant sections in the [Object Properties](#) below.

## Editing Objects

You can edit objects either by invoking the object properties dialog box (for non-text objects), or by using the special toolbars (for text objects). Do this by double-clicking on the object, or by clicking on the **Properties** buttons.

**Animations** and **Clocks** CANNOT be resized in **TitleBox**, whereas **Pictures** and **Text templates** with graphics background can.

**Crawls**, **Rolls** and **Text Templates** without graphics background can be resized even on-air. You can resize all dimensions of Text objects, Crawl's width and text size, and Roll's height and text size. However, resizing a Crawl's height and a Roll's width should be done while the objects are stopped.



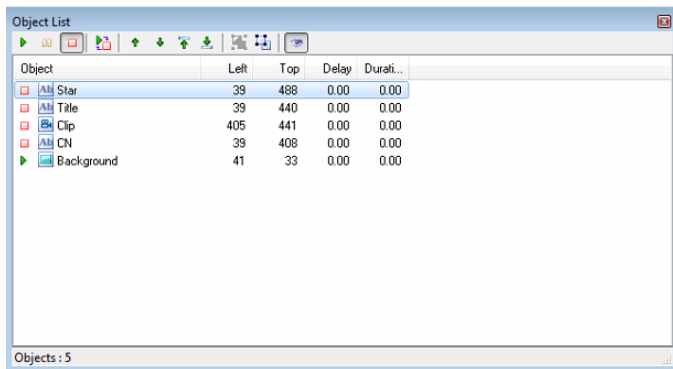
## Deleting Objects

An object can be deleted by selecting it and clicking on the **Delete** button on the **Main Toolbar**. It can also be deleted by using the <**Delete**> key on your keyboard.

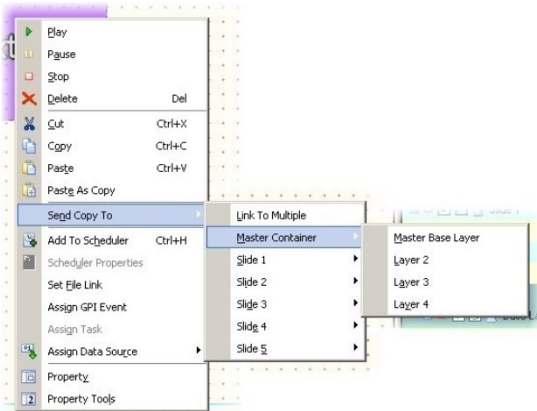
All objects can be deleted by clicking on the **New Project** button and clicking on the **No** button in the dialog that asks you if you want to save your changes.

## Object List

The **Object list** is intended for fast switching and reviewing of objects and their properties, such as **Left** and **Top** side positions, **Duration** and **Delay**. Here you can easily **Group/Ungroup** objects, change their **Order** (z-order), and control their playout and visibility status by pressing the relevant buttons on the toolbar.



## Object Properties



The **Object Properties** dialog boxes are different, depending on the object type. All buttons have specific pictures and provide hints when you slide the mouse pointer over them.

Right-clicking on any object invokes a menu, like the one shown in the screenshot to the left.

The upper half of the menu contains the most common status and editing commands.

The **Send Copy To** command allows you to easily create copies of the currently selected object of all slides and layers in your project.

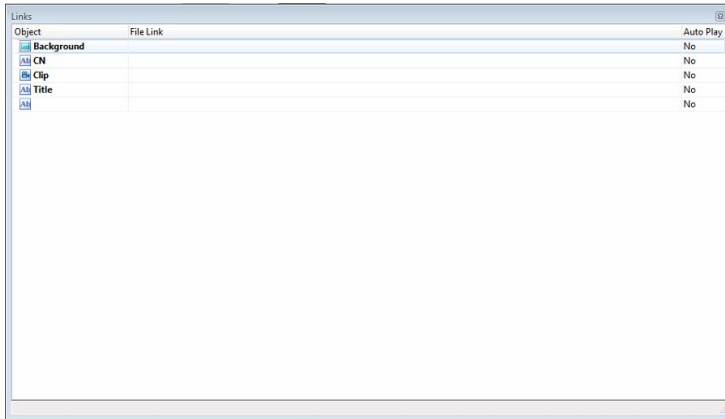
The **Add To Scheduler** item will open the **TitleBox Scheduler**, where you can set the scheduling for the selected object.

If the object has already been added to the **Scheduler**, use the **Scheduler Properties** item to view the settings of all objects in the current project.



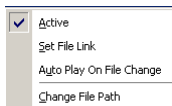
**Set File Link** – select this item to invoke a browse dialog to locate the text file (\*.txt or \*.rtf), to which you would like to link your object. Thus, your object's content will be updated every time the linked text file is saved. An additional window will open to display all objects and their file links (if any) in the current project. You can set/change the file links from within the **Links** dialog. Just press the browse button to the far right of the relevant row to open a new browse dialog.

If you right-click on the selected row, the following dialog will open:



If you do not want to have a link anymore, just uncheck the **Active** row. When you create a new link, **Active** is checked automatically.

Pressing **Set File Link** will open a new browse dialog for selecting a text file for the link.



**AutoPlay On file Change** is used for starting the object automatically, if there is a change in the linked text file.

**Change File Path** allows for changing the file path to the linked file. It is used in case the linked file is moved to another location.

**(!) TIP:** When a **Roll**, **Crawl**, or **Text** object is linked to a text file (\*.txt or \*.rtf), you can insert a still picture in the text – the image will be displayed among the characters, according to the position of its script in the text. Write the following command in the text file:

**<BITMAP>[file path of the image file]</BITMAP>**

Make sure that BITMAP is written in capital letters. Here is an example:

#### Example 1 – Inserting a Still Picture in a Text File

- 1) Save your image. Let us assume the file name is "logo.bmp" located on D:\
- 2) Enter the text in a file, for example "Hello, this is a test project".
- 3) Continue writing the following: <BITMAP>D:\logo.jpg</BITMAP>.

Thus, your text file will be:

Hello, this is a test project <BITMAP>D:\logo.jpg</BITMAP>

It could also be:

Hello, this <BITMAP>D:\logo.jpg</BITMAP> is a test projector Hello, this is a <BITMAP>D:\logo.jpg</BITMAP> test project

Every time you edit the text and save the changes, the text on the output will be refreshed.

If you want to change the picture, change the file name and location part in the script (here: D:\logo.jpg).

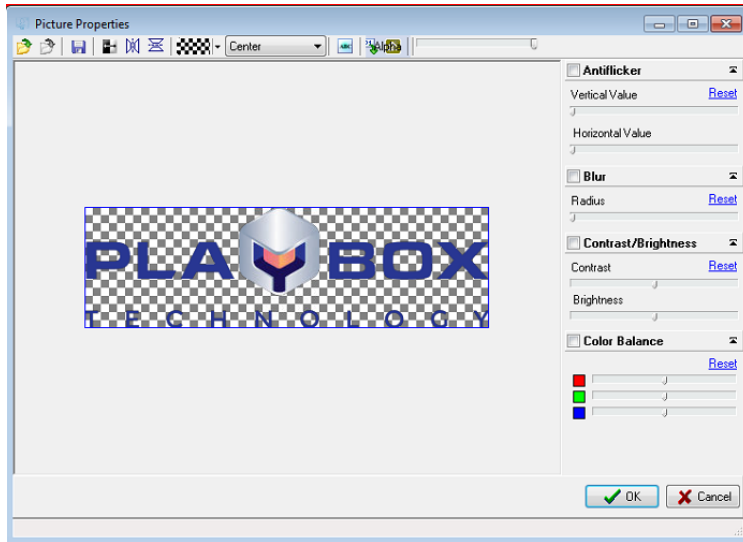
If you want to insert an animated GIF file in the text, write <MOVIE> instead of <BITMAP>. For example:

Hello, this is a test project <MOVIE>D:\smilie.gif</MOVIE>





**NOTE:** There is no spell-checker implemented in **TitleBox** text objects (roll, crawl and text template). You can use some external application for spell-checking and then just copy and paste the text into **TitleBox** object.

## Still Picture Properties



The **Picture Properties** dialog box looks like the one to the left.

The **Toolbar** allows you to **Open** and **Save** the image, **Load** and **Invert** the mask (the alpha channel), and **Flip** the image horizontally and/or vertically. If your picture does not have an alpha channel, you can import one separately, by using the **Open mask**  button.

The **Draw Alpha Only** button  provides a new, interesting option. Push this button to create a Picture object over other objects in the project. Thus, you will overlay its alpha over all underlying objects. Try to make effects this way!

At the right part of the window you can adjust the object's *Contrast*, *Brightness* and change its *Color Balance*. You can choose to *Blur* the object and set the blur radius by using the slider below.

The *Anti-flicker* option is designed for smoothing the high-contrast computer graphics when overlaying it over video. Change the *Vertical Value* to prevent flickering of the graphics' edges.


## Text Object Properties

The Text objects have three groups of settings options:


### Text and Background Properties

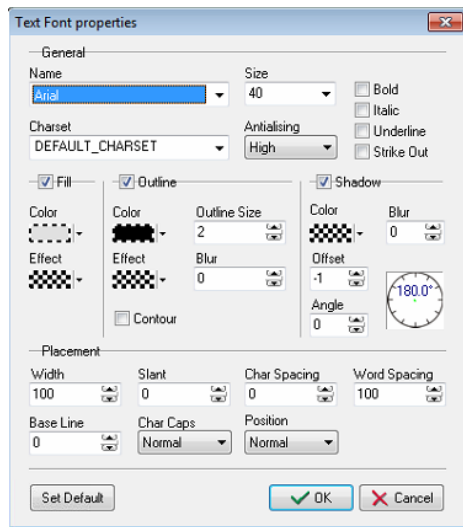
You can edit the common text attributes (such as font selection, size, color, blur, shadow, etc.) and background attributes from another object-specific toolbar:



This toolbar appears in the last row of the **TitleBox** toolbar when you double-click on a Text template object. Another way to invoke it is by pressing the Property button  (in the second row of the toolbar) while a Text object is selected (single click).

Write the new texts directly in the object!

Pressing  invokes the **FontDialogEx** dialog, which allows you to manage the font of the selected text. Here you can find all the formatting options, as known from other windows-based editing applications:

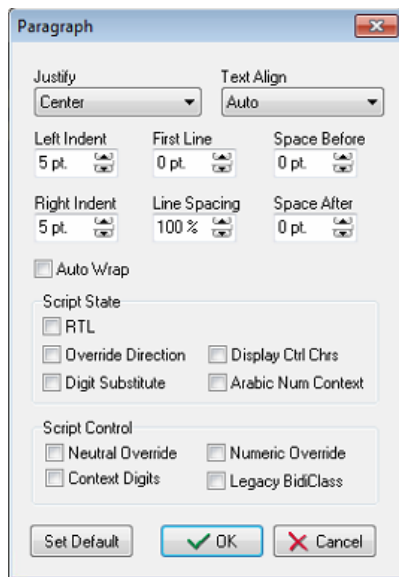


The **Set Default** at the bottom button restores the **TitleBox** native settings.


**NOTE:** **TitleBox** supports all true type fonts.

Pressing **P** invokes the **Paragraph**-formatting dialog box, where you can set all paragraph-formatting options.

The *Script State* and *Script Control* areas are related to the use of Arabic language and Arabic (Arabic-Indian) digits in the project.




The **Set Default** button at the bottom restores the **TitleBox** native settings.

The  button in the toolbar allows you to change the color of the background.

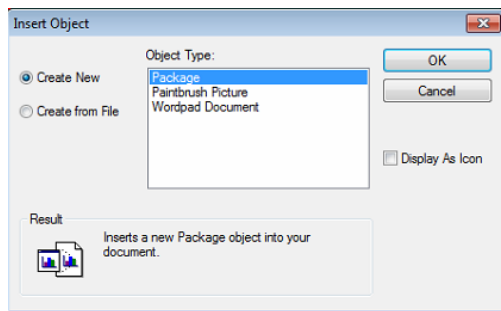
Press  to view a transparency background in the preview area while editing.

The following fields in the toolbar provide general text-formatting options (font, font size, bold, italics, underlined, text alignment, font color).

The  button inserts a still picture. Press it to open a browser dialog for selecting a picture image to be inserted inside the text into the text object.

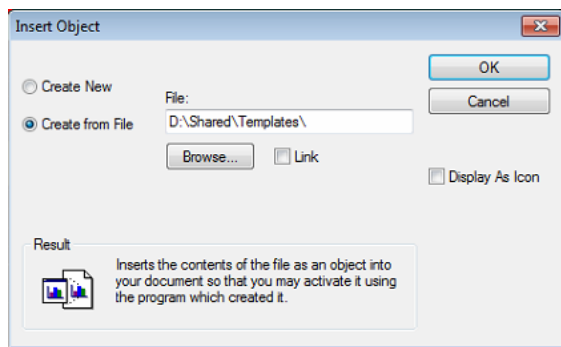


The  button inserts OLE compatible objects.



When you press it, the following dialog will appear:


If you choose  **Create New**, the relevant application opens, and you will be able to create the desired object. Any changes in the relevant **OLE** object will be visible in **TitleBox**, when saved.




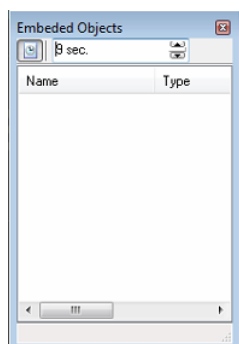
If you choose  **Create from File**, you will be prompted to browse for it. You can create a link, or display it as an icon through checking the relevant box.


The drop-down list for vertical alignment is used to manage the position of the inserted OLE objects into the text object.

## **Background properties**

If you click once on a text object, the following object-specific toolbar appears in the last line of the **TitleBox** toolbar: 

The  button is related to the objects (OLE object or pictures), inserted into the text. Press this button to open the list of all objects that are already inserted into the text objects. The insertion of the objects is described in details in the previous section.



You can set an auto refresh period for the object. Press the clock  button in the upper left corner to activate this function and set the refreshing period in seconds by using the arrows. Thus, if you update the original file, it will be refreshed automatically in your **TitleBox** project. In addition, right-mouse-clicking on an object in the list will invoke a context menu, containing several useful options:




**Update** – it is valid for OLE objects only;

**Size...** – you can set a custom size to your inserted item;

**Full size** – display the original size of the object;

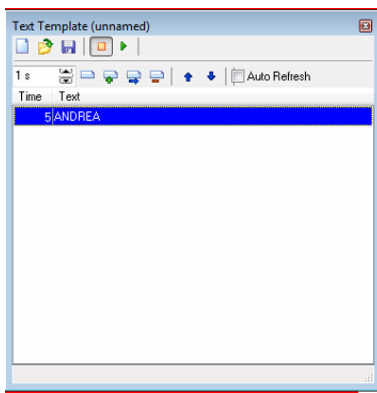
**Invert Alpha** – inverts an object's alpha channel;


**Update from file** – it is valid for picture objects only.

If you want to open an image as a background, push the **Open image** button . Use the drop-down list to the right to fix its layout in the object (Normal, Stretch, Tile or Center).


Delete the background image by pushing the  button.


## Text Object Property Tools





The property dialog box for text objects allows for displaying different texts consequently, for specified periods in seconds. The different texts are shown as a list of texts (separate lines) in the **Text Template** window – press the  button to view it (or right-mouse-click over the object and select **Property Tools**).


You can prepare texts in advance or create them online. Use the buttons in this dialog to do the following:

**New**  button – open a new text template.

**Open**  button – You can open a previously prepared text file, by using the **Open** button. Each paragraph in the text appears as a separate line in the **Text Template** window.

**Save**  button – saves the entered text as a file.


**Stop**  button - stops displaying the text lines in the preview/output window.

**Play**  button – starts displaying the text lines in the preview/output window.

**Global Time** spin-box – defines the frequency of changing the text lines. This is actually a “duration,” applied for each new line added in the text template. If you need to specify a different duration for a particular line, use the spin-box in the text input dialog.


**Edit item**  button – opens a dialog box for editing the selected text line.


**(!) TIP:** You can change the text by right-clicking on a text string as well.


**Add item**  button – adds a text line. A dialog box opens for typing the text in it.






**Insert item**  button – inserts a text line. A dialog box opens for typing the text in it.

**Delete item**  button – deletes the selected text line.

**Moving up**  – moves the selected text line up.

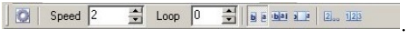
**Moving down**  - moves the selected text line down.


**Auto refresh** box – automatically displays the changes made in the currently loaded text file, even during its play-out.

## **Roll/** **Crawl Properties**

There are three groups of properties for these objects, similar to the text template: common text attributes; continuity mode and queue options; and dynamic speed properties. The first two are controlled through object-specific toolbars, while the third one can be set in the specially designated dialog box.

### Continuity and Queue mode


When a crawl/roll object is selected (single click) the following string appears in the last row of the toolbar: 


The **gear-wheel**  button opens the **Embedded Objects** list. This is the one already described for text objects. Please, check [above](#) in the Manual.



**Speed field** - controls the speed of dynamic objects, such as animations, crawls, and rolls. Its value can be positive (right-to-left movement) or negative (left-to-right movement). If zero, the object is frozen.

**NOTE:** You can change the speed interactively at any moment, even when the object is running on-air.

If the value in the **Loop** field is [0], the object will be displayed endlessly.

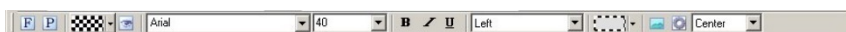
The next three buttons  refer to the object's background continuity mode. The first one means that the background will be displayed only while the text is running. The second one will "glue" the texts one after another, without any space in between. The third button will display the background continuously.


The last two buttons in this toolbar  concern the Queuing functionality. After pressing one of these buttons, **TitleBox** will "remember" the changes for background colors and text colors and it will play them one after another, i.e., you will be able to make a queue of color changes (loops). If none of the above buttons is pressed, **TitleBox** will show only the latest change. You can choose how to switch between changes (loops) while in play mode:

Use  if you want to display the subsequent loops one by one, i.e., to leave some space between them (empty or with background color, depending on your settings – see the previous paragraph). In addition, use  if you do NOT want any space between the successive loops in the object. **TitleBox** will generate one loop out of all loops in the queue.

### Text and Background Properties

You can edit the common text attributes (such as font selection, size, color, blur, shadow, etc.) and background attributes from another object-specific toolbar:



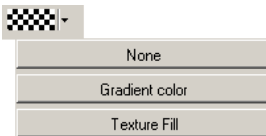
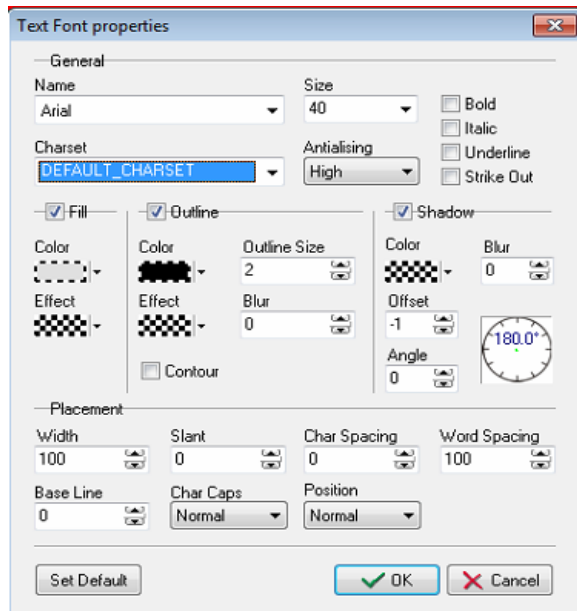
It appears in the last row of the **TitleBox** toolbar when you double-click a Roll/Crawl object. Another way to invoke it is by pressing the Property button  (in the second row of the toolbar) while a Crawl/Roll object is selected.

Write the new texts directly in the object!



**NOTE:** There is no possibility to insert images as background but you can still insert images and OLE objects in the Roll/Crawl objects.

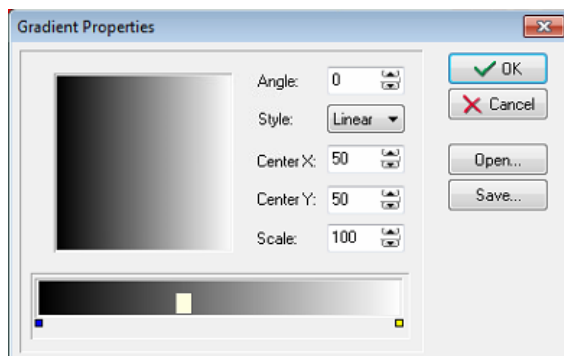
Pressing **F** invokes the **FontDialogEx** dialog, which allows you to manage the font of the selected text. Here you can find all the formatting options, as known from other windows-based editing applications:



Use the **Fill Effect** button and its additional buttons to modify the colors for your text. You can select **None** for color, **Gradient color**, or **Texture Fill**.

If you click on **Gradient color** from the drop-down list, you can select from 255 levels of graded transparency and a vast variety of colors.

In the **Gradient properties** dialog, fix the desired settings:



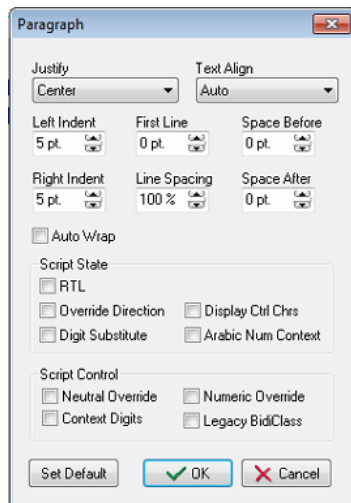
- Choose a Style for the gradient from the drop-down list.
- In the square field to the left drag the cursor to change the gradient positioning.
- In the rectangle field below, define the gradient colors and their initial points.



- Place the plus-sign cursor in the desired position and click. A black triangle will appear in that position to mark the currently selected point. All other marks will become white.
- Double-click on the black triangle to invoke the **Color setting** dialog.
- Modify the color for this color-change point at your will and click **OK**.

Back in the **Gradient properties** dialog, you can change the position of the point by dragging it. Also, you can add as many color-change points to the gradient, as you like. Finally, click **OK**.

Pressing **P** invokes the **Paragraph**-formatting dialog box, where you can set all paragraph-formatting options:



- The two buttons to the right concern the background. Press to view a transparency background during editing in the preview area.
- Text-formatting buttons - font, font size, bold, italics, underlined, text alignment, font color.
- Object links buttons.
- Push the button to insert a still picture object link.
- Push the button to insert OLE compatible objects. Use the drop-down list to fix their position within the Roll/Crawl object.
- The [Text Template Properties](#) section above contains a detailed description on how to import **OLE**-compatible.

**(! TIP:** When a **Roll**, **Crawl** or **Text** object is linked to a text file (\*.txt or \*.rtf), you can insert a still picture in the text – the image will be displayed among the characters, according to the position of its script in the text. Write the following command in the text file:

**<BITMAP>[file path of the image file]</BITMAP>**

Make sure that BITMAP is written in capital letters. Please, check Example 1 [above](#).

- Save your image. Let us assume the file name is "logo.bmp" located on D:\
- Enter the text in a file, for example "Hello, this is a test project".
- Continue writing the following: <BITMAP>D:\logo.jpg</BITMAP>.

Thus, your text file will be:

**Hello, this is a test project <BITMAP>D:\logo.jpg</BITMAP>**

It could also be:

**Hello, this <BITMAP>D:\logo.jpg</BITMAP> is a test project**

or

**Hello, this is a <BITMAP>D:\logo.jpg</BITMAP> test project**



Every time you edit the text and save the changes, the text on the output will be refreshed.


If you want to change the picture, change the file name and location part in the script (here it is **D:\logo.jpg**).

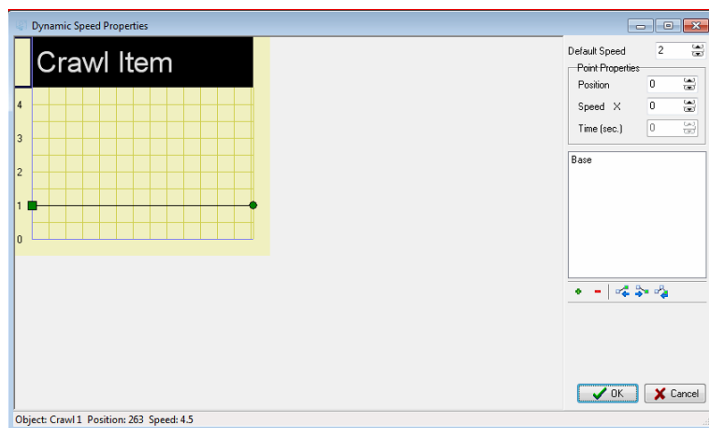
If you want to insert an animated **\*.gif** file in the text, write **<MOVIE>** instead of **<BITMAP>**. For example:

**Hello, this is a test project <MOVIE>D:\smilie.gif</MOVIE>**

**NOTE:** There is no spell-checker implemented in the **TitleBox** text objects (roll, crawl, and text template). You can use some external application for spell-checking and then just copy and paste the text into the **TitleBox** object.

## Dynamic Speed Properties


Pushing the Property tools button  while a Roll/Crawl object is selected will open the Dynamic properties dialog box:



This property dialog box allows you to specify different speeds of the **Roll**'s and **Crawl**'s movement.

The movement is represented graphically and you can define the speed of each point of the graphics. The horizontal axis of the graphics represents the position of the **Crawl/Roll** on the screen. The vertical axis represents the speed multiplier (0; 1; 2; etc.) of the default speed, which is set in the main screen (see the [Toolbar](#) section above). The zero value means **0x**default speed, i.e., the object does not move; one means **1x**default speed, i.e., the object moves with the default speed; two means **2x**default speed, i.e., the object moves twice as fast as the default speed, etc.


On the top of the graphics, you can see the object's (**Roll/Crawl**) text. By moving the mouse pointer over the grid (the blue lines) or by using the arrow keys, you can select the position in the text, where you would like to change the speed. The text section, which will be displayed at the selected "speed change" point, is enclosed in a frame.

By default, the first point is situated at the beginning of the graphics. A new point is added by pressing the **Add**  button or by double-clicking in the yellow-squared area.

When you select a position to change its speed, a green point will appear in the grid and its properties (*Speed* and *Position*) will be displayed in the *Point Properties* area to the right. The position's coordinates are also displayed in the status bar.

If you set a speed of '0' for any position, then you will have to define a delay period. This is the period (in seconds), during which the object will remain stopped. The wait-time appears in a red square under the zero-point.

All points are shown in the *Point list* to the right of the graphics. Their names are [Point #], where the # stands for the sequential number of the point.


The **Align buttons**  allow for aligning the selected point toward the previous, the next or the first point in the graphics.


The *Default speed* field shows the default speed, as it is defined in the main **TitleBox** window (see the [Toolbar](#) section above).

In the *Point Properties* area, the following properties of the selected point are shown: *Speed*, *Position*, and *Time* (for zero-speed points only).



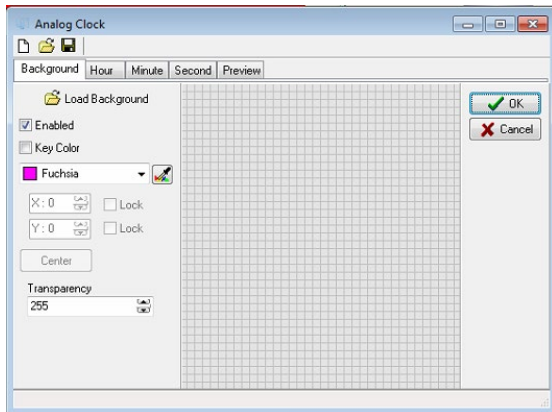
The *Point list* shows the list of all “speed change” points in the object.

The **Add** button  adds a new “speed change” point in the object graphics.

The **Delete** button  deletes the selected “speed change” point in the object graphics.



## Analog Clock Properties



The **Analog Clock** properties dialog box looks like the one above.

The **New** button opens an empty clock property object.

The **Open** button loads a previously created clock object (\*.clc).

The **Save** button saves the current clock image into a file (\*.clc).

There are different pages for each clock layer – background (clock plate), hours, minutes, and second hands, as well as a preview page of the overall clock layout.

All pages have an identical structure: a *settings area* and a *preview area*.

### Settings:

**Load Image** – loads the relevant image (for the background, hour, minute, or second hands).

/  *Enable* – enables/prohibits displaying the relevant element.

/  *Key Color*– key color for the image. If the image does not have a mask, you can select the key color.

/  *Lock position* – locks the [X/Y] position of the image.

*Transparency*– sets the image transparency

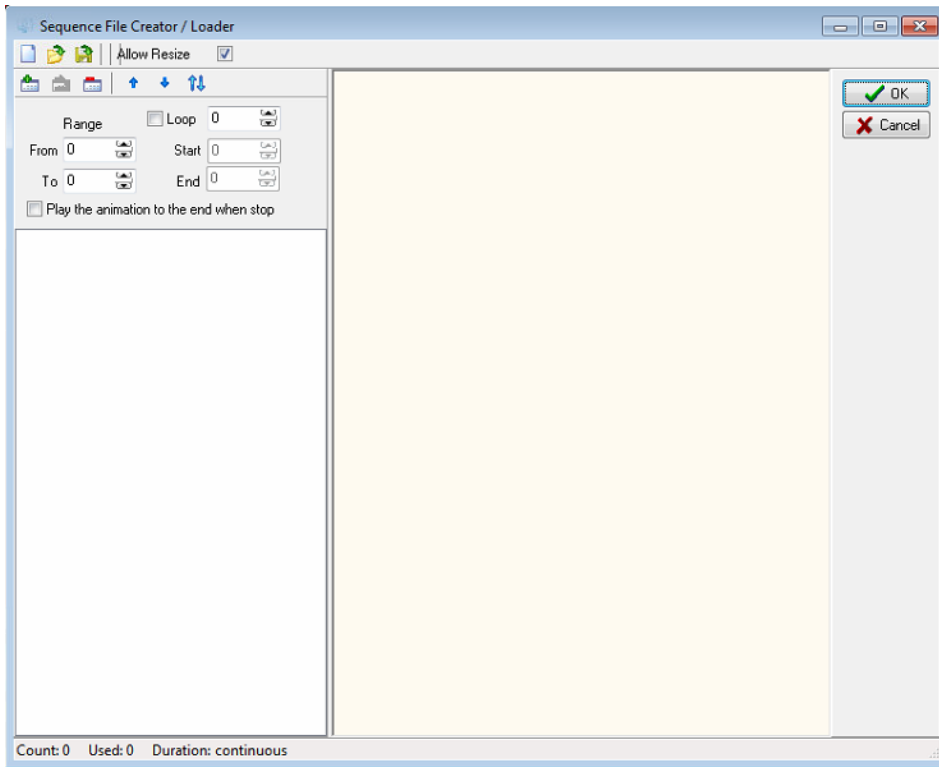
Preview area– It is used for previewing the corresponding clock element.

**(!) TIP:** To achieve a satisfactory result, use a picture-editing application to create four square images with equal dimensions. Save them in separate files – one for each element of the clock (background, hour hand, minute hand, and second hand). Be sure to place the hands' ends on the exact centers of the relevant images. Keep in mind that the clock object will have the same size as the image in the file. It cannot be resized!



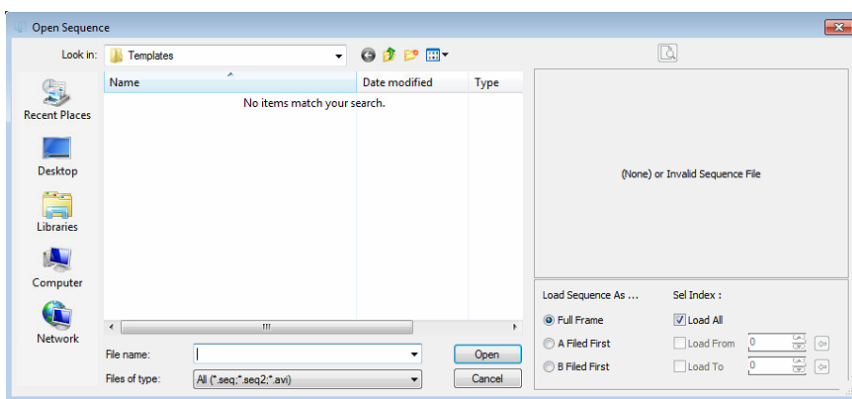
## Animation Properties

The **Sequence Property** dialog box looks like this:



The **New** button allows for creating a new sequence.

The **Open** button loads a file (\*.seq) or a sequence of files (\*.tga) for an animated logo:



Besides, you can load animated \*.gif files. If you use the **Add** button, **TitleBox** will load only the first picture of the file.

The **Open** button, **TitleBox** will load the entire range of \*.gif images.


**NOTE:** We do NOT recommend using \*.gif sequences for high quality applications, since they have only 256 indexed colors and do NOT have 8-bit transparency (just one color can be either entirely transparent, or entirely solid).


# TitleBox User Manual




If you want to load only a part of the files, uncheck the  *Load All* box and enter either the number of the first or the last file of the sequence, or both, by checking the  *Load From* and  *Load To* boxes and filling the relevant spin-boxes. Use the blue arrows to the right to enter the number of the currently selected file in the relevant field. If you are loading an interlaced animation, specify the first field – **ⓐ Field First** or **ⓑ Field First**.


Click **Open**. The sequence will be displayed in the animation property window. The currently selected file from the sequence will be shown in the preview area to the right.


The **Export**  button saves the current sequence as a file (\*.seq).

The **Add**  button adds a new file to the sequence.


The **Delete**  button deletes a selected file from the sequence.


The **Insert**  button inserts a file into the sequence.

The **Invert alpha**  button inverts the alpha channel of the selected file.

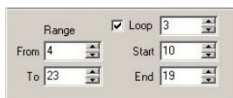
The **Move Up/Down**  buttons move the selected file up/down the list.

The **Reverse**  button reverses the files' order

The **View**  buttons are used for changing the sequence files' view to **list** or **thumbnail** mode.

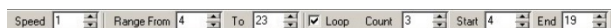
If the animation files do NOT have an alpha channel, you can select a key color by using the **Key color** tool: 

The *Range* and the  *Loop*-related spin-boxes are situated below the key-color setting kit. Select the *Range* of frames that will be used in the sequence – enter the values you wish in the *From* and *To* fields. The names of the frames that are out of this range will become red, and their background – grey. The background of the working range will remain white. If you want to loop between two frames in this working range, place a check in the  *Loop* box, and fill in the *Start* and *End* fields, as well as the number of loops. The frames, included in the loop will be highlighted in pale yellow. A blue arrow to the right of them will mark the final and the initial frame of the loop.



Range	<input checked="" type="checkbox"/> Loop	3	
From	4	Start	10
To	23	End	19

After adding your animation to the preview area, an additional toolbar will become active. In order to see it, click on the animation object:



Select the desired *Speed* and enter the number of times you want to  *Loop* the animation.

This taskbar duplicates the range and loop settings in the **Properties** dialog box. You can change those settings here as well. Just choose a *Range* of animation frames to be displayed, or check/uncheck the  *Loop* option (it enables leader-loop-trailer functionality). When you check  it, select the range of frames, within which you want to loop. If this option is selected, the animation will start from the beginning. Run to the *End* frame and loop between *Start* and *End* frames. If you want the animation to run regularly again, simply uncheck  the *Loop* box.

**NOTE:** The *From* and *Start* fields represent the first frame that will be shown, and the *To* and *End* fields represent the first frame that will NOT be shown. Thus, the difference between the *To* and the *From* (and between the *End* and *Start*) values will be equal to the number of frames that will be shown in your sequence. These numbers will be displayed in the status bar of the sequence's properties dialog box. *Count* stands for the total number of frames in the list, *Used* stands for the number of frames in the working Range, and *Loop* stands for the number of frames that will be looped.

**IMPORTANT!** You must have enough uninterrupted free RAM to load a TGA sequence in **TitleBox**. The minimum free RAM needed for loading a TGA animation is calculated with the following formula: (Animation\_Width multiplied by Animation\_Height, multiplied by 4) multiplied by the Animation\_Frame\_Count.





## Example 2 – Minimum Free RAM for Animation Sequence

To save or open a project, which contains such an animation, you will have to multiply the needed RAM by 2.

Thus, the minimum free uninterrupted RAM for saving/opening a 100-frame animation with 300x200 frame size will be:

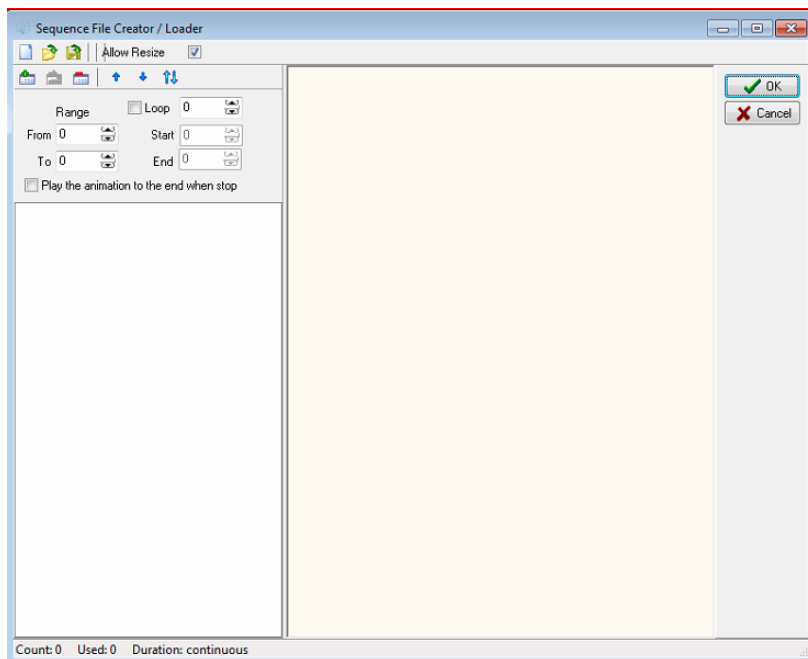
$(300 * 200 * 4) * 100 * 2 = 48,000,000$  bytes = ~45.77 MB.

**TIP (!)** In order to reduce these requirements, you could export your longer animations to sequence files by using the **Create Animation File** object (see the section below).

**NOTE:** Fielded animations are NOT supported! Use frame-based animations with each field in a separate file. Then, use speed 2 for proper speed and field interpretation.


**NOTE:** The Animation/Sequence is limited to 32-bit TGA.


## Create Animation File



.This object was created to avoid the real-time rendering of sequences, thus, reducing the system resources' load during playout. We recommend using it for large-sized animations with numerous frames. The resulting file format is **\*.seq2**.

The properties of this object appear in the dialog above. It is similar to the *Animation properties* dialog and most of the options are the same.


To load an already existing sequence file, click on the  **Open** sequence button.

To create a new sequence file, click on the  **Add Images** button and browse for them. In the **Open sequence** dialog, select the images to load and click **Open**.



You can use the arrows to change the image order.

In the *Range* spin-boxes, specify which images should be included in the sequence. All of them are included by default.

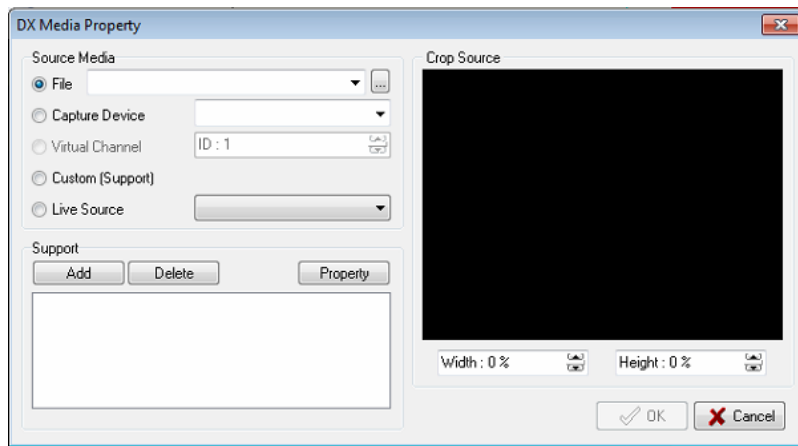
Once you have loaded all the frames you want to include in the sequence, press the  **Export Sequence** button, and save the file.

**NOTE:** Fielded animations are NOT supported! Use frame-based animations with each field in a separate file. Then, use speed of '2' for proper speed and field interpretation.

**NOTE:** The Animation/Sequence is limited to 32-bit TGA.

## Direct Show Media Properties

**TitleBox** allows for inserting all kinds of Direct-Show compatible media.



First, you have to specify the source media:

The **File** string contains the file path or URL to play in the object. Press the **Load** button to the right to browse for it.



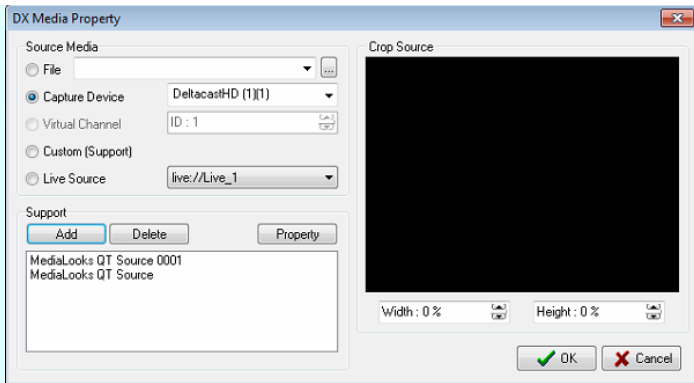
**NOTE:** In the dialog you should specify the URL of a live network stream, not a link to a file.

The **Capture Device** drop-down list contains all direct-show compatible capture devices installed in the PC (like FireWire camera, DeckLink card, etc.). Select one of them.

**Custom (Support)** – this is an advanced option. Click on it to select the filters to use in this object on your own. You can add filters to the graph by clicking on the **Add** button. A list of all available filters will open for you to select from. To view the properties of a filter in the graph, select its line in the list and press **Properties**.

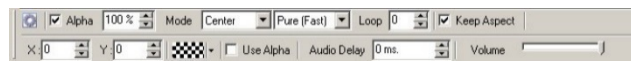
### Example 3 – Direct Show Media Capture Device


If there is a separate DeckLink card installed, **TitleBox** could use it as a capture device. It will be visible as DeckLink Video Capture (2) in the capture list, as in the screenshot below:



You can crop the image [Width] and [Height] by using the relevant spin-boxes under the preview window.

When a direct show object is selected in the **TitleBox** preview area, an object-specific toolbar appears under the standard ones:



The  **Properties** button opens a list of all filters used in the current graph.

Check  *Alpha* to use the video's alpha channel. You can adjust it by using the percentage spin-box to the right.


In this toolbar, there are two drop-down lists: one for selecting the display *Mode*, [Stretch] or [Center], and another one for the scaling quality. If the [Pure (Fast)] quality is selected, the CPU usage is lower.

Enter the *Loop* number to repeat the video as many times, as you want. *Loop* = [0] means that the video will be endlessly repeated; *Loop* = [1] means that the video will be played once; *Loop* = [2] means that the video will be played twice, and so on.

Any loop number different from zero represents how many times the object will be looped.

*Keep Aspect* – check this box to preserve the aspect ratio when resizing the object.

The next two spin-boxes control the image's[X] and [Y] offset in relation to the object's center. Use them while in [Center] mode, to move the video vertically or horizontally within the object boundaries.


When in [Center] mode, you can use alpha matte to fill-in the space between the edges of the video and the object's borders. You can select its *Color* from the  palette to the left of the  *Use Alpha* check-box.

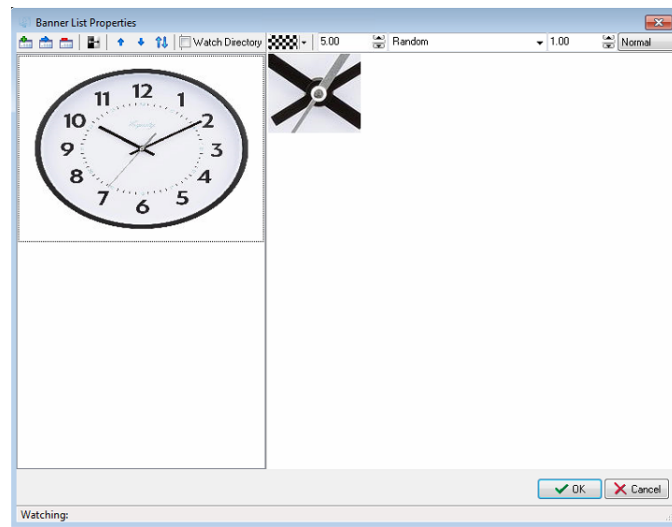
The two sound controls are situated at the end of the toolbar. The *Audio Delay* spin-box allows for adjusting the A/V sync of the object. To the right of it you can find the *Volume* control.


**NOTE:** The audio of your DirectShow objects will be output on the Default Audio device, set in Windows **Control Panel**→**Sound and Audio Devices**⇒**Audio**.


**WARNING!** As the playout of direct show media objects is carried out by third-party filters, we cannot guarantee the A/V sync of these objects.


## **Banner Properties**

Pushing the **Insert Banner**  button invokes a dialog box for you to create a list of picture files. They will be displayed as a slideshow in this object:



Use the respective buttons  to **add**, **insert**, and **delete** pictures from the list; **invert their alpha**, move them **up** and **down**, as well as **reverse** their order.

When you select a picture from the gallery to the left, it is displayed in the preview window to the right. The buttons above it are relevant for the currently selected picture. The  button allows you to choose a background color for it. Specify how long it will be displayed (in seconds), the transition pattern and duration, as well as the picture's layout (from the drop-down list in the upper right corner).


If you would like to set the same parameters for all pictures, included in the banner, simply press the **Set As Default**  button.

Thus, the settings you have already made will affect all newly inserted images in this object.


When you check the  *Watch Directory* option, a browser dialog will open for you to specify a folder (watch folder) that contains picture files. Opening the folder will activate a kind of slideshow in the banner and the images from the watch folder will be shown in the banner object in a random order.

**NOTE:** Keep in mind that all files, used in the banner objects are kept into the project. In this way, the size of the project is increased and it is possible to overload it. To avoid such overloading, we recommend using the *Watch Directory* option.

## Chat note objects

Another object from the object palette is the **Chat note** object . This kind of object can be used together with some third-party applications, like SMS, chat, etc. It creates a text file. You can insert a file link ([Project → Plugins → File Link](#)) in this text file. The chat note will be updated every time the relevant file is saved. The object's properties are controlled in the same way as the other objects (see the [Text object properties](#) section).

A **Chat Note** can be also treated as a **Textobject** but when you enter a text in it, the text is always shown as a new line coming from the bottom, and the old text is rolling to the top. The information entered is not saved into the project.

When you click on this object once, an additional object-specific toolbar will appear below the standard ones. Specify the desired speed for changing the text lines in the chat note in seconds via the *Speed* spin-box: 

When you double-click on the object, a window will appear. The properties are the same as for the Text object's [Text and Background properties](#).

**IMPORTANT:** When the object is linked to \*.txt file, then the font formatting is taken from **TitleBox** font properties. When the object is linked to an \*.rtf file on the other hand, then the font formatting is taken from the \*.rtf file itself.



## Sound objects



The sound objects are actually links to DirectShow-compatible sound files. After drawing the object rectangle in the work area, a browse dialog opens for you to locate the sound file. When a sound object is selected, the following toolbar appears under the standard toolbars in **TitleBox**:



Here you can specify a number of loops for this sound and change its appearance in the work area from the color palette.

The sound filename is displayed to the right. If you want to change the sound or the file path, double-click on the object to invoke the browse dialog.

**(!) TIP:** You could apply transition effects to your sound objects via the *In* and *Out* transition toolbar, situated above the object-specific toolbar. It contains two drop-down lists of transition effects and two spin-boxes for the *In* and *Out* transition duration.

Select [Fade] from the *In* drop-down list and enter some figure in the spin-box next to it. Thus, the sound volume will increase gradually according to the time set – the longer the time, the slower the increase.

Select [Fade] from the *Out* drop-down list and enter a figure in the spin-box to the right of it. Thus, the sound volume will decrease gradually at the end of the sound file.

**WARNING!** Be careful when moving **TitleBox** projects containing sound files! If the sound file is not available at the new location (its file path is not the same as the one saved in the project), the sound object will not be executed!

**NOTE:** The sound objects are quite heavy. Inserting more than three sound objects per project could slow down the rendering of **TitleBox** objects.

**NOTE:** This object's sound will be output through the Default Audio Device (set in Windows **Control Panel** → **Sounds and Audio Devices** ⇒ **Audio** tab).

## Digital Clock Properties



The default **Digital Clock** object looks like the one to the right:

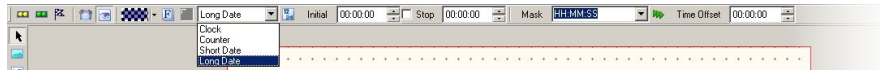
It displays the current system time. You can change it by using the *Time Offset* spin-box (see the bottom of this page).

A text-formatting toolbar appears upon double-clicking on the digital clock object:



Use this toolbar to edit the digital clock's common text attributes (such as font, size, color, blur, shadow, etc.) and background attributes. This toolbar has already been described in the [Text template properties](#) section above.

If you click once on your **Digital Clock** item, the toolbar below will appear below the standard **TitleBox** toolbars



The **Digital clock** has four modes of operation. It can run as a [clock] or a [counter], or it can display [Long] or [Short Date].

While in [clock] mode, the object runs as an ordinary digital clock. You can choose its time format through the **12/24 hours clock** button . The twelve-hour time format is active when the button is pressed, and the twenty-four hours format – when it is not pressed.

The background color is changeable through the **Palette** button.

Select a mask for the clock in the **Mask** drop-down menu. Update the mask by using the green arrows button next to it. If you have changed the mask, you will have to press this button to show it on the display.

In [counter] mode some additional options are available. You can select the counter to be countdown by pressing the **Countdown** button. If you push this button, the counter will count from an **Initial** time to a **Stop** time or until the  **Stop** box is checked. If not pushed, the counter will count up. After selecting the counter type and entering the **Initial** and **Stop** times, as well as a **Mask**, you are ready to run your **Counter**. Run your project and show your object by using the well-known **Play** button. Then, **Prepare** your counter and **Start** it.

After pressing the **Prepare** button your counter sets to **Initial** time and waits for a **Start** command. **Stop** the counter with the button. Pushing the **Intermediate** button will “freeze” the counter. During freeze, the counter will be running in the background but you will see a still frame, displaying the time of the counter from the moment, when the **Intermediate** button has been pressed. When you push the **Intermediate** button once more, the counter will start showing the current time again.

Two buttons are active for both the [clock] and [counter] modes – **Properties** and **Visible** . Pressing **Properties** opens the font formatting dialog. The **Visible** button determines if the Clock/Counter will be visible on the monitor, or not. It does not stop the clock/counter but it just hides/shows it.

**Time Offset** – This spin-box allows for creating digital clocks for different time zones.

In [Date] mode the object will display the system date instead of the system time. There are two date formats: [Long Date] and [Short Date]. Their appearance depends on the regional settings of your **TitleBox** server. You can change them in Windows **Control Panel**→**Regional and Language Options**→**Regional Options**. After pressing the **Customize** button, go to the **Date** tab. There you can check your current system date settings and change them at your will.

## Flash objects




Flash objects, similar to sound objects, are actually links to flash files. To create a flash object, press the flash button in the **Object palette** and draw a rectangle. Two windows will open as soon as you release the mouse button: a standard browse dialog to locate the flash file and an interactive properties window. The name of the Flash object is displayed in the caption of the properties window.


When a flash object is selected in the work area, the following toolbar appears underneath the standard **TitleBox** toolbars:



It provides the following options:



Press the **Pointer** button  to show your mouse pointer on the output. Thus, the operator's actions will be shown on the monitor.

Use the **Invert** button  to change the color of the pointer that is displayed on the output.

Select the mode, in which your flash should be displayed from the first drop-down list to the left. It contains all standard flash settings. Make your choice in accordance to the system capabilities of your PC.

Use the second drop-down list to set the quality of your flash image. Practically, this is an anti-aliasing setting of the flash object.

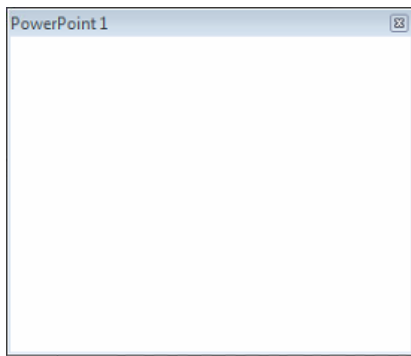
You could select a color from the last drop-down list to the right and make it transparent by checking the  **Use Key Color** box.



The last check-box to the right is related to aspect-ratio incompatibility. If your flash has a 4:3 aspect ratio and the output (in [Project menu](#) [→Options](#) [→Output](#)) is set to some other aspect ratio (such as 16:9, for example), checking this box will ensure the correct displaying of your flashes in the working area.


**NOTE:** You must have a Flash Player installed on your PC in order to play these objects.

**WARNING!** Do NOT close the preview window of the flash object and do NOT minimize the **TitleBox** window.

## **Power Point Objects**



**TitleBox** allows you to insert Power Point presentations in your projects. Just press the **PowerPoint Presentation**  button from the object palette and draw a rectangle in the work area. Then browse for the file location. You can change the file later by double-clicking in the Power Point object or by pushing the **Properties**  button in the toolbar.

If you right-click on a PowerPoint object or push the **Properties 2**  button, the PowerPoint properties window will open. It is interactive and allows you to control your slide shows.

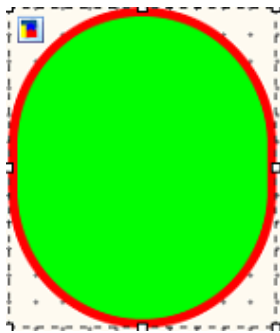
The name of the currently selected object is written in the window caption.

**WARNING!** Do NOT minimize the **TitleBox** window while displaying a power point presentation – this will hide it from the output! Do NOT close the preview window of the object.

**IMPORTANT:** You must have a full PowerPoint version installed on the **TitleBox** PC in order to run these objects. Only PowerPoint 2003 is supported.

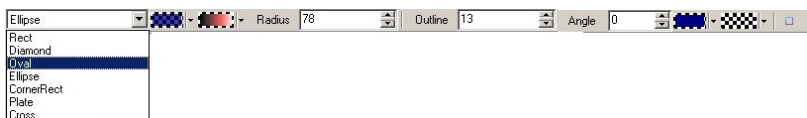


## Primary Shapes



You can create your own mattes for the text objects directly in the **TitleBox** work area. They are true-color with 256-level transparency.

Press the **Create Shape** button in the object palette and draw a rectangle in the work area. By default, an oval shape will appear in it, and the object-specific toolbar will appear under the standard ones:



The toolbar provides a wide variety of editing options. The drop-down list of available shapes is situated to the far left. The color palette next to it is designated for the background color. You can select it from the available ones, or create it on your own.

Further to the right there is another palette for the fill effects. You can choose between **None**, **Gradient color**, or **Texture Fill**.

The fill and outline effects are the same as in text objects. Check the Roll/Crawl [properties section](#) above.

**NOTE:** The background color serves as an alpha channel to the fill-effect. The background color (and transparency) will be mixed with the fill-color and its transparency.

Use the **Radius** spin-box to enlarge your objects or to make them smaller.

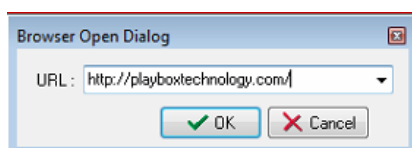
The next spin-box determines the **Outline** width. The color palettes next to it are for the background color of the outline and its fill effects respectively. Similar to the shape background and fill, these will be mixed as well.

The **Angle value** is in degrees and it is valid for **polygon** and **star** shapes only.

The **Square** button  in the far right of this toolbar fixes the aspect ratio of the shape object. It will change your object to fit a square and will keep this shape when you resize it.

**WARNING!** You will NOT be able to revert to your previous shape after checking the **Square** check box.

## Browser Properties



The browser object uses Internet Explorer to display web pages as graphics on the screen.






As soon as you release the mouse key after drawing the object's rectangle, the Open page dialog will appear in the work area. Type in the *URL* you would like to browse, or paste it from the clipboard.

Once you click **OK**, the object-specific toolbar will open under the standard toolbars:

In it, you can type another *URL* to be loaded in the browser object:

URL :

The **Properties 2**  button will invoke the Browser Open Dialog. In it, you can see the currently loaded web page. If the object's dimensions are smaller than the page itself, you can use the scrolls to select which part of the page to be shown on screen.

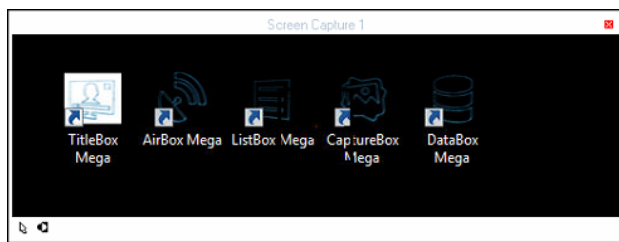


**IMPORTANT:** If a link in the webpage is set to open in a new window, you will not be able to open it in the same browser object. There is no practical way to grasp such a window into the same object.

**WARNING!** Do NOT minimize **TitleBox** while using browser objects! This will make the objects disappear off the screen!

**WARNING!** Do NOT close the preview window, as it might freeze the source web site.

## Screen Capture object



This object will allow you show some parts of your desktop on the screen.

The window above will appear as soon as you create the screen capture object. Its size is the same as of the rectangle you drew in the work area. Drag it to the area you would like to show on the screen.

In the upper right corner, there is a two-sided arrow button. Use it to minimize/maximize the main **TitleBox** window but leave the Screen Capture window ON. Please, note that if you use the minimize/maximize buttons in the main **TitleBox** window; the screen capture dialog will be affected as well.

In the lower left corner, you can find two other buttons that control the mouse pointer. The first one is for showing/hiding it from the screen. The second button will invert the cursor color. Thus, if the background of your desktop is light, you can turn the cursor black, so it will be easier seen on the screen.

When a screen capture object is selected in the work area, the object-specific tool bar will appear under the regular toolbars:





The first two buttons in it duplicate the ones for cursor control in the lower left corner of the Screen Capture window. To the right is the  *Antialiasing* check box. Use it to make the sharp edges smoother, thus avoiding possible aliasing effects on the output.

**NOTE:** It is NOT possible to capture overlay video in this object.


## Chat Line


Chat line objects appear in the work area as the one below.



Chat lines, like Chat notes, are developed to display text messages, coming from SMS service applications. Chat line objects should be linked to text files, where third-party applications save the newly-received messages. Unlike chat notes, the messages in chat lines are moving constantly, even if there are no new messages arriving in the linked text file. When a new message appears there, it will be displayed during the next queue rotation.

 The Chat line toolbar contains the following settings:

 - use this button to format the chat line font.

 - paragraph formatting allows you to change the alignment, spacing, word wrap, and other properties of the currently selected paragraph.

You can change the speed of the text through the *Speed* spin-box: the higher value, the faster will move the text. If you want to change the direction, type a minus [-] in front the speed value.

The *Queue Length* spin-box is used to define the number of messages to be displayed in the chat line. By default, it is three '3', which means that in the chat line you can see only three messages at a time. When a new message appears in the linked file, it will be displayed at first place in the chat-line object. The oldest message will disappear, so the number of the messages in the chat-line will remain three (3).

## Chat Roll

The function of this object is the same, as it is with the Chat Line and Chat Notes (displaying messages coming from third-party applications to a linked text file). The only difference is that it displays the saved text as Roll ticker.

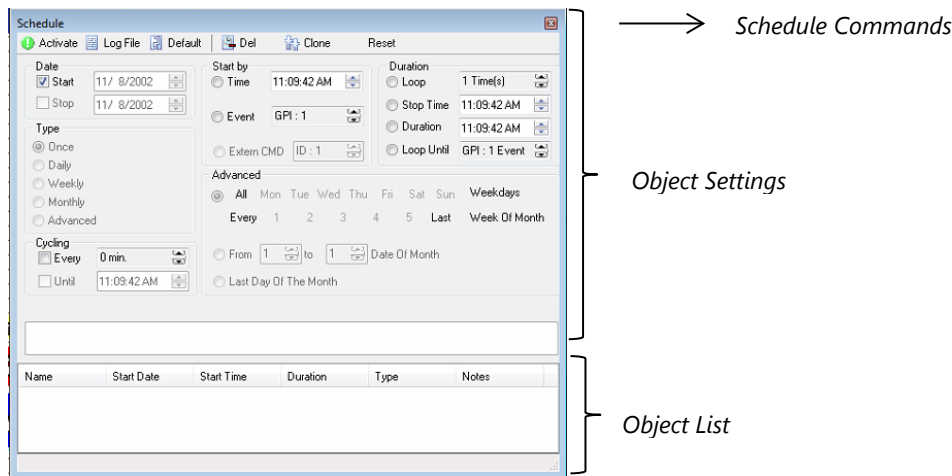


To create a Chat roll object, Press the  button in the object palette. Then, right-click on the object to set the File link.


The object-specific toolbar is identical to the Chat lines. Please, check the Chat line section above for details.



## SCHEDULER



The **Schedule mode** allows you to create schedules for playing the objects in your project.

To add an object to the **Schedule**, press the **Add to Schedule**  button from the **TitleBox Toolbar**. Note that this button appears on the toolbar after you have selected an object from the **Work Area**. The **Schedule** window will open. It consists of *Schedule commands*, *Object settings*, and an *Object list*.

## Scheduler Commands

**Activate**– press it to activate the schedule for the current project.

**Log File** – press it, if you want to create a log file. A browse window will open for defining a log file.

**Default**– sets the default settings. You can create your own default settings as well.

**Del**– deletes a selected object from the *Object list*.

**Clone**– “clones” the settings of the selected object. A new line for the same object will appear in the *Object list*. Then you can modify its settings. This feature is useful, when you want to define a different behavior of the object in the different days or hours, for example.

**Reset**– resets the Schedule.

## Object Settings

The available settings for scheduling objects are situated here, in the main part of the **Schedule** dialog.

You should select an object from the list below to make the settings for it.

*Date* – defines the **Start** and/or **Stop** date. By default, the **Start** day is today. The **Stop** date is not mandatory.

*Start by* – defines the starting trigger. It could be a specific time, an event (for example *GPI*) or *CMD*.

*Duration* – defines the duration of the object appearance through

*Loop* – the number of loops. This field is active only for dynamic objects.

*Stop Time* – the stop time within the relevant day.



*Duration* – duration within the relevant day.

**NOTE:** This duration is different than the object's duration, set in the project. You have to be very careful when you define different scheduling and object duration.

If the object duration is shorter, the object will disappear after its end, even if its schedule duration is not finished.

If the scheduling duration is shorter, the object will disappear after its end, even its own duration is longer.

*Loop Until* – defines a stop event for the object's occurrence in the output (e.g., GPI trigger).

*Type*– defines the frequency of the object occurrence in the output – **Once**, **Every day**, **Every week**, **Every month**, or in a more sophisticated pattern, **Advanced**.

*Cycling* – defines the period for repetition of the object occurrence in the output in minutes. It is possible to define the termination of the object's occurrence within the day –  *Until* spin-box. Note that the latter spin-box is only active if you have defined a cycling period in the  *Every* spin-box.

*Advanced* – to get access to these settings, you have to click on the **Advanced** button in the *Type* field. Here you can define a specific day of the week or a date for showing the object.

## Objects List

The *Objects list* occupies the lower part of the **Scheduler** window. All the scheduled objects and their settings are displayed in it.

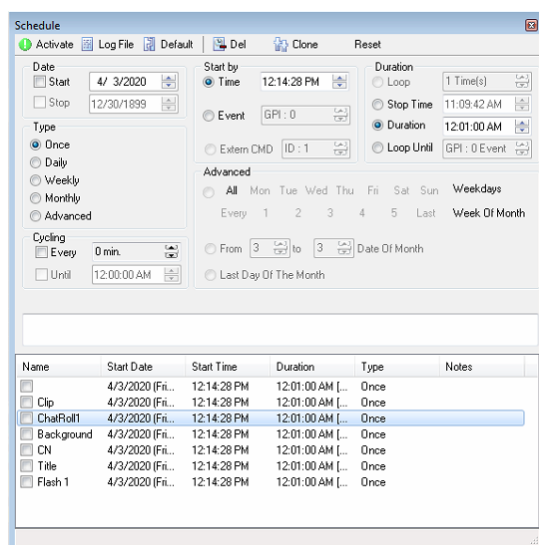
In front of each object in the list, there is a check-box. It is used for  activating/ deactivating the scheduler settings of the particular object.

Right-clicking on an item from the *Objects list* opens a context menu for copying and pasting the object settings. You can also use the following key combinations:

**<Ctrl + Alt + C>** - Copy

**<Ctrl + Alt + V>** - Paste

Please, refer to the screenshot below:






As it is visible in the *Date* field, [Picture1] starts at 10 o'clock on November 18<sup>th</sup>, and it is put on schedule until December, 31<sup>st</sup>. Furthermore, in the *Duration* field it is set to be displayed for 15 minutes every hour. Finally, [Picture1] is also being displayed on Tuesdays and Saturdays, as set in the *Advanced* area.

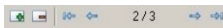
## SLIDE MANAGER

In the past, you could create your **TitleBox** projects in a single layout only. If you had to open a new layout, you had to load a new project. It was like having a single sheet of paper per project. With the **Slide Manager** your project turns into a sketchpad, allowing you to organize your projects in a multi-slide layout. Each sheet of this pad is called a **Slide**. In each **Slide** you can have numerous groups of objects, called **Layers**. Finally, in each **Layer** you can have as many objects as your system can handle.

You can now control your slides/layers/objects manually, or play the slides consecutively (or simultaneously) by using the [Slide Controller](#).

The **Slide Manager** is locked to the right of the work area. You can open and close it by using the **Slide Manager**  button, situated in the lower right corner of the **TitleBox** interface. You can also drag the **Slide Manager** out of the main **TitleBox** window.

### Slide control buttons

These buttons  are situated in the lower right corner of the **TitleBox** interface. They provide simple **slide control** options. The numbers in the middle represent information about the currently selected slide (i.e., slide 2 of 3 in the screenshot above).

The main **Slide Manager** window consists of three logical parts – control buttons at the top, slides' thumbnails and a project structure, situated in the predominant part in the middle (further divided into three tabs), and a properties area at the bottom, where you can find a detailed description of the currently selected element in the list above.


### Project control buttons




The first three buttons at the top of the Slide Manager window are used for controlling Slides. You can **Add** , **Insert** , and **Delete**  slides by using them.

**NOTE:** The difference between **Adding** slides and **Inserting** slides is that when you add a slide, it appears at the bottom of the slides list, while when you insert a slide, it appears above the currently selected slide.


The following two buttons are used to control Layers. You can **Add**  or **Delete**  layers.

While in the **Details** tab, you can show/hide slides' thumbnails by pushing the **Thumbnails** button .

In case you need to move a slide or an object, just drag-n-drop it to the new place where you would like it to be.

If you want to show/hide/pause a slide, a layer, or an object manually, slide the mouse pointer to the end of its row. The **Play/Stop/Pause**  buttons will appear. They are related only to the currently selected line. The elements' hierarchy will be observed, i.e., when you push **Play** for a Slide, all Layers and objects that belong to it will be played (if not disabled). If you push **Play** for a Layer, only the objects in this layer will be played. Respectively, if you push **Play** for an object, **TitleBox** will show only this object.

If a Slide is in **Stop** mode, it will not prevent you from playing its Layers and objects.

Pressing the  **Shortcut mode** button will enable the shortcuts functionality. Thus, if you have assigned some shortcuts to slides in your project, you will be able to control these slides using the relevant shortcuts. Please, check the [Preview section](#) below for details.

### Project Preview area

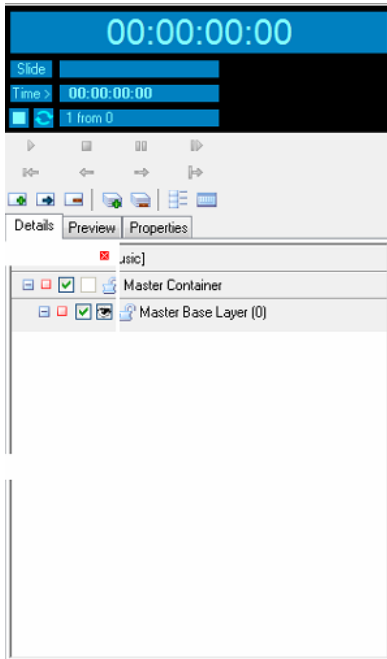
The project preview area consists of three tabs as described below.



## Details


In this tab you can view the hierarchical structure of your project. It is divided in two parts – project tree and properties.

### Project tree



The project tree part is further divided in two levels – Level 1 that contains the Project line and the Master container and Level 2 that contains all “regular” slides. Right-clicking anywhere in the project tree invokes a context menu, which is described in the dedicated [section](#).

#### Level 1

The **Project line** is situated on top. It displays the filename of the current project. The **line control buttons**  here have a slightly different functionality as compared to the other lines below. These buttons control the Slide Controller that will play all slides in the project one after the other (see the [Slide control](#) section for details). Under the project line, you will find a list of all unassigned objects (if any). These objects are present in the object but do not belong to any of its slides. Click on the plus sign in the beginning of the project line to view them. You might need to show/hide such objects manually, without affecting the rest of the project.

The **Master Container** line is situated right under the project line. In it, you can create layers to be assigned later to the “regular” slides below. By using these master layers, you will avoid inserting the same objects in numerous slides.

*For example, if you need a logo shown in Slides 1, 3, 7 below, just create a Layer in the Master container. Then, go to Slide1, click (left mouse button click) the **Assign Layer** button and select the logo-containing layer from the drop-down list. Repeat the procedure for Slides 3 and 7. Thus, each time **TitleBox** displays Slide 1, 3 or 7, it will also display the logo-containing layer from the Master Container.*

Another way to do this is just drag-n-drop the Layer from the Mater Container to the relevant slide.


#### Level 2

All slides are listed in this part of the **Slide Manager** window. You can create as many Slides and Layers as your computer can handle. You **MUST** have at least one Layer per slide.


Each element is displayed in a separate line. Each line starts with a plus sign. You can click in it to expand the relevant level.




Next to the plus sign, you can see a play status notification. Further on there are several controls that vary within the different levels. You can find a detailed description of each level below.

 In a **Slide line**, you will see Enable/Disable check box, **Assign Layer** button, **Lock/Unlock** button (*it is under development now*), and the Slide name (check the [Slide Line Properties](#) section below on how to change Slide names).

If you have assigned some layer(s) from the Master Container to a slide, their number will be reflected in the Assign Layer button. Thus, if you have assigned one layer, a small dark-blue square will appear in the button. If you assign two layers to the slide, there will be two dark-blue squares in the Assign Layer button, and so on.

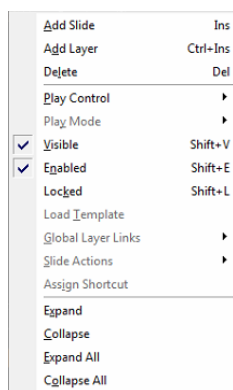
 In a **Layer line** there is no **Assign Layer** button as this contradicts the hierarchy. However, there is a **Visible/Invisible** button. This button will hide the currently selected layer from the work area, so its objects will not impede you while creating another layer in the same slide. After the Layer name, there is always a figure in brackets. It represents the number of objects contained in this layer.

 In an **Object line**, you can see the status of the object (play / pause / stop), its visibility status, an icon, distinguishing its type, and the name of the object.

**NOTE:** The object visibility concerns only the status of the object within the project and it does NOT affect the output.

In case you need to move a slide or an object, just drag-n-drop it to the new place, where you would like it to be.

## Right-Click



The first three commands are the same as the [project control buttons](#):

**Add Slide** – use this command to add a slide at the bottom of the project tree

**Add Layer** – use this command to add a new layer at the bottom of the currently selected slide

**Delete** – use this command to remove the currently selected item from the project tree

The **Play Control** menu allows you to **Play**, **Stop** or **Pause** the currently selected item.

Use the **Play Mode** menu to add a play mode to the currently selected slide. Play modes are described in the [Properties](#) section.

You can also enable/disable the following properties:

**Visible** – this option is only applicable for layers, it shows/hides the currently selected layer.

**Enabled** – use it to enable/disable the slide/layer.

**Locked** – use it to lock/unlock the currently selected slide/layer for editing.



**Load Template** – use this command to load an already created **TitleBox** template in the currently selected slide. Select your template from the browser that appears.

Use the **Master Base Layer** command from the **Global Layer Links** menu to link the currently selected slide to the master layer. In this way the slide will use the design of the master layer as a base.

Use the **Slide Actions** menu to assign an action to the currently selected slide:

**None** – no action will be assigned

**Stop Playing** – the slideshow will stop playing when it reaches this slide

**Jump To...** – select an existing slide. Whenever the slideshow reaches this slide, it will jump to the slide you select.

**Wait Key Press** – the slide will not be executed until you press a key.

If you select the **Assign Shortcut** command the following dialog will be invoked:



In the *Shortcut* field enter a key combination by pressing the desired buttons on the keyboard.

In the *Command* field select the desired command to be executed by the shortcut – **Toggle Play/Stop** or **Play From Slide**.

The last four commands relate to the appearance of the project tree:

**Expand** – expands the currently selected slide/layer

**Collapse** – collapses the currently selected slide/layer

**Expand All** – expands the whole project tree

**Collapse All** – collapses the whole project tree to level one

## Properties

In the area under the project tree, you can see the property's window for the currently selected line (project, slide, layer, or object).

- The properties of the **Project Line** contain some default settings that will affect all newly created slides:

Project [Music]	
Property	Value
DefaultColor	
DefaultSlideD...	3
DefaultSlidePl..	pmCrossPlay
Loop	0

**Default color**– this color will be applied in the preview thumbnails of your slides. You might need it in case there are some white characters in a text object that would be impossible to see on a white background.





**Default slide duration**– the time for showing the slides when playing them with the Slide Controller;

**Default Play mode**– the way of showing the slide on the screen:

*Cross Play*– if the previous slide has an out transition, the following slide will appear as soon as the out transition starts.

*Stop Previous*– if the previous slide has an out transition, the following slide will “wait” for this transition to end and only then will appear on the screen.

*Add* – the previous Slide will remain on the screen and the following will be overlaid on top of it. Currently, there is a known limitation, related to this type of play mode – once you pile up a slide on top of another one, the first slide will remain on the screen, even if the second one is stopped. Thus, if you want to stop the first slide, you will have to do this manually.

*Clear Previous* – the previous slide will be removed immediately off the screen, its out transitions (if any) will not be executed.

**Loop** – specifies how many times the Slide Controller should play all slides in the project. The default value [zero] means endless loop – the Slide Controller will play all slides one by one repeatedly.

- If a **Slide Line** is selected, you will see the following properties:

Slide 1	
Property	Value
BackColor	
Duration	3
Enabled	True
Lock	False
PlayMode	pmCrossPlay
ViewName	Slide 1

**Duration** – this is the time (in seconds) for showing the selected slide with the Slide Controller. If you decrease this value to zero, the duration will be set to [Auto]. Auto duration is applicable when there are objects with different duration in the slide. In such a case, the Slide Controller will show the slide until the longest duration is through, then will go to the next slide in the list.

**TIP (!)** Use it for objects like crawl/roll and animation when they are looped.

**Enabled** – it is equal to the  *Enable* check box in the slide’s row.

**Lock** – reflects the padlock status in the slide’s row.

**Play mode**– how **TitleBox** should treat the previous slide on starting the currently selected one. Please, check the project properties above for details.

**View Name**– click in this line to change the slide name at your will.

- If a **Layer Line** is selected, you will see:

**Enabled** – shows the status of the  *Enable* check box of the currently selected layer.

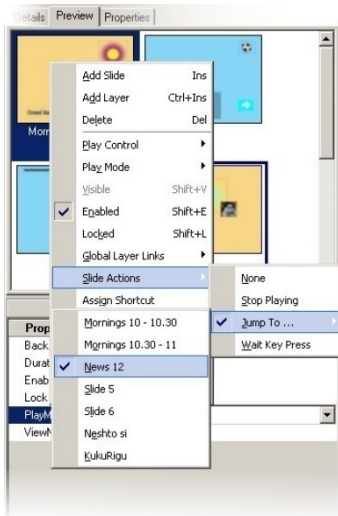
**Lock** – reflects the padlock status in the layer’s row

**Visible** – shows the status of the Visible/Invisible button.

- When an **object** is selected, you will see below all its properties that can be adjusted in the main **TitleBox** interface. These properties are different for the different objects and have already been described in the [Object Properties](#) section.



## Preview



In this tab you can see thumbnails of all Slides in the current project.

Double-clicking in a slide's thumbnail will load it in the work area for editing. If you select an object in the work area, its properties will be displayed in the Details area below.


Right-clicking in a slide will open a context menu. It contains basic slide controls.

The **Play Mode** represents the way of showing the slide on the screen in relation to the previously played slide. Please, check the project properties section above for further details.


Sliding the mouse over the **Global Layer Links** line will open a drop-down list of all Layers available in the Master container.

The **Slide Actions** list contains four commands related to the Slide Controller. They will be executed when the duration of the current slide is over.

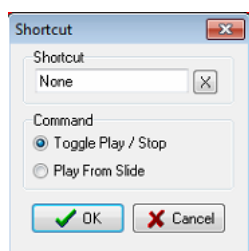
None is the default slide action, no command will be sent to the Slide Controller when the slide duration is over.

Stop Playing – when the duration of the currently selected slide is over, it will send a stop command to the Slide Controller. A small sign  will appear in the lower right corner of the slide thumbnail to notify the operator about the assigned action.

A Jump To... action will make the Slide Controller loop between the current slide and the specified Jump To... slide.

Wait Key Press is a pause command sent to the Slide Controller. Playback of slides will resume at any key stroke. The following sign  will appear in the thumbnail to notify the operator.

Selecting **Assign Shortcut** will open a shortcut-defining dialog. To define a shortcut, click in the *Shortcut* string and press the keys you would like to assign.



Below, select the command to be executed at pressing these keys:

⊙ **Toggle Play/Stop** – to show/hide the slide off the screen;



Ⓞ **Play From Slide** – to start displaying all following slides in a row (via the Slide Controller).

To clear the shortcut, press the **X** button to the right of it.

All assigned shortcuts will appear in the slide's thumbnail:




The blue background means the assigned command is Play From.



The green background means the assigned command is Toggle Play/Stop.



The red background means that there is an assigned shortcut to this slide but the Shortcut mode button is not pressed.

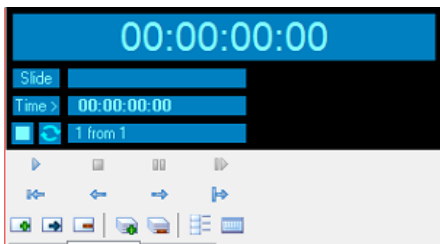
The assigned shortcuts will be active only when the  **Shortcut mode** button is pushed.

**NOTE:** In shortcut mode, the newly assigned shortcuts have higher priority than the default ones. Thus, while in shortcut mode, the default shortcuts in **TitleBox** will operate normally unless duplicated with newly assigned shortcuts. For example, the default command for **<Ctrl + H>** is Add to schedule, but if you assign **<Ctrl> + <H>** to be Toggle Play/Stop of a slide, the shortcut will act as Toggle Play/Stop.

## Properties

In this tab, you can see the detailed properties of the line that is currently selected in the Details tab.

## Slide Controller



The slide controller will play your slides consecutively, with duration at your will. If you want to rearrange the order of showing the slides, you will have to drag-n-drop them in the **Details** tab. If you want to skip a slide, just disable it by un-checking the  **Enable** checkbox in its row. .

The Slide Controller window contains four rows and eight buttons. The Controller counter is situated in the top row. It counts the time since the slide controller was started.

Below, you can see the name of the currently playing slide.

Further down is the Slide counter. By default, it shows the elapsed time since the start of the current slide. If you want to see how much time remains to the end of the slide duration, click in the **Time>** cell. It will turn into **<Time** and will start counting down.

The last row has three parts – play status, loop value and the number of the currently playing slide.

At the bottom of the Slide Controller are situated the playout control buttons. They provide hints and hotkeys:




Button	Command	Shortcut
	Play	<b>F5</b>
	Stop	<b>F6</b>
	Pause	<b>F7</b>
	Resume	<b>F7</b>
	Go to First Slide	<b>Ctrl + Home</b>
	Go to Previous	<b>Ctrl + PageUp</b>
	Go to Next	<b>Ctrl + PageDown</b>
	<b>Skip Next Slide</b>	<b>Ctrl + Left arrow</b>

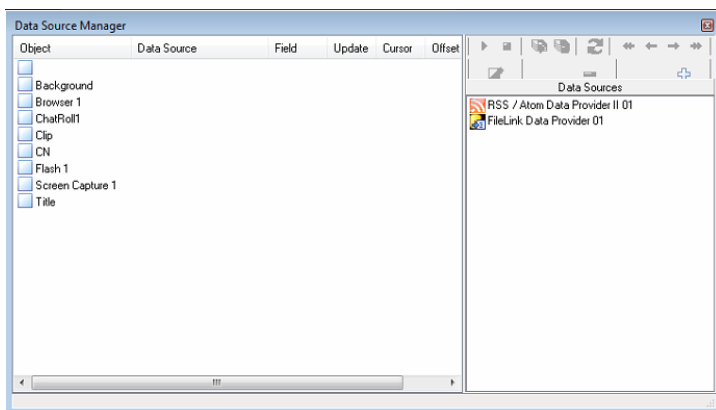
By default, the Slide Controller is a separate window that appears outside the Slide Manager. If you want to lock it to the Slide Manager window, right-click in the counter and select Dock in Manager.

## DATA SOURCE MANAGER


The Data Source Manager allows users connect to different data providers through the selected plug-ins.

This engine allows connecting different properties of a **TitleBox** object to different data sources or providers, like RSS channels, \*.html files, \*.xml files or third party data bases. Besides, the same data source(s) can be related to different objects in **TitleBox**.

The dialog below opens after pressing the Data Source Manager  Button in the main **TitleBox** interface.



The left part of this dialog contains a list of all objects in the current project and already assigned to them data source instances.

To add a new data source instance, press the **Add**  button.



The list of currently available data provider plug-ins will open. In it, you can choose the data source to be used as a new instance. Select its line and press **Config**. The configuration dialogs vary according to the selected plug-in.

You can read more detailed description of each plug-in definition further in that manual.

When you configure the plug-in, and press **OK** button, into the main screen a new data provider appears in the list.

Select the needed data provider and drag and drop it over the **TitleBox** object in the left.

Then a **DataDistributor Properties** window will open. Here you have to define the way of data source appearing.

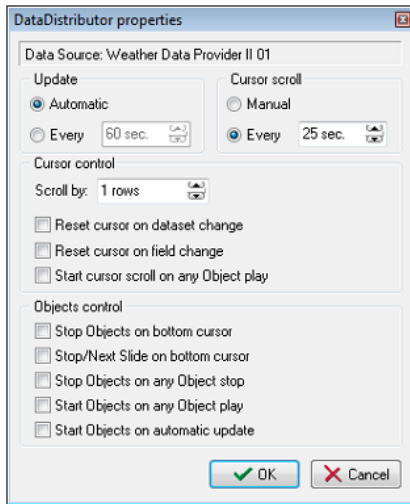
Into the header line of the Property window is visible the name of the selected data provider.

Into **Update** area you have to specify how often to update the source information. Select **Automatically**, to update the data automatically on source change. Automatically option is available only for EAS data provider and File link data provider.

Select **Every xx sec**, to update the data periodically at every xx sec. By default, the update is at every 60 seconds.

The term "**cursor**" further in that section, is used to define the cursor position into the data source.

Into **Cursor scroll** area, you can specify how to move the cursor to the next row, if the data source has many rows: Manually, or at Every xx seconds. If you select Manually, you can start the data provider manually from main data provider menu



**NOTE:** The cursor continue cycling, even the connected object is not played on air. This means that when you start again the object, it will show data from current cursor position.

**TIP (!)** If you want at every object start to show the first line from the data source, leave **Cursor scroll** to be Manual and check **Start cursor scroll on any Object Play**.

In **Cursor control** area, there are the following settings:

**Scroll by X rows**, where **X** is a number of rows to scroll the cursor. If **X** =1, the cursor will scroll each row. If **X** =2, the cursor will scroll each second row (i.e. the cursor will skip one row), etc.



**Reset cursor on dataset change** – check this, if you want to move the cursor to the initial position, when the data in [Master column](#) of data provider is changed.

**Reset cursor on field change**- check this, if you want to move the cursor to the initial position, if any field into data provider changes.

**Start cursor scroll on any Object Play** – check this, to start the cursor scrolling from the first position in data source, any time, when the object is started.

In **Objects control** area there are the next settings:

**Stop Objects on bottom cursor** – check it, if you want to stop the object(s) connected to the data provider, when the cursor is at the end of the data in data source.

**Stop/Next Slide on bottom cursor** - check it, if you want to stop the object and go to the next Slide, when the cursor is at the end of the data in data source.

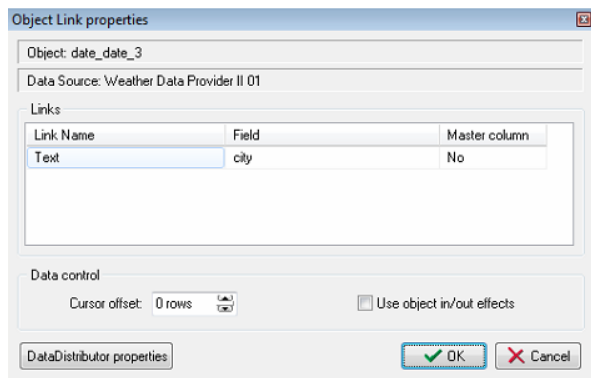
**Stop Objects on any Object stop** – check this, to stop all objects connected to the same data provider, if any of these objects is stopped. It is useful, when the project is control remotely.

**Start Objects on any Object play** check this, to start all objects connected to the same data provider, if any of them is started.

**Start Objects on automatic update** - check this, to start all objects connected to the same data provider, on automatic update of the data source. It is relevant, if Automatic update is checked. Automatically is available only for EAS data provider and File link data provider

Press **OK** button to save **Source Link** settings.

After that, an **Object Link Properties** window will appear. Here you have to define how to link the data from the data source to the **TitleBox** object.



In the header of the window, you can see the name of the **TitleBox** object to which data provider is assigned and the name of the data provider.

Next is a **Links** area, where are three main columns:

**Link Name** – this is the type of the selected object (*Text, Sound, etc.*).

**Field** – here you can see the list of all available fields from the data-source. Select from drop-down list one of them, which you want to show into the **TitleBox** object.

**NOTE:** You can assign a data-source field to a **TitleBox** object, only if they are from the same type – text, sound, etc.

**Master column** – you can select one of the data provider's columns to be a **Master** column. Enter YES in Master column field, if the selected field is from this master column. Set a column to be a **Master**, if you want to reset the cursor for all columns, on data change in this particular column,

In **Data control** area you can set a **Cursor offset** – this is the offset upon the cursor for displaying the data into this particular **TitleBox** object.



## Example:


Let us show the data from data source in a table 3x3 (3 rows and 3 column) in *TitleBox*. You have to draw 9 objects in *TitleBox* project, each connected to a separate field from data provider. Into Data Source Link Properties->Cursor control->Scroll by, enter 3(scroll should be equal to the numbers of table rows). Into Object Link Properties ->Cursor offset, cursor offset should be 0 for the first row; 1 for the second row and 2 for the third row.

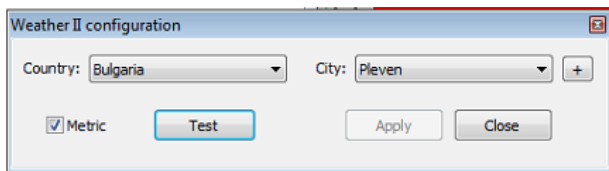
If you want to use in/out effects, you have to check the relative check-box.

Press the OK button to save the settings.


## Weather Data Provider

.This data provider displays information coming from weather forecast sources. For now, the only available provider is **YAHOO!** weather. In this way online weather information will be shown in the objects.

To adjust the *Weather Data Provider*, click the  button in the main *TitleBox* interface. The *Data Source Manager* dialog will open. The left part of it contains a list of all objects in the current project. To add a new *Weather Data Provider* instance, press the **ADD** button. A list of all currently available data source plug-ins will open. Select the *Weather Data Provider* and after that press the **ADD** button. A dialog will appear for specifying a station:

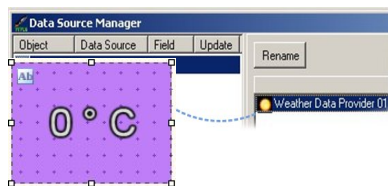


Use the *Country* and *City* drop-down lists to pick your desired location. Keep the  *Metric* box checked if you want the temperature to be displayed in Celsius. Uncheck it for Fahrenheit. Press the **Test** button to see if the connection to the station is successful. Press **Apply** to save your configurations and then press **Close**.

After pressing **OK**, a *Weather Data provider* instance will appear in the list to the right part of the *Data Source Manager* dialog. For displaying information, using this instance, select the  *Weather Data Provider* line and drag it to the object desired.

When you release the mouse button, a fine-tune dialog will open so you can select what kind of information will be shown in the object (as Temperature, Postal Code, Current city humidity, etc.).

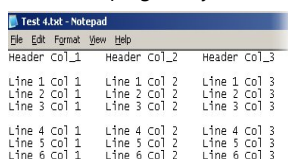
This plug-in will display the weather information according to the settings you have made. *For example*, if you intend to display information about *Sofia* town and you have chosen *Temperature C* from the *Field* drop-down list, the *Weather Data Provider* will show the current temperature within the object



(have a look at the example to the right).

## FileLink Data Provider

Select this plug-in if you intend to link certain objects in *TitleBox* to text documents.



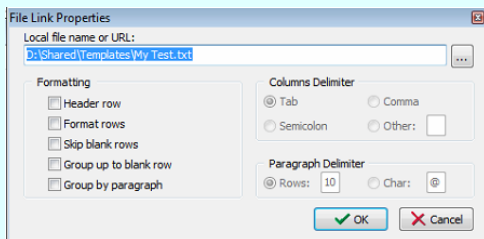


Earlier in this Manual it was described a simple [file link](#). It is assigned to a text object and shows the whole text document, as it is.

Unlike *simple File Link*, the *FileLink Data Provider* allows the user to modify how the text from the text document will be displayed on screen.

Example:

By using *FileLink Data Provider*, one text document could be interpreted as a table. It is applicable for texts, where the data is organized in columns and rows (like in the example Test4.txt file in the screenshot below). The user can define in what kind of order to display the rows and columns.



Select the *File Link Data Provider* and press *Config*. Browse for the file you need, select it and press *Open*.

The dialog above will appear, so you could define Formatting of the linked file.

Check the  *Header row* check box, if the first row of the linked document is a column header and should not be visible on the screen.

Check the  *Format rows* check box if the file consists column delimiters (like coma, semicolon, etc.). Checking this check – box will activate Columns Delimiter radio-buttons.

Check the  *Skip Blank rows* check box, when there are empty rows in the linked file; they will be skipped on data scrolling in the text object.

Check the  *Group up to blank row* check box, if you want to show into the text object, a group of data separated by empty row.

Check the  *Group by paragraph* check box, if you want to show into the text object, a group of data separated by some Paragraph Delimiter. Checking this check – box will activate Paragraph Delimiter radio-buttons.

*Columns Delimiter* is a symbol, which is used for separating the data by columns in the text file. You can choose between Tabulation (Tab), Comma (,) and Semicolon (;) as delimiter, or you can define your own delimiting symbol in field *Other*. In our example, the column separator is Tab.

*Paragraph Delimiter* is a symbol, which is used for separating the data by paragraphs in the text file. You can select a certain number of rows as paragraph separation or you can define a symbol, which to be used as paragraph separator. Enter the symbol in Char field.

If you want to separate the source data by paragraphs, but there are not delimiter symbols inside your file or the numbers of rows per each paragraph is different, then you can use the PlayBox Delimited Inserter.

**PlayBox Delimited Inserter** (TextConvert.exe) is an additional application, which helps the users to separate the source data file at different paragraphs. It is installed through the main **TitleBox** installation. You can start the application from **TitleBox** main folder (by default: C:\Program Files\PlayBox Technology Ltd\TitleBox).



Into the **Source** field, browse the location and name of the source file.

Into *Destination* field browse the location and name of the resulting file.

Select the type of the delimiter symbol in *Delimiter* field.

It is possible to have alternating paragraphs with different length (number of rows). Then you can select the number of rows for the first paragraph in TitleRows field and the number of fields for the second paragraph in Descr. Rows.





Example:

Let the source file contains news data, organized into two paragraphs: in the first one there are the titles of the news and in the second one, there are the full texts of the news. Then the "Title Rows" number is different than the "Description" rows number.

When you are ready, press the Save And Apply button. The new file will be created in destination folder. Depending on status of this process, the next three squares will flash:

First square is Running status : Red=Stop; Green=Running;

Second square is Error status: Red=Error; Green=OK;

Third square is Converting status: Yellow=Converting; Green=Waiting for file change.

If the source file is updated periodically, keep the Delimiter inserter opened. It will create a new file at each saving of the source file. Converting square will show the status.

Back into the Data Source Manager window; drag the FileLink data source from the right to a text object in the left.

When you release the mouse button, a link fine-tune dialog will open so you could select which column from the file should be linked to that text object.

**NOTE:** Currently with File Link Data Provider, you can link your text object only to \*.txt files!


## ODBC Data Provider

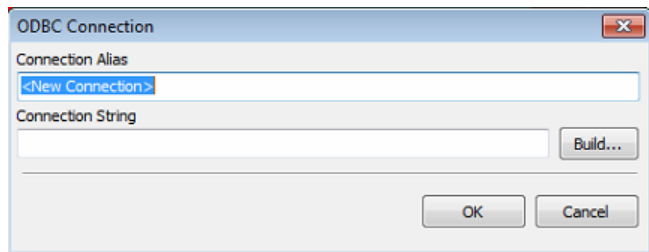
This plug-in allows connecting to ODBC-compatible database formats and displaying the information they contain on the screen.

Choose the ODBC Data Provider in the list of available plug-ins and press **Config**.

The ODBC Connection Manager will open.



Press  to create a new connection. The following dialog will open for you to specify the Connection Alias and view the Connection String.



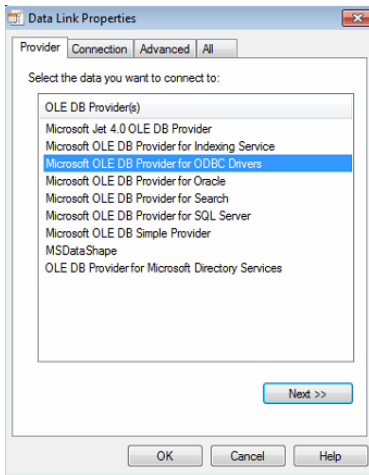
Press **Build** to configure the connection. The window below will open:

The ODBC configuration dialog contains four tabs:

In the **Provider** tab, select the data base type you want to connect to.

When you click **Next**, the **Connection** tab opens, so you could configure the connection to the selected database.

The **Connection** tab looks differently depending on the Provider you have chosen. When you are done entering the connection details, presses **Test Connection** to check if it works correctly.



After configuring the connection, you can open the **Advanced** tab for more configuration options, which differ depending on the selected Provider.

Finally, you could view all settings for the current connection in the **All** tab. Click **OK** and **Close** in all open windows until you return to the Data Source Manager. In it, double-click on the newly-created ODBC Data Provider instance. A fine-tune dialog will open for you to configure the data Query. Press **Execute Command** to check if the connection is working correctly.

Press **Apply** and **Close**.

To assign the ODBC Data Provider instance to an object, drag it from the list in the right to the relevant object in the left. Close the Data Source Manager window.

## XML Data Provider

Choosing this plug-in allows you to insert data into **TitleBox** object from an \*.xml file.

Choose it from the list of available plug-ins and press **Add** button. The following window will open:

Into **XML** field, enter the location of the \*.xml file.

Into **XSL** field enter the location of the related \*.xsl (\*.xslt) file.

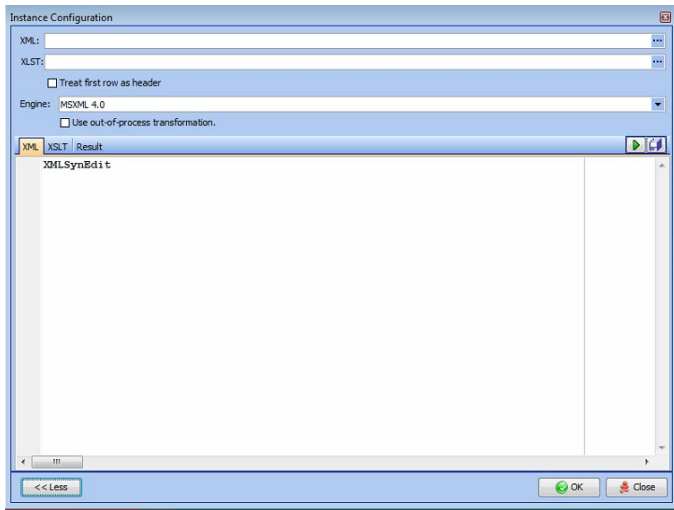
An XSL (eXtensible Stylesheet Language) file is needed for defining XML document transformation and presentation. Since the XML language does not use predefined tags, it is necessary to provide the application with information on how to interpret the XML document.

In **TitleBox**, the XSL file is needed to transform the XML file to coma separated text table.

**NOTE:** Currently, under Microsoft Windows 7 you can use only Msxml4.0 (with update msxml4-KB973685). There is no update from Microsoft Windows 7 for Msxml6.0.

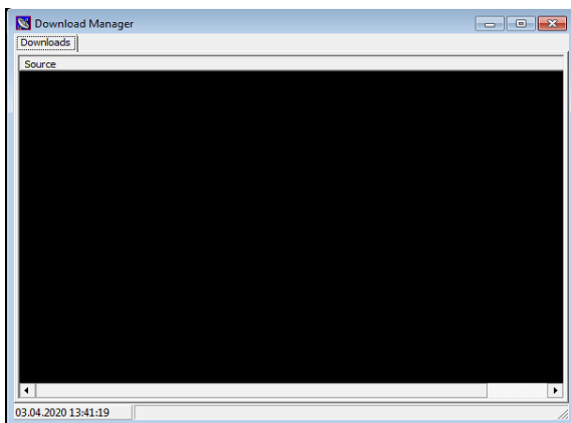
If you have a header line, check the check-box **Treat first row as header**.

Press **More** button to see the source \*.xml file, the related \*.xls file and the result of the transformation. In order to see the **Result**, first press **play** button.



**NOTE:** If \*.xml files used with XML data provider are web based files, it is needed to download these files locally on your PC first. For downloading, you can use a dedicated application (*Downloader.exe*) included in **TitleBox** installation.

You can find *Downloader.exe* in **TitleBox** folder. Start the application and the following window will open:



To add a new source for downloading, right mouse click on the black area and the set-up dialog will appear. Here you have to enter the URL of the source and the location for saving the file. If access to the source needs some Username and Password, enter them in the relevant fields.

## RSS Data Providers

Choosing this plug-in allows you to connect to RSS feeds from the Internet.

RSS (Really Simple Syndication) is an XML-based format for sharing and distributing Web content, such as news headlines for example. Currently many web sites provide RSS feeds.

There are 2 types of RSS data providers in **TitleBox**. The main difference between them is that *Data Provider I* shows RSS feed line by line into the linked object, while *Data Provider II* shows all lines of RSS source together into the linked object.

### RSS/Atom Data Provider I

Choose it from the list of available plug-ins and press **Add** button. The following window will open.



In **RSS/Atom Feed** field enter the URL of the needed RSS feed.

To check the connection, press **Test Feed** button. You will receive a message if the connection is successful.

There are two advanced options available for choosing the proper behavior if the RSS feed is not reachable or if there is no data.

Check the first check-box, into **Advanced** area if the RSS feed is not reachable, but you want to see the old data in the relative **TitleBox** object;

Check the second check-box, if there is no any data in RSS feed, but you want to have some text into related **TitleBox** object (like "No data available" or "please excuse us...", etc.).

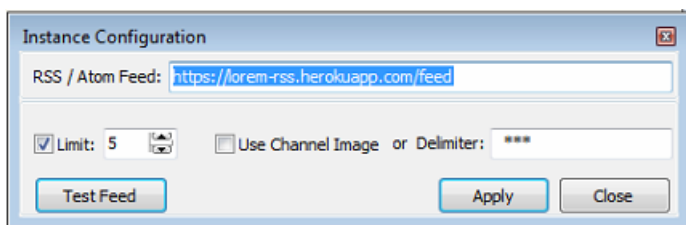
Press **Apply** button to accept the settings.

Press **Close** button to close the window without changes.

## RSS/Atom Data Provider II


As mentioned before, *Data Provider II* shows all lines of RSS source together into the linked object. The different lines are separated by some delimiter.

Choose Data Provider II from the list of available plug-ins and press **Add** button. The following window will open.



In **RSS/Atom Feed** field enter the URL of the RSS feed.

Check **Limit** check-box if you want to limit the number of displayed lines and enter the number of lines into the next field. By default, Limit check-box is checked and the number of lines is five.

Then enter the **Delimiter** sign. If the source channel provides the Channel image,  you can select **Use Channel Image** check-box. If it is checked, but there is no channel image, then the **AirBox** channel image will be shown by default. If you want to use your own picture as delimiter, you can use bitmap or movie tags into the field. Inserting of the bitmap or movie tags is the same like for text objects. See [here](#) for description.

**NOTE:** RSS data providers works with Internet Explorer only. You need Internet Explorer v.8.0 or higher.

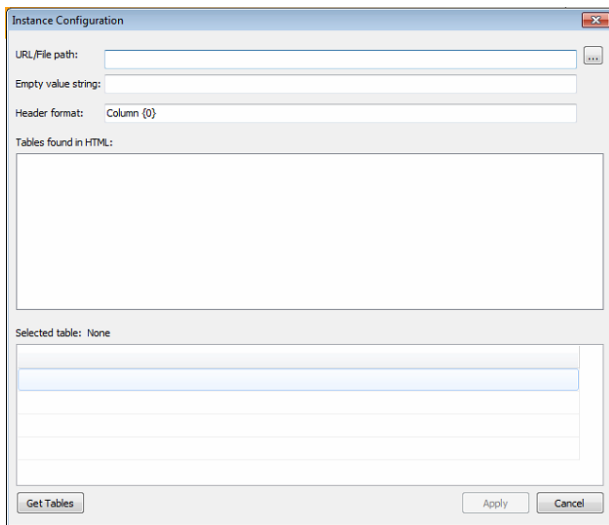
**NOTE:** It is possible to see RSS data in **TitleBox** differently ordered than it is visible in the RSS source site. It is because some RSS sources (news channels for example) give opportunity to user to order the visible into browser data differently - by date or by title, etc., while **TitleBox** data-provider extracts the data from the original \*.xml file of the RSS and there is no possibility to re-arrange their order.

**TIP (!)** You can use downloader.exe, described in [XML data provider section](#) for downloading RSS feed first. It is useful in case the link to the RSS breaks.

## HTML Table Data Provider

Choosing this plug-in allows you to insert data into **TitleBox** object from an \*.html file.

Choose it from the list of available plug-ins and press **Add** button. The following window will open.



**URL/File path** – enter here the location of your \*.html table

**Empty value string** – enter here the string, which you want to be visible if there are no data into the table.

**Header format** – this is the format of the table header. By default, it is "Column {0}"

Into **Tables found in HTML** area, you will see the list of tables found into the source \*.html file. In order to see a table, you have to select a line and press **Get Table** button in the bottom of the window.

Press **Apply** button, to accept the settings.

Press **Close** button, to close the window without saving the changes.

## EAS (Emergency Alert System) Data Provider

The **Emergency Alert System (EAS)** is a national warning system in the United States of America put into place in 1997. The EAS requires broadcasters, cable television systems, wireless cable systems, satellite digital audio radio service (SDARS) providers, and direct broadcast satellite (DBS) providers to provide the communications capability to the President to address the American public during a national emergency.

The system also may be used by state and local authorities to deliver important emergency information, such as AMBER alerts and weather information targeted to specific areas.

Each State and several territories have their own EAS plan.

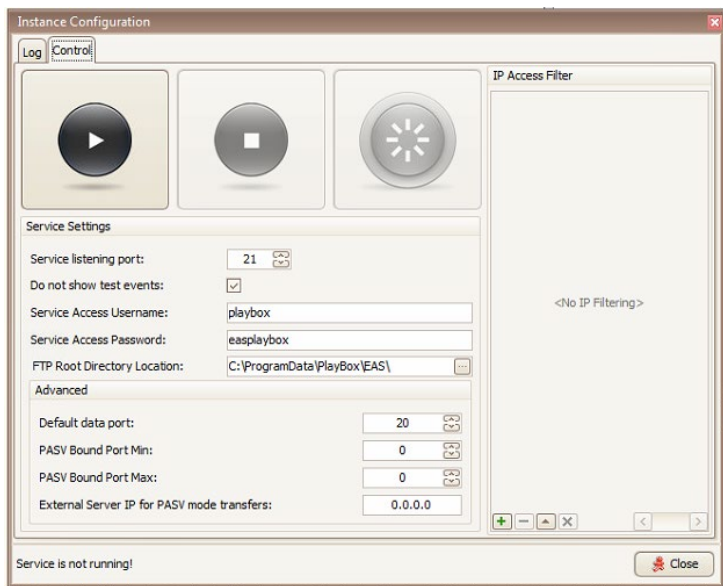
The EAS regulations and standards are governed by the Public Safety and Homeland Security Bureau of the FCC (Federal Communications Commission).

All EAS equipment must be FCC certified for use.

**TitleBox** is connected to EAS decoder unit via LAN connection. The messages coming from EAS unit are connected to **TitleBox** objects via EAS data provider.

EAS data provider is started as service. It could be assigned to a text and to an audio object.

When you select EAS data provider from a list of data providers, the following setup win window opens:



Here, you have to enter the IP address of the EAS unit, as well as some advanced details of the connection. If there is connection username and password defined, you have to enter them also.

When you are ready, press the **Play** button, in order to start the service.

**NOTE:** It is very important to set the proper time zone to the system and the system clock to be accurate, in order the service to work correctly.

**IMPORTANT:** **TitleBox** works with decoders, produced of Digital Alert Systems, LLC with FCC ID: R8VDASDEC-1EN.

## TASK MANAGER

**Task Manager** is an instrument for creating specific **tasks** in **TitleBox**.

A **task** is an action, which is executed in **TitleBox** (like play object, stop object, etc.). The **Task** usually is performed, when happens a specific condition. The condition (trigger) on which **Task** is executed is defined by user.

The condition (trigger), on which the **task** is performed, is called **event**. An **event** could be the object's status (like play, stop, etc.) or a command (like incoming GPI signal; **TitleBox** internal command, etc.).

First you have to create a **task**, and then you can assign this **task** to an **event**.

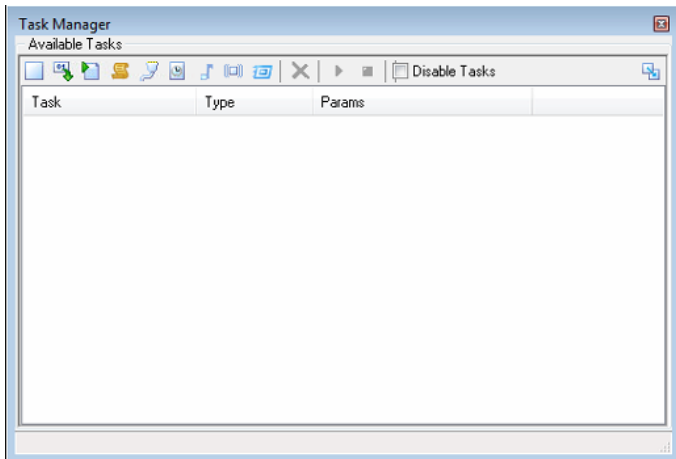
Suppose that the user wants to start *Object1* when *Object2* stops. Then the **task**, which have to be created is "*Start Object1*" and the condition, i.e. the **event** is "*OnStop Object2*".

Another example is, if the user wants to start *Object1*, when a *GPI* signal comes. Then the **task** is "*Start Object1*" and the **event** is incoming the *GPI* signal. Another example: To generate a *GPI* signal when *Object1* starts, then the **task** is "*GPI*" and the **event** is "*OnStart Object1*".

## Tasks

For creating **tasks**, you can start the **Task Manager** from **Project** menu → **Plugins** → **Task Manager**.

The window will open also if right mouse click over an object in **TitleBox** working area and select **Task Manager** from the list.



All **tasks** are visible into the **Task** list.

To create a new **task**, press the related icon from the menu bar.

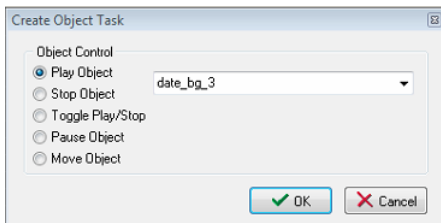
To delete a **task**, select it and press delete button from the menu bar.

The **tasks** could be executed manually (for test purposes). Press play button to start a **task** and stop button to stop a **task**.

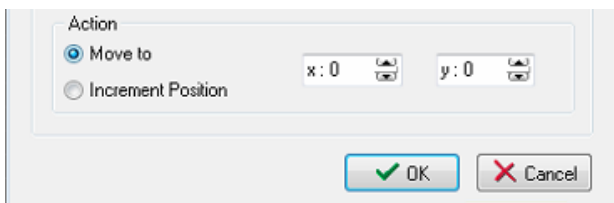
The available **tasks** are:

## Object Control

It provides options for creating object-related **tasks**: *Play Object*; *Stop Object*; *Toggle Play/Stop*; *Pause object*; *Move Object*. Select one of these options and then select an object from a drop-down list at the right of the window. In this drop-down list are listed all objects existing in the current project. Press **OK** and you will see your **task** in the **task** list.



If you select the **Move Object** control, an additional window will open where to enter the moving options.



Select **Move to** and enter **X** and **Y** values, to move the object to a position with exactly these **X** and **Y** coordinates.

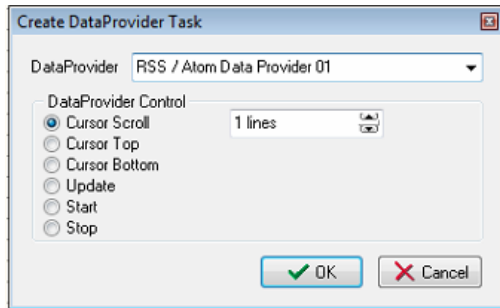
Select **Increment position** and enter **X** and **Y** values, to increase the existing coordinates of the object with these values.

Repeat the procedure till you have all necessary **tasks** related to objects.



## DataProvider Control.

Here you can create **tasks** related to data providers.



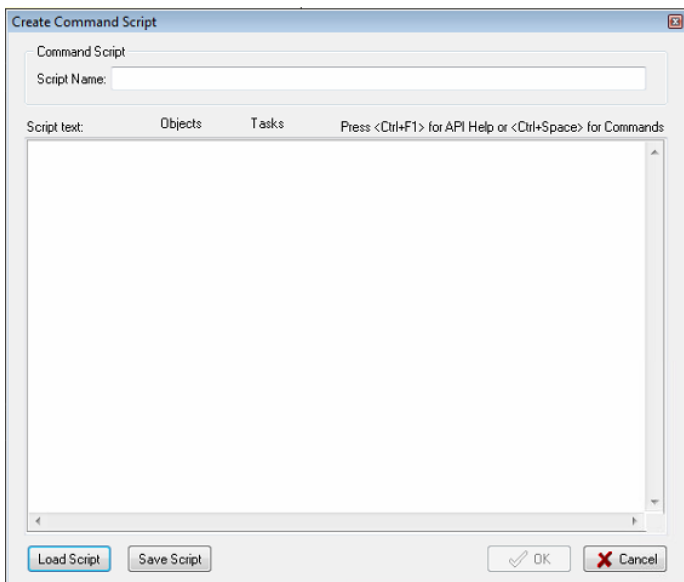
At the first field in the window, you have to select a data provider from a list of all existing data providers. Then you can choose an activity for this data provider, like **Cursor Scroll**, **Cursor Top**, **Cursor Bottom**, **Update**, **Start**, **Stop** data provider.

If there are more than one row in the data provider source (like in RSS data provider or in ODBC data provider), then you can select also **Cursor scroll** –move cursor to next row; **Cursor top** – move cursor to the first row and **Cursor bottom** – move cursor to the last row.

If **Cursor scroll** is selected, an additional field is visible. There you can enter the number of rows to scroll cursor.

## Command Script task

This command is used, if you want to create a **task** based on **TitleBox** internal commands. You can refer to the **TitleBox** API to learn more about **TitleBox** internal commands. In the Command Script dialog you can press <Ctrl+F1> to invoke the API help and <Ctrl+Space> to invoke the list of commands. To exit from the two, press <Esc>.



For example, you can create a *Script* for loading a new project in **TitleBox**:

```
"PROJECT LOAD=template1.tpl"
```





Further you can assign this **task** to any of the existing **events**.

Enter the name of the **task** in the **Script Name** field. Be aware that if you save your script with a certain name, recognized by **TitleBox**, it will be executed automatically under given conditions. Check the **Auto-executed Scripts** function, described [below](#).

Enter the **TitleBox** command into **Script text** field.

Press **OK** button to create the **task**.

If you want to save the script from created **task** in a file, press the **Save Script** button

If you want to use an existing script, press **Load Script** button and select the script file.

**IMPORTANT:** In order to use the Script task properly, you need to be familiar with the TitleBox API commands, which are listed and explained in the [TitleBox API User Manual](#). To obtain the latter, please, contact our support team at [support@playboxtechnology.com](mailto:support@playboxtechnology.com).

## Program Script Task

The **Program Script Task** is an extended feature in **TitleBox**, which not only gives you the opportunity to write program scripts for all the tasks that can be done in **TitleBox** by writing a code for them but it also allows you to make your projects even more functional and user friendly. Note that in order to take full advantage of this **TitleBox** option, you need to be an advanced user of the program and you must have at least a basic programming knowledge.

This **TitleBox** feature supports four different language syntaxes:

- Pascal script
- C++ script
- Visual Basic script
- Java script

**TIP (!):** Considering the **TitleBox** functionality, it is advisable that you use **Pascal** or **C++** script, instead of **Visual Basic** or **Java** script. Since you always have to declare the type of a variable in **Pascal** and **C++**, the latter are more thorough in terms of avoiding mistakes. In **Visual Basic** and **Java** script, on the other hand, new variables are not defined in terms of their type. They simply adopt the variant type and can accept all kinds of variables, no matter if they are string, integer, etc., throughout your script, which inevitably becomes a prerequisite for errors.

Please, be aware that **TitleBox's Program Script Task** does NOT support all the functionalities that each of the programming languages, listed above offers. All the **classes**, **functions**, **types**, and **variables** that this interface supports, are listed in the tree view, situated in the right area of the **Create Program Script** window.

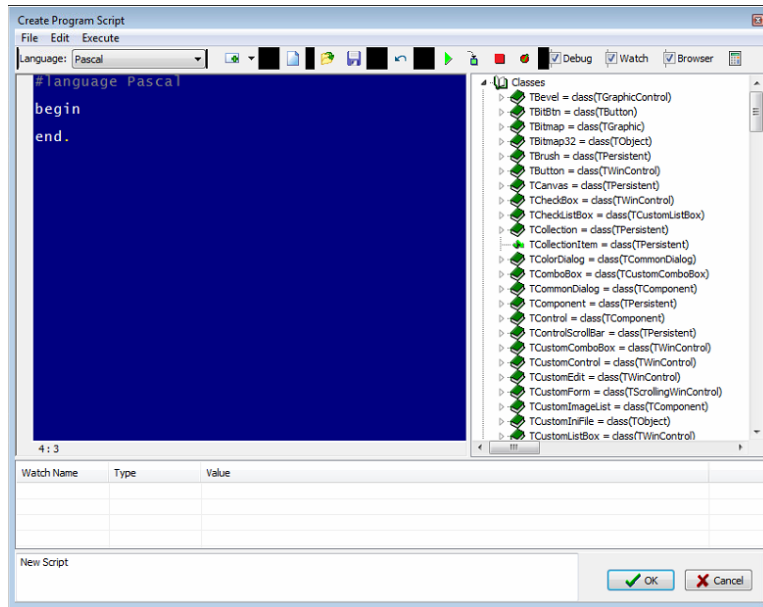
**IMPORTANT! PlayBox Technology**, does NOT offer support for third party scripts, developed with the **Program Script Task**. This function should be used at the sole responsibility of the corresponding third party developer.



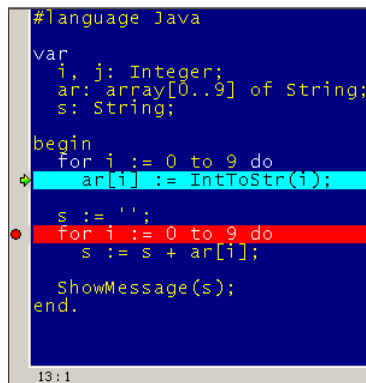
## User Interface

### Work Area

Pressing the **Program Script**  button in the **Task Manager** window invokes the following user interface:



### 1. Program View



This is the area, marked with number '1' in the picture above. This is the place, where the user enters their script. Depending on the programming language, selected from the *Language* drop-down menu above it, the script will automatically load a language identifier in the beginning of the script window, which is in the following format: [#language Pascal/C++/Basic/Java], depending on the language selected.

Note that the grey area to the left shows certain events, added manually in the script, like **Execute next step**, and **Toggle breakpoint**, as explained in the [Toolbar](#) section below.

Also, the grey area at the bottom of the program script view shows the position of the cursor, [13:1] in this example. This information is useful when debugging, since the errors are indicated in accordance to these position coordinates in the *Debug* view.







## 2. Browser View

This is the area, marked with number '2' in the picture above. This view shows you a list of all the [Classes], [Functions], [Types], and [Variables], available for you to include in your program script. You can expand/shrink the list of each of the above items by pressing the plus/minus sign next to it, or by double-clicking on them.



The Tree View supports **drag-n-drop** and **double-click** functionalities, as explained below:



### – Drag-n-Drop

If you drag-n-drop an item, designated with  or  in the program script, only the class type will be implemented in the script. For example, if you drag-n-drop the [TWinControl = class(TControl)] to the script, only [TWinControl] will be implemented in the script view.

If you drag-n-drop an item, designated with  or  in the program script, the whole line will be implemented in the script. Thus, if you drag-n-drop the [property Cursor: Integer], the whole [property Cursor: Integer] line will be placed in the script view.

### – Double-Click

Besides drag-n-dropping, you can also double-click on the items from the Tree View. If you double-click on a  or a , the list will correspondingly expand or shrink.

If you double-click on an item, designated with  or  on the other hand, it will appear in your script as a name and brackets next to it with the appropriate number of places for you to fill in, separated by commas. For example, if you double-click on the function [DirectoryList(const Path: String; List: TStrings)], the following will be implemented in the program script: [DirectoryList(,)].

The *Browser* area is only visible while the  *Browser* box in the toolbar is checked.

## 3. Watch View

This is the area, marked with number '3' in the picture above. It allows you to have an additional control over your program via "watching" how your variables behave in your script.

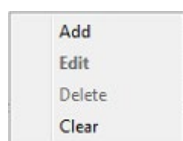
The watch area consists of three columns:

**Watch Name** – this column shows the name of the item/variable that is being watched

**Type** – this column shows the type of the corresponding item/variable (like integer, string, float, etc.)

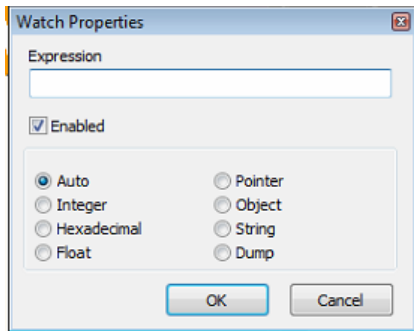
**Value** – this column shows the current value of the corresponding item/variable. Note that if the script has been executed up to a certain line only, the entry will represent the value of the item until this step

Right-clicking on any line in this area will invoke the following context menu:





If you select **Add** or **Edit**, the **Watch Properties** dialog will open. Another way for invoking this dialog is by pressing <Ctrl> + <F7>. In the *Expression* field enter the name of the item you wish to watch. Select the type of the variable by pressing one of the radio buttons below. If you choose **Auto**, the type of the variable will be obtained automatically from the script, if possible. If you wish to transform the result in another value type, you can select a type, different from the default type of your variable, as long as the two types are compatible.



Checking the  *Enabled* box will add a check-box to the corresponding row in the *Watch List*. The idea of this check-box is to give the user the ability to control which variables should be “watched,” and which should not be “watched” at a time. Thus, if an item is not checked , its value in the **Value** column will not be changed until the box next to it is checked  back.

The **Delete** and the **Clear** commands from the context menu are self-explanatory.

The *Watch* area is only visible when the  *Watch* box in the toolbar is checked.

#### 4. Debug View

This area shows all the syntax errors that appear in the script while debugging. Note that logical errors will not be traced by the script engine.

The *Debugmode* is only active when the  *Debug* box in the toolbar is checked.


**NOTE:** The **TitleBox** script engine allows you to check whether or not you are in *Debug* mode via a special variable, named [DEBUG], which is of type Boolean. When [DEBUG = True], this means that you are in *Debug* mode, and when [DEBUG = False], the Debug mode is off, so the script is executed without interruptions. Please, check *Example 13* in *Appendix 7* [below](#).

#### Toolbar:




Use the drop-down menu in the toolbar for selecting the programming *Language*, in which you are going to write your script. Your options are **Pascal**, **C++**, **Basic**, and **Java**.

Several buttons and check-boxes are situated after the drop-down menu. Their functions are as follows:

 - This is the **Insert** button. It is used for inserting **Modules**, **Objects**, **Slides**, **Tasks**, and **Data Providers**. When you press the black arrow next to it, a context menu will appear:

Use module	Ctrl+Alt+U
<b>Object</b>	<b>Ctrl+Alt+O</b>
Slide	Ctrl+Alt+S
Task	Ctrl+Alt+T
DataProviders	Ctrl+Alt+D




Here you can select the type of the item you would like to insert in your script. Notice that there is a combination of keys written next to each item. Thus, instead of clicking on the respective item from the **Insert** drop-down menu, you can simply enter the corresponding key combination. Furthermore, note that in the screenshot above the second line [Object] is marked with a bold font. This means that when you press only the **Insert**  button instead of the arrow next to it, an **Object** will be added to the script. If, next time you insert an item you select [Slide] from the context menu, its line will be bolded and a **Slide** will be added after clicking only the button instead of the arrow.


If you select the **Use module** option from the context menu above, you can insert any script, situated in **TitleBox**'s Library, regardless of the language, in which it is written. This option is very useful if you need to use the same function in different scripts. Please, check *Example 15* in *Appendix 7* [below](#) to see how modules are inserted in different script syntaxes.


 – Use the **New** button to clear the *Program View* and start working on a new script.

 – Use the **Open Script in editor** button to load an already saved script.

 – Use the **Save Script to file** button to save your script.

 – Use the **Undo changes** button to undo your latest changes. You can also press the <Ctrl>+<z> keys instead of this button. Up to 60 steps can be reversed.


 – Use the **Run Script** button to debug your entire script. You can also press the <F9> key instead of this button.

 – Use the **Execute next step** button to only execute the line after the one, where the cursor is situated. You can also press the <F8> keys instead of this button.

```
#language Pascal
var
  i, j: Integer;
  ar: array[0..9] of String;
  s: String;
begin
  for i := 0 to 9 do
    ar[i] := IntToStr(i);
  s := '';
  for i := 0 to 9 do
    s := s + ar[i];
  ShowMessage(s);
end.
```

In the example above, please notice the light-blue line with a green arrow in the beginning. After a step is executed via the button, described above, this step is marked like that.

 – Use the **Stop execution** button to stop debugging your script. You can also press the <Ctrl>+<F2> keys instead of this button.

 – Use the **Toggle breakpoint** button to insert a breakpoint in the script. You can also press the <F5> key instead of this button. When a breakpoint is added, the next time you start executing your script, it will pause executing at that breakpoint.



```
#language Pascal
var
  i, j: Integer;
begin
  j := 1;
  i := 0;
  while i < 10 do
  begin
    i := j + 1;
    Inc(i);
  end;
  ShowMessage(j);
end.
```


In the example above, please notice the red lines with a red circle in the beginning. After a breakpoint is included, the line that corresponds to the breakpoint will be marked like that.

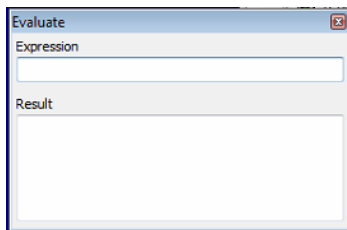
Also, be aware that you can add a breakpoint also by clicking in the grey area to the left of the script view. If you want to remove that breakpoint, simply click on it again.

*Debug* – If checked, the script will be debugged when executed. Otherwise, only the final result will be shown and no messages will appear in the *Debug* area at the bottom.

*Watch* – If checked, the *Watch* area will be visible.

*Browser* – If checked, the *Browser* area will be visible.

 - The **Evaluate expression** button invokes the dialog, shown below. In the *Expression* field you can enter different mathematical expressions, in which you can include variables from your script. After pressing <Enter>, the program will show you the result of the expression in the *Result* field. If you have entered a variable and/or other symbol that cannot be recognized by the script engine, it will return an error message in the latter field.

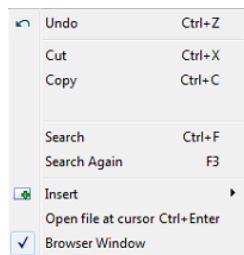


## Menu Bar

### File

The File menu consists of an **Open**, **Save**, and an **Exit** command, which are all self-explanatory. Be aware that your script can be automatically executed if you **Save** it under a name, recognized by **TitleBox**. Check the **Auto-executed Scripts** function, described [below](#).

### Edit



# TitleBox User Manual



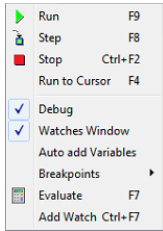
The edit menu consists of several commands. The first three, **Cut**, **Copy**, and **Paste**, are self-explanatory.

The **Insert** command is the same as the insert button, explained above in the [Toolbar](#) section.

The **Open file at cursor** command simply allows the user to load a script at the current location of the cursor. As visible in the screenshot above, this command can also be invoked with the <Ctrl> + <Enter> key combination.

The **Browser Window** command is the same as the  *Browser* checkbox in the [Toolbar](#), explained above.

## Execute



The Execute menu is very similar to the [Toolbar](#), explained above. However, it offers some additional commands, explained below. Also, it shows the equivalent keyboard buttons that could be used for the commands.

The **Run to Cursor** command is not included in the toolbar. When selected, this command will execute the script to the point, where the cursor is currently situated.

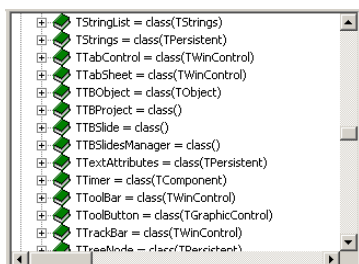
This menu also offers an additional function for the **Toggle** breakpoints. If you go to **Breakpoints**, a context menu will be opened, from which you can either add a breakpoint (**Toggle** command), or remove all breakpoints, via the **Clear All** commands.

If checked, the **Auto add Variables** option will automatically add all variables used in the script to the [Watch View](#) at the bottom of the **Create Program Script** dialog.

## Creating TitleBox Items

Take a look at the Browser View in the Create Program Script dialog. When you expand the classes section, a large list of classes appears. The classes are arranged in alphabetical order. Most of them are self-explanatory and refer to common programming options and functions. Several of them classes, however, were created strictly for **TitleBox**:

### TTBObject



This class operates all objects in **TitleBox**, Text objects, Pictures, Rolls and Crawls, 3D objects, etc. Be aware that the possible properties for all **TitleBox** objects are listed here. Some of them are strictly related to a particular type of object. For example, you cannot apply [ClockPrepare] to a Text object. If you try to apply a property or method that the object in question does not support, the script engine will simply ignore it.

There are two ways, in which you can create an object in **TitleBox**: with the **Create** and with the **CreateNew** command.

The **Create** command is used when an object has already created in the *Work Area* via the *Object Palette*. Thus, when you **Create** an object, you simply link your script command with that object by using the object name. If you use this option for creating objects, instead of typing in the name of the



particular object, you can simply press <Ctrl> + <Alt> + <O>, which will invoke a dialog with a list of all objects in the project, like the one shown below. To select an object to link to, double-click on its name in the list.

The **CreateNew** command, on the other hand, is used when the object you want to create does not already exist in your project. When you create it, you can define its X and Y parameters, as well as its width and height. If you do not define any parameters, it will be placed at the (0,0) coordinate in the grid.

A description of all properties, procedures, and functions that the [TTBObject] class supports is available in [Appendix 6 below](#). Furthermore, you can check [Example 1](#), [Example 2](#), [Example 5](#), and [Example 10](#) in [Appendix 7](#) below to see how it works in practice.

**NOTE:** If you try to create a new object with the name of an existing one, be aware that **TitleBox** will automatically add '1' to the end of the name, so that it does not coincide with the already existing name.

**IMPORTANT:** You should always set your objects free in your script via the **Free** command after you stop working with them. This is done to prevent your memory from overloading. Be aware that the **Free** command does not delete an objects. It simply breaks the link between the object and the program script.

**NOTE:** If you want to remove an object both from the script, and from the grid, you should use the **Delete** command.

**TIP (!)** Place your **Free** command after the **Try–Finally** command combination to make sure that the **Free** command is executed.

**TIP (!)** If you are using the [TTBObject] class to update text formatting in a certain text object, do NOT forget to use the **UpdateParams** command to apply the new formatting to the text. Check [Example 5](#) in [Appendix 7 below](#).

## TTBSlide

This class is used in the same way as the [TTBObject] class. You can create new slides (**CreateNew** command) and link to already created slides (**Create** command), and you need to set your Slide free after you stop working with it (**Free** command).

A description of all properties, procedures, and functions that the [TTBSlide] class supports is available in [Appendix 9 below](#). Furthermore, you can check [Example 4](#), [Example 5](#) and [Example 9](#) in [Appendix 10](#) below to see how it works in practice.

**NOTE:** If you attempt to create a new slide with the name of an already existing one, **TitleBox** will return an error in the *Debug* view.

**NOTE:** If you use the **Delete** command to delete a slide, only the slide will be removed and not the objects, contained in it. They will simply be moved to the 'zero' slide, which contains all objects that are not assigned to a particular slide.

## TTBSlidesManager

The [TTBSlidesManager] class is used for operating with the slides manager. Like the [TTBProject] and [TTBStorage] class, the user does not create, nor free this class, as it stores only one variable, [Slides], which is created upon **TitleBox** initialisation.

A description of all properties, procedures, and functions that the [TTBSlidesManager] class supports is available in [Appendix 7 below](#). Furthermore, you can check [Example 9](#) in [Appendix 7](#) below to see how it works in practice.

## TTBDataProvider

The [TTBDataProvider] class is used for operating with the data providers that are already set in the [Data Source Manager](#). Be aware that when you **Create** such a [TTBDataProvider] variable and link it to a certain provider, the variable in the script will control all the objects that are linked to that particular Data Provider. Furthermore, you need to set the [TTBDataProvider] variable **Free** after using it. This class is often used with [Transform Events](#).



# TitleBox User Manual



A description of all properties, procedures, and functions that the [TTBDataProvider] class supports is available in *Appendix 6* [below](#). Furthermore, you can check *Example 6*, *Example 7*, *Example 8*, and *Example 12* in *Appendix 7* [below](#) to see how it works in practice.

## TTBDataDistributor

The [TTBDataDistributor] class is used for operating with the data distributor properties that are already defined for each Data Provider in the [Data Source Manager](#). This variable is NOT set free after operation, since it is freed together with its corresponding Data Provider.

A description of all properties, procedures, and functions that the [TTBDataDistributor] class supports is available in *Appendix 6* [below](#). Furthermore, you can check *Example 7* in *Appendix 7* [below](#) to see how it works in practice.

## TTBProject

This class is used for controlling your whole Project. As it stores only one variable, [Project], which is created upon **TitleBox** initialisation, you do not create, nor do you set free a [TTBProject] class.

A description of all properties, procedures, and functions that the [TTBProject] class supports is available in *Appendix 9* [below](#). Furthermore, you can check *Example 10* in *Appendix 7* [below](#) to see how it works in practice.

## TTBStorage

The [TTBStorage] class is used for storing specific values to be used in different scripts and / or projects within one **TitleBox** session. It allows the user to save up to 100 values and it is extremely useful when you need to use the same values in more than one script or even a whole **TitleBox** project. As it stores only one variable, [Storage], which is created upon **TitleBox** initialisation, you do not create, nor do you set free a [TTBStorage] class.

A description of all properties, procedures, and functions that the [TTBStorage] class supports is available in *Appendix 6* [below](#). Furthermore, you can see how it works in practice in *Example 14* in *Appendix 7* [below](#).

## Additional Tips and Notes

There are two additional classes, which are not **TitleBox** specific but are commonly used in the **TitleBox** scripts for various operations and functions. They are the [TRichEdit] and the [TBitmap32] class.

The [TRichEdit] class is often used with all **TitleBox** text objects, especially with procedure TTBOject.TextAssign(RichEdit: TrichEdit), described in *Appendix 6* [below](#). It allows you to change all kinds of formatting in a given text selection.

The [TBitmap32] class is used for creating and controlling Bitmap32 images. It is often associated with procedure TTBOject.ImageAssign(Bmp32: TBitmap32), described in *Appendix 6* [below](#). A Bitmap32 image will not be shown in the **TitleBox** work area, unless you paste it in a certain object in the grid. These variables need to be set **Free** after using in the script.

A description of all properties, procedures, and functions that the [TBitmap32] class supports is available in *Appendix 6* [below](#).

For your convenience, check the table below for the major functions of the classes, specifically created for **TitleBox**. It also shows whether or not they have to be set Free in the script after operation:

Class	Function	Free
TTBOject	To be linked to or create all types of object in <b>TitleBox</b> .	Yes
TTBSlide	To be linked to or create slides in your project.	Yes



<b>TTBSlidesManager</b>	To be linked to the <a href="#">Slides Manager</a> in your project.	No
<b>TTBDataProvider</b>	To be linked to all Data Providers set in the <a href="#">Data Source Manager</a> .	Yes
<b>TTBDataDistributor</b>	To be linked to all Data Distributors, set in the <a href="#">Data Source Manager</a> .	No
<b>TTBProject</b>	To be linked to your current project in <b>TitleBox</b> .	No
<b>TTBStorage</b>	To save certain variables to be used in different scripts and / or projects within one <b>TitleBox</b> session.	No

**TIP (!):** If you are not certain whether you need to **Free** a certain variable or not, you can check the list of functions, procedures, and properties, related to its class in the [Browser View](#). If [procedure: Free] exists there, then you have to free that particular variable. Otherwise, the variable does not need to be set free.

## Working with More than One TitleBox Item Simultaneously

If you want to operate with more than one **TitleBox** object or slide at a time, the script engine allows you to group your desired items via the List commands. The Program Script Task has two types of list commands – for grouping of objects and for managing already grouped objects.

### Create Lists of Items

The program script engine allows you to operate with several items together via the List variables. In order to do that, first, you need to define a set of objects or slides to belong to a certain list. For this purpose, you can use the available List commands, which are situated within the TTBProject and TTBSlidesManager classes:

**ObjectsList** – This command is situated in [Classes → TTBProject] in the *Browser Area*. It lists all the objects in the project and you can choose which objects you wish to keep in the list, with which you are going to operate.

**TagsList** – This command is also situated in [Classes → TTBProject] in the *Browser Area*. It lists all objects that share the same tag. You can read more about Tags in TitleBox in the relevant section above.

**TasksList** – This command is also situated in [Classes → TTBProject] in the *Browser Area*. It lists all tasks in your project.

**SlidesList** – This command is situated in [Classes → TTBSlidesManager] in the *Browser Area*. It lists all slides in your project.

**DataProvidersList** – This command is situated in [Classes → TTBProject] in the *Browser Area*. It lists all the objects in the project and you can choose which objects you wish to keep in the list, with which you are going to operate.

### Manage Lists of Items

Once you have grouped your desired items via one of the **List** commands, described above, you can work with these groups within your script. There are three **List** functions, available in the script engine, [PlayList], [StopList], and [ToggleList]. All of them are situated in [Functions → TitleBox].

Please, check *Example 3*, *Example 6*, *Example 7*, and *Example 9* in [Appendix 7](#) below to see how to use the **TitleBox List** commands in practice.

## Error and Exception Messages

For the purposes of avoiding memory load and endless scripts, two error checks are implemented in the **TitleBox Program Script**:



## Object Free Check

If some of the objects, used in the script have not been set free with the **Free** command, the **Program Script** will show an error message, listing all the variables that have not been freed during the script execution.

## Execution Time Check

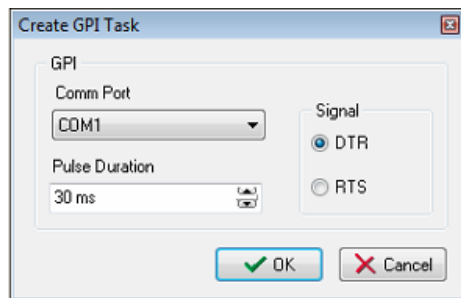
If the execution of a certain script takes more than 1 - 2 seconds, the **Program Script** will terminate the script execution and return an error message, stating at how many seconds the execution has been stopped.

## Exception Messages

Sometimes it is very useful to work with Exception Messages and the **Try – Except / Try – Catch** block, if you are working with a certain procedure in your script, which does not work in some exceptional cases. You can check *Example 11* in [Appendix 7](#) below to see how it works in practice.

## **GPI Task**

This **task** is designed for creating **GPI** pulses.

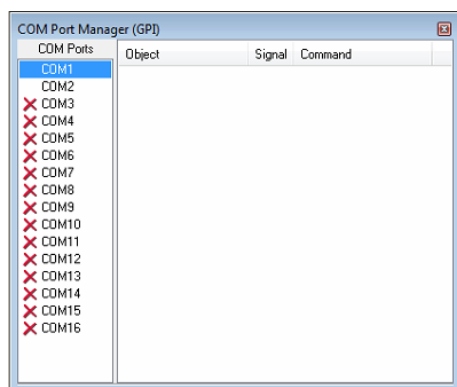


Into the **GPI task** window, specify the **GPI Signal** type (RTS or DTR) and to which COM port it should be sent. Specify also the Pulse duration in milliseconds.

Press **OK** to add this **task** to the **Task Manager** list.

The GPI events window can be started from **Project->Plugins-GPI manager** as well.

In the GPI manager, you can view all GPI events and the objects to which they have been assigned.



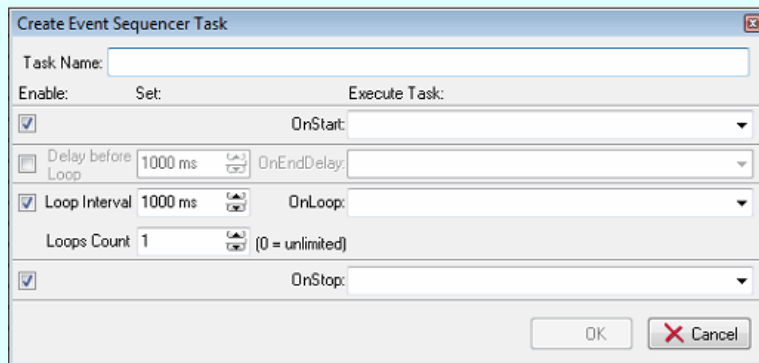
## **Event sequencer Task**

This **task** is designed for generating a time sequence for executing another **task**.

An *example* of such sequence is, if you want to start a Digital Clock, then to start a picture and move it horizontally on screen over a specific period of time and at finally to start another task. You will have the following sequence of tasks:



1. **Task** "Play Digital Clock 1"
2. **Task** "Play Picture 1" (to be started 5 sec. after the beginning of the sequence)
3. **Task** "Move Picture1" (to be started 10 times at every 5 seconds)
4. **Task** "Start timer1"



In this example, you have to fill the setup window like it is shown in the following screenshot:

Into Task Name, enter the name of your time sequence (sequence 1).

Into OnStart field, select the name of the first task in the sequence (Play Digital Clock 1). Select the name from the drop-down list. If you don't want to have task on sequence start, uncheck relative Enable check-box.

Into Delay before Loop field, enter the interval of delay in milliseconds (5000ms=5sec.). This is the delay after the sequence start and before the start of the loop task. Into OnEndDelay field enter the task which will be executed with this delay (PlayPicture1). Select the name of the task from the drop-down list with existing tasks. If you don't want to have delayed task, uncheck relative Enable check-box.

Into OnLoop field, enter the task which will be executed on every loop (Move Picture 1). Into Loop Interval field enter the interval for repeating the task (5sec=5000ms). Into Loops Count field enter how many times the task will be repeated (10).

If Loops Count is zero (0), the looping will be infinite, until task is running.

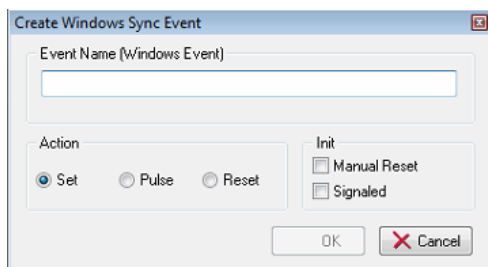
Into OnStop field, enter the task which will be executed at the end of the sequence (Start timer1). This task will start when the looping is finished or when a Stop sequence command is send.

When you are ready with definition of the sequence, press the OK button to save it. You will see two tasks in the task list, created automatically: Start sequence name and Stop sequence name. The command Stop sequence name usually is used to stop a sequence when the Loop Interval is zero (infinite).

## Play Sound

It is designed for creating sound **tasks**. Pushing the icon will open the **Play Sound** dialog and all you have to do is to browse for the sound you need. You can open all Direct-Show compatible sound files (\*.wav, \*.mp3, etc.). You can preview all sound **tasks** in the **Task Manager** using the **Play** and **Stop** buttons.

## Windows Named Event Task



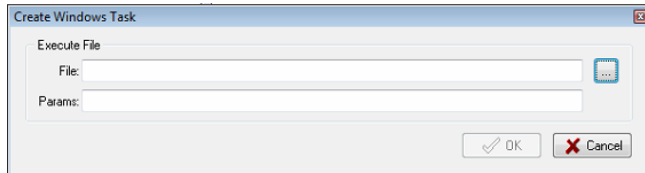
This **task** is related to Windows Named Event Objects. For more information about them, you can refer to Windows documentation.



The **Event Name task** was developed to synchronize the external applications to **TitleBox**. Thus, if there is a third party application that can accept certain event names, **TitleBox** can trigger events in these applications.

Each system event should have a Name. You can type it in the **Event Name** string. Below, you have to choose the **Action** and the type of **Initialization**.

## Create Windows Task



This **task** is related to executing of the external applications.

Into field **File** enter the executable file name (including the full file's path). You can start \*.exe \*.bat or \*.cmd files.

Into **Params** field, you can enter some parameters for starting the application.

*For example, if in the **File** is entered **AirBox**, into **Parameter** you can enter the name of the playlist which you want to start or the **AirBox** instance number.*

If there is a file association existing, you can enter into the File field, only the file name of the document to start the associated application.

*For example, if you have a file association for **Windows Media Player**, into field **File** you can enter only the location and name of the \*.mp3 file.*

## Auto-executed Scripts

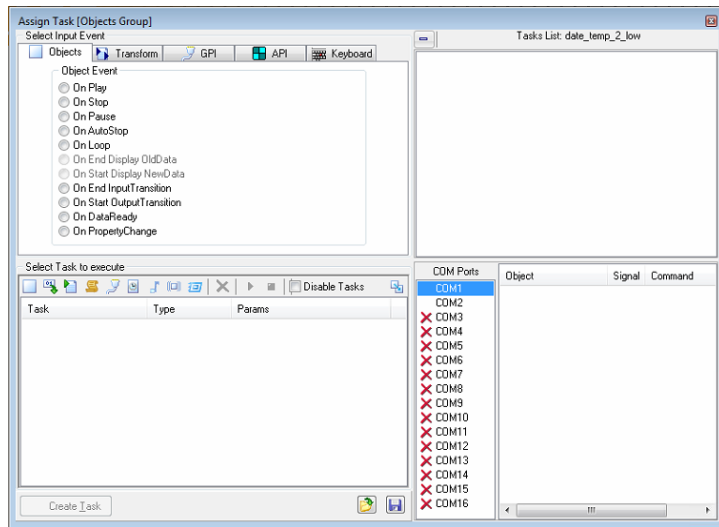
**TitleBox** provides an additional way to assign scripts to certain events. If you want a certain script, created either via the [Command Script Task](#), or by the [Program Script Task](#) to be executed when you **load**, **unload**, **start**, or **stop** your project, simply save the corresponding script with one of the following names:

- **"OnProjectLoad"** –this script will be executed right after a project is loaded;
- **"OnProjectUnload"** –this script will be executed right before unloading a project;
- **"OnProjectStart"** –this script will be executed right before starting a project;
- **"OnProjectStop"** –this script will be executed before stopping project.

**NOTE:** If you have more than one script, bearing one of the above listed names, **TitleBox** will only execute the first one from the list, i.e., the oldest one.



## Input Events



The **events** are triggers, on which the **tasks** are performed.

The **Input Event** dialog opens when you select an object, right mouse click and select **Task Manager** from menu.

The input events are grouped in four major groups: Object related events, GPI events, API events, Keyboard events.

### Objects events

The available object's **events** are: Play, Stop, Pause, AutoStop, Loop, End display old data, Start display new data, End Input transition, Start Output transition, Data Ready, Property Change.

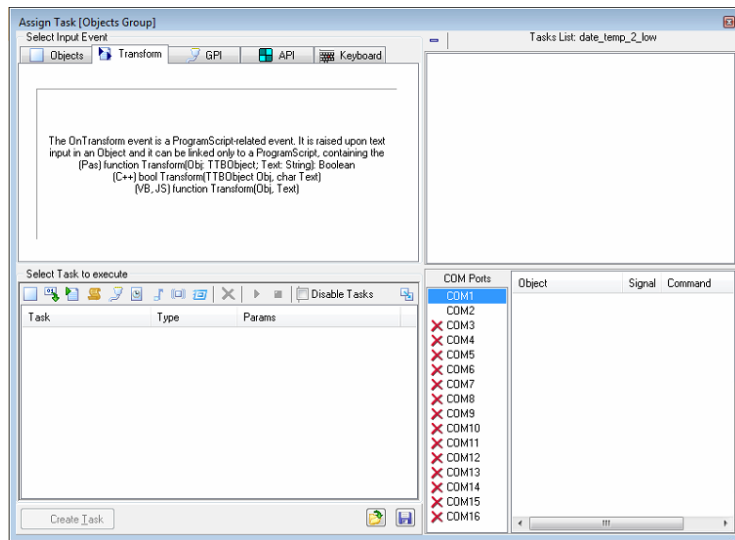
AutoStop – this event concerns rolls, crawl, and animation objects. This event is generated when a number of loops for the object is set and the last loop is executed.

End display old data and Start display new data events are connected to changing of the content into the rolls, crawls and animations. Because of the nature of these objects, it is possible to have both – old content and new content on screen. The End display old data event happens when the old data stops to be visible on the screen. And the Start display new data event happens when new data become to appear on the screen.

Data Ready – this event is related to changing the data into an object. When the new data is ready to be displayed, than Data Ready event is happening.



## Transform events




As explained in the image above, these events are used in combination with the [Program Script Task](#). In other words, the only way to use such an event is to link it to a certain **Program Script** by entering the following function in the script:

For **Pascal** script: `<function Transform(Obj: TTBObject; Text: String): Boolean>;`

For **C++** script: `<bool Transform(TTBObject Obj, char Text)>;`

For **Java** and **Visual Basic** script: `<function Transform(Obj, Text)>;`

The **Transform** event is used for transforming the information that is fed by a [Data Provider](#), [Net Control](#), or [File Link](#) to **TitleBox**. You can link such an event to all types of objects in **TitleBox**. Thus, you could change the format of the text that is provided by the data source, erase some parts of it, add new text to it, or even assign a picture to be displayed instead of the text that is being fed.

In order to create a **Transform** event for a certain object, two things need to happen. First, you need to save a program script, which contains the aforementioned transform function with a reference to that particular object in the [Program Script Task](#). Then, you need to click on that object, press the **Task**  button on the toolbar, and go to the **Transform** tab. Select the script that contains the corresponding **Transform** function from the list in the *Select Task to execute* area, and press the Create Task button.

Notice that the **Transform** function is of type **Boolean**. This means that you should implement a **True** or a **False** value to be returned by your script to the particular object, containing the **Transform** event. If a **False** value is returned to the object, it will ignore the **Transform** event and output the information, as provided by the **Data Provider** linked to it. Otherwise, if the object receives a **True** value from the **Transform** function, it will show the output as it is provided by the latter.

Please, check *Example 12* in [Appendix 7](#) [below](#) to see how the **Transform** function works in practice.

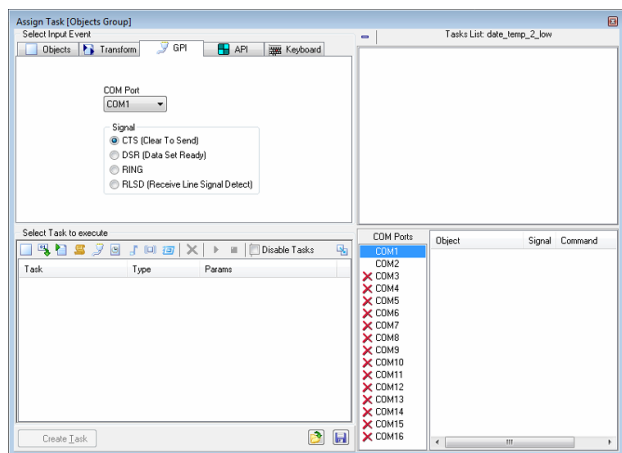
**IMPORTANT:** The **Transform** event only executes the **Transform** function, shown above, NOT the whole script. Thus, you do NOT need to save a separate script file, containing the transform function only. You can simply add the desired transform function to any program script that you use for your project.

**NOTE:** If the script that you assign to the **Transform** event does NOT contain a **Transform** function, **TitleBox** will simply ignore this event. Rather, it will display the corresponding **Data Provider** entry exactly as it is being fed to the object, linked to it.

**NOTE:** If there are any modal windows (e.g., Show Message dialogs) within the **Transform** function, **TitleBox** will ignore them. This precaution is introduced for the purpose of preventing the **Transform** function execution from postponing.



## GPI events



This feature in **TitleBox** allows controlling objects through receiving certain signals on the PC COM ports. In order to specify the “meaning” of each signal to each COM port, select the Com port and then select the type of signal.

## API events

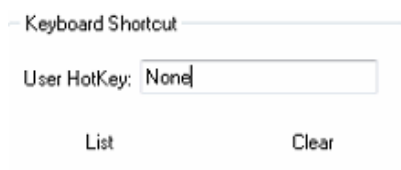


These events are connected to usage of **TitleBox API** (Application Program interface).

In **TitleBox API**, there is a command “EVENT xxx”, where xxx is a number equal or higher than 1001. In **Task Manager->API events**, you can define a specific number which to be used further in such Event command.

When to the API event (for example: number=1001) is assigned a task and **TitleBox** receives a command “EVENT xxx” (xxx=1001), then the task assigned to this API event will be executed

## Keyboard events



Keyboard events are user definable keyboard shortcuts for some action.

These events are connected to some **task**, so when the keyboard combination is pressed, the assigned **task** is executed.

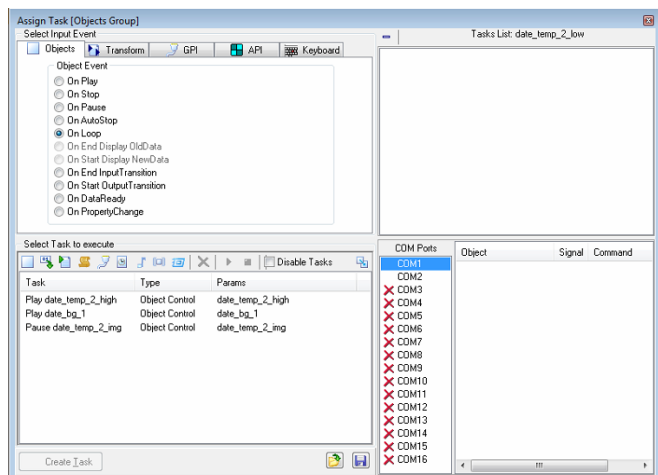
Press the **List** button to see all created keyboard shortcuts.

Press the **Clear** button to delete all created keyboard shortcuts.





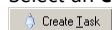
## Assigning a task to an event



To assign a **task** to an **event**, you have to open the **Assign Task** dialog. It is available after right mouse clicking on an object in **TitleBox** project and selecting **Assign Task** from menu.

You will see the list of already created tasks. If there are no tasks, first create them into **Task** area of the window.

Select an **event** from the available events and then select a **task** from **Task** list. Press the **Create Task** button in the bottom of the **Assign task** window



In the right side of the **Assign Task** window, you can see the list of all **tasks** related to the currently selected object.

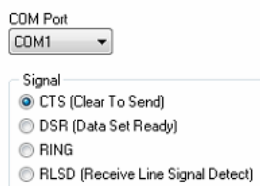
If you want to see the **tasks** related to another object, just select this object and press the  **task list** button from the main **TitleBox** window.

To see the **tasks**, which are not related to any object (keyboard shortcuts), unselect all objects in **TitleBox** and press the  **task list** button.

There is a specific definition, if you want to control the whole **TitleBox** project on incoming GPI signal.

To invoke the global **Assign GPI event** window, unselect all objects in **TitleBox** project, right mouse click and select Assign GPI event from menu.

Select the PC COM port and the type of the GPI signal which will be received (CTS, DSR, RING or RLSD).



In the lower part of the screen, select the Command which **TitleBox** have to execute when the GPI signal is received. There are three possible commands: Play, Stop and Toggle.

In the *example* from screenshot above, when on port COM1 a CTS signal is received, **TitleBox** will start to play the current project

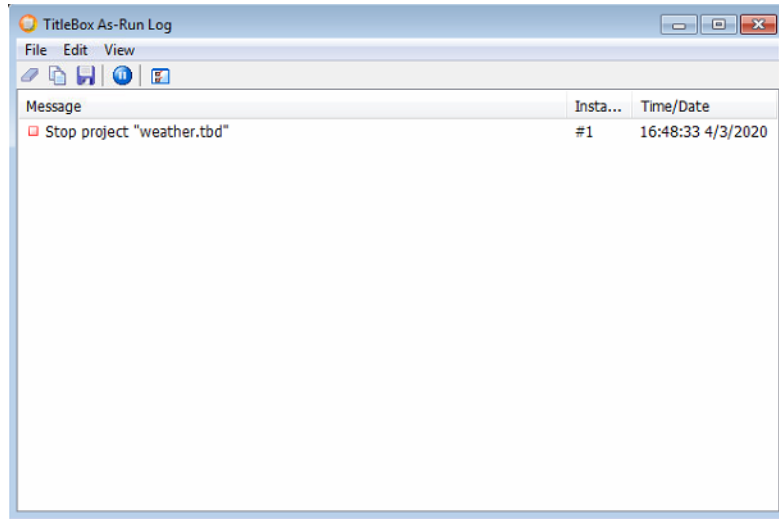


## AS-RUN LOG

The As-Run Log is an external application, displaying log information for all **TitleBox** instances, which are running locally. It can be enabled/disabled from the dedicated button in the main **TitleBox** toolbar. Also, it is configured per instance in [Project menu → Options ⇨ As-Run Log](#) tab.

## User Interface

The interface of the **As-Run Log** is very simple and user-friendly.





## Grid


The largest area in the **As-Run Log** is dedicated to displaying messages about all the events that take place in the local **TitleBox** instances, which have enabled logging. Each **Message** contains information about the **Instance**, to which it refers, as well as the **Time/Date** of the event.


## Toolbar

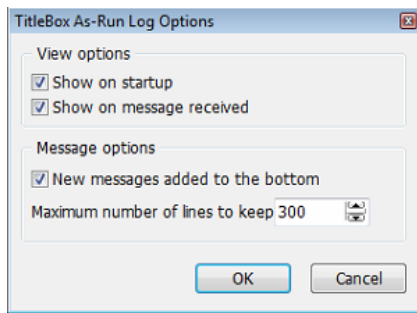
The toolbar allows you to perform the following simple commands:

 **Clear window**—press this button to clear the log window

 **Copy selected lines**—press it to copy the currently marked lines from the log. Use the <Shift> and <Ctrl> keys to select lines

 **Save to file**—press this button to save the current log messages to file

 **Pause messages**—press this button to temporary stop logging messages



 **Options...** - press this button to invoke the Options dialog, which looks like the image below:

Use the check-boxes to configure the following:

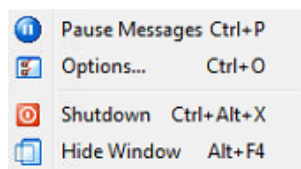
- Show on startup*—check this box if you want to view the **As-Run Log** when you initiate **TitleBox**
- Show on message received*—check this box if you want the log to be displayed every time it receives a message from **TitleBox**
- New messages added to the bottom* —check this box if you want the log messages to be displayed at the bottom of the field as opposed to the top

In the *Maximum number of lines to keep* specify the number of lines you wish to keep visible in the **As-Run Log** interface. You can enter a maximum of 1,000,000 lines. If you enter 0, then the log will display all messages without clearing any lines.

## Menu Bar

The **As-Run Log** menu bar contains the following menus:

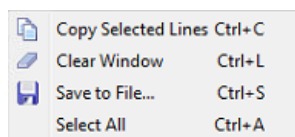
### File Menu



The first two commands are the same as the ones in the toolbar – the **Pause Messages** command is used for temporary stopping the log messages and the **Options** command invokes the options dialog, described above.

You can also exit from the **As-Run Log** via the **Shutdown** command or just minimize it with the **Hide Window** command.

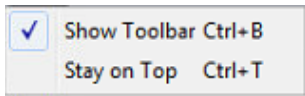
### Edit Menu



Commands here also repeat the commands from the toolbar. You can copy the lines selected in the log, clear the log window, or save the messages to a file. Also, you can select all lines via the **Select All** command.



## View Menu



The **View** menu allows you to show and hide the toolbar from the user interface. Also, you can check the **Stay on Top** command so that the **As-Run Log** is always displayed on top of the other windows.



## APPENDIX 1 - PlayBox GPI

### GPI IN AIRBOX

*GPI* (General Purpose Interface) is implemented in **AirBox**, **TitleBox** and **CaptureBox** as a set of triggers, associated to certain pins on the standard *PC RS-232* Serial Ports (*COM1*, *COM2*, etc.). In order to function, the *COM* port should be correctly installed in the Windows environment (See Device Manager → Ports).

**GPI-IN** can be used to slave **AirBox**, **TitleBox** and **CaptureBox** to triggers from external devices or simple contact switches, 4 triggers per *COM* port.

**GPI-OUT** can be used to slave external equipment to events taking place in **AirBox** or **TitleBox**, 2 triggers per *COM* port.

### GPI PINOUT

Each *COM* port accommodates 4 **GPI-IN** pairs and 2 **GPI-OUT** pairs, but not at the same time. A particular *COM* port can be assigned as either **IN**, **OUT** or **Unused**.

**GPI-IN** pairs are located at output pins *DTR&RTS* and input pins *DSR*, *CTS*, *RI*, *CD*. You can use any of the output pins for supplying voltage to the *GPI* circuit.

**GPI-OUT** pairs are located at output pins *DTR&RTS*, the first *GPI* trigger generates a pulse on *DTR*, and the second *GPI* trigger generates a pulse on *RTS*.

### GPI-IN IMPLEMENTATION

The simplest triggering device would be a pair of wires running from the *COM* port *GPI* pair, soldered to a momentary contact switch. This switch can be either **Push Button Normally Open** (PBNO) or **Push Button Normally Closed** (PBNC). The trigger type is configured in *AirBox GPI Settings Panel* as **High** or **Low** pulse. Many external devices like switchers or mixers have dedicated *GPI* connectors or screw terminals with a description of what *GPI* trigger type (low or high pulse) was implemented. Depending on the trigger setup, a trigger can be a temporary closing or opening of the *GPI* circuit. High pulse means the *GPI* trigger is activated when the circuit is temporary closed. Low pulse means the *GPI* trigger is activated when the circuit is temporary opened.

According to the number of installed *COM* ports, **AirBox** supports up to 32 **GPI-IN** triggers (up to 8 *COM* ports with 4 **GPI-IN** triggers per port).

### GPI-OUT IMPLEMENTATION

In order to control external devices by *GPI*, a simple 12V contact relay should be used. It should be connected to the corresponding *COM* port pin pair (4-6 or 7-8) and it should close or open the *GPI* circuit of the external device. Depending on the **AirBox** setup, the *GPI* trigger will generate a pulse (low or high, user-defined) to the corresponding output pin for a short user-defined period (pulse duration).

According to the number of installed *COM* ports, **AirBox** supports up to 16 **GPI-OUT** triggers (up to 8 *COM* ports with 2 **GPI-OUT** triggers per port).

### AIRBOX AS A GPI SLAVE

A wide variety of **AirBox** actions can be associated to a *GPI* trigger:

- Start playback
- Pause/Resume playback
- Stop playback
- Jump to next clip
- Jump to specific clip (predefined playlist index)



- Jump to specific location in the clip/playlist (predefined timecode)
- Playlist reset (during stop mode only)
- Hardware reset (terminates the playback!)
- Turn logo on
- Turn logo off
- Cue the selected clip
- Cue to specific clip
- Jump to bookmark in time range
- Jump to bookmark name

## AIRBOX AS A GPI MASTER

**AirBox** can activate a *GPI* trigger through specially designated GPI Output event. Please check the following page for GPI Output reference.

## AIRBOX GPI SETTINGS PANEL

All **GPI-IN** triggers can be assigned to a specific **AirBox** action. From *AirBox Settings* → *Modules*⇒*Remote Control*, select "GPI Input" **enabled**, to allow the **GPI-IN** trigger. Press the **Configure** button, to invoke a table for setting the *COM* ports, GPI groups and the available actions for them.

All **GPI-OUT** triggers can be activated by specific **AirBox** event: From *Settings* → *Modules*⇒*Remote Control*, select "GPI Output" **enabled**, to allow the **GPI-OUT** trigger. After that you can Add/Insert GPI Output event in the playlist, by right mouse clicking.

## RS232 9-PIN D-SUB PINOUT REFERENCE

Pin	Name	Description	Direction
1	CD	Carrier Detect	In
2	RXD	Receive Data	In
3	TXD	Transmit Data	Out
4	DTR	Data Terminal Ready	Out
5	GND	System Ground	-
6	DSR	Data Set Ready	In
7	RTS	Request to Send	Out
8	CTS	Clear to Send	In
9	RI	Ring Indicator	In

## GPI INPUT REFERENCE

(Triggers that control **AirBox**, **TitleBox**, and **CaptureBox**)



<b>GPI Input</b>	<b>Name</b>	<b>Contact Pins</b>
GPI 1	CST	8 + 4
GPI 2	DSR	6 + 4
GPI 3	RI	9 + 4
GPI 4	CD	1 + 4

## **GPI OUTPUT REFERENCE**

(Pulses sent out from **AirBox** and **TitleBox**)

<b>GPI Output</b>	<b>Name</b>	<b>Contact Pins</b>
GPI 1	DTR	4 + 5
GPI 2	RTS	7 + 5

## APPENDIX 2 – Graphic Rules’ Commands, Used for Communication between AirBox and TitleBox

In the table below you can find a list of commands, used in the AirBox [Graphic Rules](#) option.

**IMPORTANT:** All of the commands used in Graphic Rules are CASE SENSITIVE!

<b>LOAD_TEMPLATE=TB_ProjectName.tmp</b>	Points out which template (created from a TitleBox project) is addressed.
<b>ObjectName.play</b>	Sends a play command for the corresponding TitleBox object, designated by its name.
<b>ObjectName.stop</b>	Sends a stop command for the corresponding TitleBox object, designated by its name.
<b>ObjectName.text</b>	Used for adding text to the corresponding TitleBox object, designated by its name.
<b>ObjectName.media</b>	Used for adding media to the corresponding TitleBox object, designated by its name.
<b>PlaylistColumn.text=%clip_PlaylistColumn%</b>	This line contains the name of the AirBox playlist column [PlaylistColumn] object [Title] and a description of the text that it should contain.
	For example, the line [Title.text=%clip_title%] contains the name of the TitleBox object [Title] and a description of the text that it should contain. I.e., in this case, we will extract information from the Title column of the relevant playlist entry.
<b>%clip_title%</b>	Use this line to show the text from the corresponding title field in the AirBox playlist.
<b>%clip_location%</b>	Use this line to show the text from the corresponding location field in the AirBox playlist.
<b>%clip_category%</b>	Use this line to show the text from the corresponding category field in the AirBox playlist.
<b>%clip_notes%</b>	Use this line to show the text from the corresponding notes field in the AirBox playlist.
<b>%clip_star%</b>	Use this line to show the text from the corresponding star field in the AirBox playlist.
<b>%clip_start%</b>	Use this line to show the text from the corresponding start time field in the AirBox playlist.
<b>%clip_duration%</b>	Use this line to show the text from the corresponding duration field in the AirBox playlist.
<b>%clip_start_date%</b>	Use this line to show the text from the corresponding start date field in the AirBox playlist.
<b>%metadata_MetadataName%</b>	Use this line to show the text from the corresponding metadata, designated by its name.



<b>%clip_title[+n]%</b>	Use this line to display information about the title of an upcoming clip in the playlist, where [+n] is the off-setter.
<b>%clip_title[BM_Test1]%</b>	Use this line if you want to use a bookmark for presenting information about an item in the playlist. Here Test1 is the Bookmark name and [+n] is the off-setter.
<b>%clip_start[+n]{HHMMSSFF}%</b>	Shows the start time and the title of an up-coming clip. Here Next is the name of the text object in TitleBox, [+n] is the off-setter, and {HHMMSSFF} is the time format to be used. Be aware that the time format can also be {HH} or {HHMMSS}.
	In case you are in NTSC mode, the start time is displayed in accordance to the PC system time. Thus, instead of {HHMMSSFF}, here you should use {T} or {TT}, where:  {T} stands for short system time  {TT} stands for long system time
<b>%clip_title[CAT_CategoryName+n]%</b>	Use this line to display the title of the next clip (if n=0) down the playlist that belongs to the category, designated by its name. You can also use [+n] as an off-setter.
<b>%clip_start[CAT_CategoryName+n]%</b>	Use this line to display the start time of the next clip (if n=0) down the playlist that belongs to the category, designated by its name. You can also use [+n] as an off-setter.
<b>{!LOGO_ON!} / {!LOGO_OFF!}</b>	Use this command to show / hide the logo from the screen.
<b>SHOW_LOGO_PRESET_n</b>	Use this command to display logo preset number [n], where [n] is between 1 and 16 and stands for the respective logo Preset, as configured in Settings → <a href="#">Logo</a> .
<b>{!AUTOMATION_ON!} / {!AUTOMATION_OFF!}</b>	Enables / disables the incoming GPI / DTMF triggers.
<b>{!MUTE_ON!} / {!MUTE_OFF!}</b>	Switches the audio on / off.

## APPENDIX 3 – Integration of AirBox with TitleBox

### **TitleBox settings:**

- 1) Run **Programs > PlayBox Technology Ltd. > TitleBox > PLNetInst.exe** and select a folder for your **TitleBox** templates.
- 2) Create your **TitleBox** projects.
- 3) Export them via **Network > Export project as template**, giving them respective names.
- 4) \*.tmpl (template) files are exported in the templates folder.
- 5) Go to **Project\Options**, look at **Network** tab. Remember the TitleBox channel ID and Port values. Confirm any changes.
- 6) Go to **Network > Net control**. Run it.

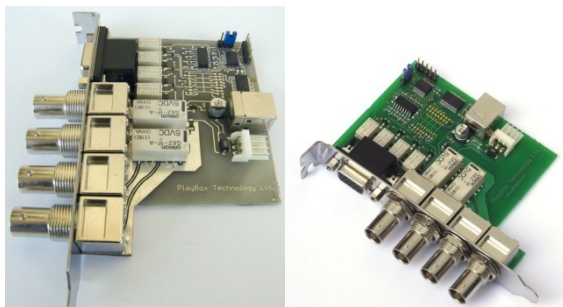
That's all for **TitleBox**. Now it stays in standby mode and executes the commands coming from **AirBox**. Please note that it is not possible to edit objects in **TitleBox** while in this mode.

### **AirBox settings:**

- 1) Go to **Settings → Modules**, then to **Remote control** tab. Enable TitleBoxNetCtrl Output setting with **Yes**. Click Configure.  
Enter same Channel ID and Port values as in **TitleBox** settings. Confirm.
- 2) Go to **Events → Add/Insert event → TitleBoxNetCtrl Output**
- 3) There are two modes for event insertion - 'Wizard' or 'Advanced' (selectable through the **Advanced** button). You are recommended to choose the 'Wizard' mode for now. The functions are self-explanatory, but since this module is still under development, some of them are not functioning as desired...
- 4) In 'Wizard' mode you can choose between Template Control and Play Project - the first one is for global **TitleBox** control commands; the second one is for project/objects control commands.
- 5) If you have entered Play Project mode, further you can select your project by list - all exported template projects should appear in this list. Select a whole project or some objects from it.
- 6) Click **Finish**. That's it!

Run **AirBox**. When the time for a **TitleBox** event approaches, respective commands are being sent to **TitleBox** and it runs the appropriate objects.

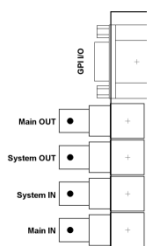
## APPENDIX 4 – PlayBox GPI board and Bypass Relay board



The PlayBox GPI Relay board has **4 GPI Inputs, 4 GPI Outputs with Relays, 1 Bypass relay (up to 2.6GHz)**

The following table is showing the pinout of the 15pin connector

Pin No	Signal Name	Remarks
1	NC	Not Connected
2	GPI OUT 3	Contact pin 2-7
3	GPI OUT 2	Contact pin 3-8
4	GPI OUT 1	Contact pin 4-9
5	GPI OUT 0	Contact pin 5-10
6	GND	Common Ground
7	GPI OUT 3	Contact pin 2-7
8	GPI OUT 2	Contact pin 3-8
9	GPI OUT 1	Contact pin 4-9
10	GPI OUT 0	Contact pin 5-10
11	+5V	5V DC From the board
12	GPI IN 0	Active - Short to +5V
13	GPI IN 1	Active - Short to +5V
14	GPI IN 2	Active - Short to +5V
15	GPI IN 3	Active - Short to +5V



**Main IN** is the signal which is coming from outside the system

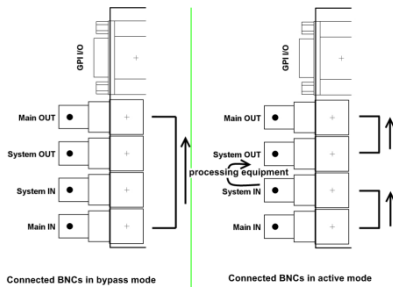
**Main OUT** is the signal which is going outside the system

**System IN** is the Bypassed system input

**System OUT** is the Bypassed system Output

All GPI inputs are using optocouplers.

All GPI outputs are relays. Both side of the contact are available on the 15pin connector.



## **Technical Specification**

Board size: 100x90mm

Bypass Relay:

- Tested Resolutions: PAL, NTSC, 720p50/60, 1080i50/60/59.94

- Contact switch time: 10ms

GPI Output:

- Maximum switching current: 0.7A

- Maximum switching Voltage: 220V AC

- Maximum switching capacity: 40W

- Minimum switching voltage: 250uV

- Resistive load: min 100 000 operations

- Contact switch time: 10ms

GPI Input:

- Maximum voltage to the input: 12V

Board Power:

- Using Floppy type connector from the machine

- Board Maximum Current: without using the +5V VCC on the connector 300mA

Board control: Thru USB (using internal connector is highly recommended)



## APPENDIX 5 – TitleBox specific Class Properties, Functions, and Procedures Explained

### TTBOBJECT (INHERITOR OF TOBJECT)

TTBObject Property/Procedure/Function	Description	Objects That Use It
<b>function Create(const ObjName: String): Constructor</b>	Links to an existing object in the project	All
<b>function CreateNew(const ObjType, ObjName: String): Constructor</b>	Creates a new object in the project and links to it	All
<b>procedure Free</b>	Frees a TTBobject variable	All
<b>procedure Delete</b>	Deletes currently selected object and sets it free	All
<b>procedure Play</b>	Runs currently selected object	All
<b>procedure Stop(AutoStop: Boolean = False)</b>	Stops currently selected object	All
<b>procedure Pause</b>	Pauses object's execution	All
<b>procedure Toggle</b>	Switches object's status from Play to Stop and vice versa	All
<b>procedure UpdateParams</b>	Updates the text when formatting properties are changed through the script	Text, Roll, Crawl, Chat objects
<b>function MoveToSlide(const SlideName: String): Boolean</b>	Moves the object to an existing slide	All
<b>procedure Move(IncX, IncY: Integer; Frames: Integer = 0)</b>	Moves the selected object by <IncX> and <IncY> in the respective directions.  <IncX> and <IncY> may be positive or negative values. The movement is relative to object's current position.	All

# TitleBox User Manual



TTBObject Property/Procedure/Function	Description	Objects That Use It
	Optionally a number of frames may be specified to define the duration of the change.	
<b>procedure MoveTo(PosX, PosY: Integer; Frames: Integer = 0)</b>	Same as procedure Move command except that <PosX> and <PosY> define the new position as absolute values.	All
<b>procedure SizeTo(Width, Height: Integer; Frames: Integer = 0)</b>	Sets object's size according to the specified values.  If some of the values are 0, the respective property of the object is not changed.	All
<b>procedure Rect(Left, Top, Right, Bottom: Integer)</b>	Sets object's position and size with one command	All
<b>procedure RectTo(Left, Top, Width, Height: Integer; Frames: Integer = 0)</b>	Defines a new position of the object as absolute values and sets the object's size according to the specified values. If some of the values are 0, the respective property of the object is not changed.	All
<b>procedure TopOne</b>	Moves the currently selected object with 1 layer to top (Z-order)	All
<b>procedure BackOne</b>	Moves the currently selected object with 1 layer to bottom	All
<b>procedure ToTop</b>	Moves the currently selected object to top	All
<b>procedure ToBack</b>	Moves the currently selected object to bottom	All
<b>procedure Loop(Start: Integer; Stop: Integer = -1)</b>	Defines the number of loops to run. If you set '0' the object will be looped endlessly.	Roll, Crawl, Animation File, Animation Sequence, Banner, Sound, Digital Clock objects
<b>procedure TextAssign(RichEdit: TRichEdit)</b>	Assigns an object of type TRichEdit, i.e., places its text with its formatting in the object at matter	Text, Roll, Crawl, Chat objects
<b>procedure TextSelect(StartPos: Integer = -1; EndPos: Integer = -1)</b>	Use this command to change the parameters of a part of the text. Selects part of the text starting at [StartPos] and ending at [EndPos], where the first letter of the text is indexed with '1'. Use a '-1' value for selecting the text from the beginning, until the end, or both, as follows: <ul style="list-style-type: none"> <li>▪ If [StartPos] = -1 and [EndPos] has a positive integer value, the text will be selected from the beginning to the letter with index, equal to the [EndPos] value.</li> <li>▪ If [StartPos] has a positive integer value and [EndPos] = -1, the text will be selected from the letter with index, equal to the [StartPos] value until the end</li> <li>▪ If [StartPos] = -1 and [EndPos] = -1 the whole text will be selected</li> </ul>	Text, Roll, Crawl, Chat objects

# TitleBox User Manual



TTObject Property/Procedure/Function	Description	Objects That Use It
<b>property AutoWrap: Boolean</b>	Enables / Disables the auto wrapping of the selected text	Text, Roll, Crawl, Chat objects
<b>property FontSize: Integer</b>	Sets the font size for the selected text.	Text, Roll, Crawl, Chat objects
<b>property FontName: String</b>	Sets the font for the selected text.	Text, Roll, Crawl, Chat objects
<b>property FontShadowSize: Integer</b>	Sets the font shadow size to the selected text.	Text, Roll, Crawl, Chat objects
<b>property FontStyle: TFontStyles</b>	Sets a variable of type TFontStyle with value fsBold, fsItalic, fsStrikeout, or fsUnderline to the selected text.	Text, Roll, Crawl, Chat objects
<b>property FontShadowAngle: Integer</b>	Sets the font shadow angle to the selected text.	Text, Roll, Crawl, Chat objects
<b>property FontShadowOffset: Integer</b>	Sets the font shadow offset to the selected text.	Text, Roll, Crawl, Chat objects
<b>property FontShadowColor: Integer</b>	Sets the font shadow color to the selected text.	Text, Roll, Crawl, Chat objects
<b>property FontEdgeSize: Integer</b>	Sets the font edge size to the selected text.	Text, Roll, Crawl, Chat objects
<b>property FontEdgeColor: Integer</b>	Sets the font edge color to the selected text.	Text, Roll, Crawl, Chat objects
<b>property FontLineSpace: Integer</b>	Sets the line space of the font to the selected text.	Text, Roll, Crawl, Chat objects
<b>property FontColor: Integer</b>	Sets the font color to the selected text.	Text, Roll, Crawl, Chat objects
<b>property FontJustify: Integer</b>	Justifies the font horizontally in accordance to the following values:  '0' for Center  '1' for Left  '2' for Right	Text, Roll, Crawl, Chat objects
<b>property FontAlignment: Integer</b>	Vertically aligns the font in accordance to the following values:  '0' for Bottom  '1' for Center  '2' for Right	Text, Roll, Crawl, Chat objects
<b>property FontUseFill: Boolean</b>	Disables / Enables fill of the text	Text, Roll, Crawl, Chat objects

# TitleBox User Manual



TTBObject Property/Procedure/Function	Description	Objects That Use It
<b>property FontUseOutline: Boolean</b>	Disables / Enables outline of the text	Text, Roll, Crawl, Chat objects
<b>property FontUseShadow: Boolean</b>	Disables / Enables shadow of the text	Text, Roll, Crawl, Chat objects
<b>property Name: String</b>	Sets or gets the name of the object	All
<b>property Text: String</b>	Sets or gets text data of the selected object. ANSI, UTF8 and RTF are supported.	Text, Roll, Crawl, Chat objects
<b>property ObjectType: String</b>	Returns the type of an object	All
<b>property State: TObjectState</b>	Returns the property state: autostop, pause, play, or stop	All
<b>property Color: Integer</b>	Sets or gets the color of the object	All
<b>property Tag: Integer</b>	Sets or gets the tag number of the object	All
<b>property Top: Integer</b>	Sets or gets the Y coordinate of an object in terms of pixels	All
<b>property Left: Integer</b>	Sets or gets the X coordinate of an object in terms of pixels	All
<b>property Width: Integer</b>	Sets or gets the width of an object in pixels	All
<b>property Height: Integer</b>	Sets or gets the height of an object in pixels	All
<b>property Delay: Extended</b>	Sets or gets a delay play time for an object in seconds	All
<b>property Speed: Extended</b>	Sets or gets the speed of an object. If you set a speed of '0' the object will not move. If you set a negative speed value you will reverse the movement of the object.	Rolls, Crawls, Animation Files, Animation Sequences, Mesh Files, Chat Notes/Lines/Rolls
<b>property Link: String</b>	Links the specified file to a TEXT or GRAPHICS object. The file's content is used as a data source for the object.	Text, Roll, Crawl objects
<b>property LinkAutoPlay: Boolean</b>	Disables / Enables Auto play of the object linked to a file whenever the data in the file changes.	Text, Roll, Crawl objects
<b>property TScrollDrawStyle</b>	<b>ScrollStyle:</b> Defines the scroll style of the object by setting a TScrollDrawStyle variable to it, which can have the following values:	Rolls, Crawls, Animation Files, Animation Sequences, Mesh Files, Chat objects





TTBObject Property/Procedure/Function	Description	Objects That Use It
	<p>[ssColorBlank] – the background of the current text will remain until the text disappears from the screen. The new text's background will appear as soon as the text is displayed.</p> <p>[ssEmpty] – there will be background only under the displayed text. As soon as the text ends, the background will disappear</p> <p>[ssSigned] – no space between the backgrounds of two successive texts</p>	
<b>procedure ClockPrepare</b>	Sets the initial clock time and waits for start command.	Digital Clock Objects
<b>procedure ClockStart</b>	Starts the clock.	Digital Clock Objects
<b>procedure ClockStop</b>	Stops the clock.	Digital Clock Objects
<b>property ClockIntermediate: Boolean</b>	Disables / Enables clock freeze.	Digital Clock Objects
<b>Property ClockVisible: Boolean</b>	Shows / hides a clock object	Analogue/Digital Clock Objects
<b>property ClockCountdown: Boolean</b>	<p>Valid only in counter mode.</p> <p>When enabled the counter counts down.</p> <p>If disabled, the counter counts up.</p>	Digital Clock Objects
<b>property ClockStyle: Boolean</b>	<p>Defines the mode of operation for the Clock object.</p> <p>If True – object is in Clock mode and runs as an ordinary digital clock</p> <p>If False – object is in Counter mode counts from initial to stop time if the option ClockUseStopTime is on.</p>	Digital Clock Objects
<b>property ClockOffset: Extended</b>	<p>Used to create clocks for different time zones.</p> <p>Specifies a time offset related to the current time. The format is:</p> <p>&lt;hh:mm:ss&gt;</p>	Analogue/Digital Clock Objects
<b>property ClockInitTime: Extended</b>	Sets the clock initial time in format <hh:mm:ss>	Digital Clock Objects
<b>property ClockStopTime: Extended</b>	Sets the clock stop time in format <hh:mm:ss>	Digital Clock Objects
<b>property ClockUseStopTime: Boolean</b>	Disables / Enables the use of the clock stop time.	Digital Clock Objects



TTBObject Property/Procedure/Function	Description	Objects That Use It
<b>property ClockMask: String</b>	Sets a mask for the format the time will be displayed in. The following options are available: HH:MM, HH:MM:SS, HH:MM:SS:MS, HH:MM:SS:FR, MM:SS, or SS	Digital Clock Objects
<b>procedure InEffect(Effect: TEffectType; Duration: Extended = 1.0; Motion: Integer = 0; FullScreen: Boolean = True)</b>	Sets an IN effect for the selected object	All
<b>procedure OutEffect(Effect: TEffectType; Duration: Extended = 1.0; Motion: Integer = 0; FullScreen: Boolean = True)</b>	Sets an OUT effect for the selected object	All
<b>procedure LoadImage(const FileName: String)</b>	Loads the specified file in the selected object	Text, Picture Objects
<b>procedure ImageAssign(Bmp32: TBitmap32)</b>	Assigns an already defined TBitmap32 image to an object	Text, Picture Objects
<b>property TStretchDrawStyle</b>	<p><b>ImageStyle:</b> Defines image display style from the following options:</p> <ul style="list-style-type: none"> <li>'1' for Stretch</li> <li>'2' for Tile</li> <li>'3' for Center</li> </ul>	Text, Picture Objects
<b>property ImageAlpha: Integer</b>	Defines image transparency 0 to 255, where 00 is full transparency, and 255 is solid color	Text, Picture Objects
<b>property ImageFlip: Boolean</b>	Disables / Enables image flip	Text, Picture Objects
<b>procedure LoadMedia(const FileName: String)</b>	Loads the specified media file in the selected object.	DirectShow Media Source Objects
<b>property MediaPosX: Integer</b>	Defines the X position of the video in the selected object	DirectShow Media Source Objects
<b>property MediaPosY: Integer</b>	Defines the Y position of the video in the selected object	DirectShow Media Source Objects
<b>property MediaAlpha: Integer</b>	Defines the transparency for the selected object from 0 to 255, where 00 is full transparency, and 255 is solid color	DirectShow Media Source Objects

# TitleBox User Manual



TTObject Property/Procedure/Function	Description	Objects That Use It
<b>property MediaStyle: Integer</b>	Sets a style to the Media object from two options:  '0' for Stretch  '1' for Center	DirectShow Media Source Objects
<b>property MediaColor: Integer</b>	Defines the background color in case the object is in <Center> mode and media does not fit in the object (there is some empty space left).	DirectShow Media Source Objects
<b>property MediaQuality: Integer</b>	Sets video quality from the following options (note that '1' is the recommended value):  '0'for Pure  '1' for Low quality  '2'for Medium quality  '3' for High quality  '4'for Highest quality	DirectShow Media Source Objects
<b>property MediaUseOwnAlpha: Boolean</b>	Disables/ Enables the use of own ALPHA	DirectShow Media Source Objects
<b>property MediaUseAlpha: Boolean</b>	Disables / Enables the use of ALPHA	DirectShow Media Source Objects
<b>property MediaKeepAspect: Boolean</b>	Disables / Enables the <Keep aspect> option	DirectShow Media Source Objects
<b>property MediaAudioVolume: Integer</b>	Sets the volume for the media in decibels from 0 to 10,000, where 0 is the highest volume, and 10,000 is mute	DirectShow Media Source Objects
<b>property MediaAudioDelay: Integer</b>	Sets the delay value for the audio related to the video in milliseconds. This command is used to synchronize the audio with the video in the DXMedia object.	DirectShow Media Source Objects
<b>procedure Event(EventNo: Integer)</b>	Triggers user defined event specified by its number.	All
<b>property OnPlay: String</b>	Sets or gets a user defined task to an object to be executed on Play	All
<b>property OnStop: String</b>	Sets or gets a user defined task to an object to be executed on Stop	All
<b>property OnPause: String</b>	Sets or gets a user defined task to an object to be executed on Pause	All

# TitleBox User Manual



TTBObject Property/Procedure/Function	Description	Objects That Use It
<b>property OnAutoStop: String</b>	Sets or gets a user defined task to an object to be executed on AutoStop	All
<b>property OnLoop: String</b>	Sets or gets a user defined task to an object to be executed on Loop	All
<b>property OnEndInputTransition: String</b>	Sets or gets a user defined task to an object to be executed on End Input Transition	All
<b>property OnStartOutputTransition: String</b>	Sets or gets a user defined task to an object to be executed on Start Output Transition	All
<b>property OnDataReady: String</b>	Sets or gets a user defined task to an object to be executed on Data Ready	All
<b>property OnPropertychange: String</b>	Sets or gets a user defined task to an object to be executed on Property Change	All
<b>property OnTransform: String</b>	Sets or gets a user defined task to an object to be executed on Transform	All

## TTBSLIDE

TTBSlide Property/Procedure/Function	Description	Used By
<b>function Create(const Name: String): TTBSlide: Constructor</b>	Links to an existing slide in the project	Slides
<b>function CreateNew(const Name: String): TTBSlide: Constructor</b>	Creates a new slide in the project	Slides
<b>procedure Free</b>	Frees a TTBSlide object	Slides
<b>procedure Delete</b>	Removes currently selected slide and sets it free. All objects from the slide are moved to the Unassigned Slide.	Slides
<b>procedure ObjectList(List: TStrings)</b>	Get the list of all objects currently contained in the Slide	Slides

# TitleBox User Manual



<b>procedure Play</b>	Plays the specified slide (Jump to)	Slides
<b>procedure Stop</b>	Stops the specified slide	Slides
<b>property Index: Integer</b>	Sets or gets the index number of a slide	Slides
<b>property Name: String</b>	Sets or gets the name of a slide	Slides
<b>property BackColor: Integer</b>	Sets or gets the background color of a slide	Slides
<b>property Duration: Extended</b>	Sets or gets the duration of a slide in seconds	Slides
<b>property Enabled: Boolean</b>	Disables / Enables a slide	Slides
<b>property Lock: Boolean</b>	Locks / Unlocks a slide	Slides
<b>property PlayMode: TPlayMode</b>	Sets or gets a property of type TPlayMode to a slide, which can have the following values: pmAdd, pmClear, pmCrossPlay, pmStopAll, and pmStopPrevious	Slides
<b>property State: TObjectState</b>	Gets a property of type TObjectState, which can have the following values: osAutoStop, osPause, osPlay, and osStop	Slides

## TTBSLIDESMANAGER

TTBSlidesManager Property/Procedure/Function	Description	Used By
<b>procedure SlidesList(List: TStrings)</b>	Lists the names of all slides in a variable of type TStrings	Slides Manager
<b>function Count: Integer</b>	Returns the number of the slides in Slides Manager	Slides Manager
<b>procedure Play</b>	Plays the slideshow	Slides Manager
<b>procedure Stop</b>	Stops the slideshow	Slides Manager
<b>procedure Pause</b>	Pauses the slideshow	Slides Manager
<b>procedure Reset</b>	Initializes all buffers used by the slideshow	Slides Manager

# TitleBox User Manual



TTBSlidesManager Property/Procedure/Function	Description	Used By
<b>procedure Prev</b>	Plays the previous slide in the Slides Manager	Slides Manager
<b>procedure Next</b>	Plays the next slide in the Slides Manager	Slides Manager
<b>procedure Delete(Index: Integer)</b>	Removes the slide, designated by index number from the Slides Manager. All objects from the slide are moved to the Unassigned Slide.	Slides Manager
<b>procedure Move(OldIndex, NewIndex: Integer)</b>	Moves the currently selected slide to a new position by shifting the indexes of the other slides from the Slides Manager.	Slides Manager
<b>index property Slide(p0: String): Integer</b>	Returns the index number of a Slide, distinguished by its name	Slides Manager
<b>index property Items(p0: Integer): String</b>	Returns the name of an item (Slide), distinguished by its index number	Slides Manager

## TTBDATAPROVIDER

TTBDataProvider Property/Procedure/Function	Description	Used By
<b>function Create(const Name: String): TTBDataProvider: Constructor</b>	Links an object to a configured Data Provider	Data Providers
<b>procedure Free</b>	Frees a Data Provider	Data Providers
<b>procedure Delete</b>	Removes the Data Provider and sets it free	Data Providers
<b>procedure ObjectsList(List: TStrings)</b>	Lists the names of all objects, linked to a given Data Provider in a variable of type TStrings	Data Providers
<b>procedure Play</b>	Plays all objects, linked to a Data Provider	Data Providers
<b>procedure Stop</b>	Stops all objects, linked to a Data Provider	Data Providers
<b>procedure CursorStart</b>	Starts the cursor in a Data Provider	Data Providers
<b>procedure CursorStop</b>	Stops the cursor in a Data Provider	Data Providers



TTBDataProvider Property/Procedure/Function	Description	Used By
<b>procedure Top</b>	Moves the cursor to the top line in the Data Provider	Data Providers
<b>procedure Bottom</b>	Moves the cursor to the bottom line in the Data Provider	Data Providers
<b>procedure Prev</b>	Moves the cursor to the previous line in the Data Provider	Data Providers
<b>procedure Next</b>	Moves the cursor to the next line in the Data Provider	Data Providers
<b>procedure Scroll(Lines: Integer = 1)</b>	Moves the cursor with a given number of lines forward.	Data Providers
<b>procedure Update</b>	Refreshes the information, sent from the Data Provider	Data Providers
<b>procedure ColumnsList(List: TStrings)</b>	Lists the names of the columns from the Data Provider to a variable of type TStrings	Data Providers
<b>function LinkObject(Obj: TTBObject; DataColumn: String; Offset: Integer = 0; UseEffects: Boolean = False): Boolean</b>	Sets the properties of a link between an object and a Data Provider as in the <a href="#">Object Link Properties Dialog</a>	Data Providers
<b>procedure DeleteObjectLink(Obj: TTBObject)</b>	Removes a link between a Data Provider and an object	Data Providers
<b>function GetValue(Col, Row: Integer; var Data: Variant; var DataType: TDPDataType): Boolean</b>	Gets the value of a particular entry in a Data Provider, designated by its row and column number	Data Providers
<b>function GetColumn(Col: Integer; var DataType: TDPDataType): Variant</b>	Gets the whole column from a Data Provider	Data Providers
<b>property Distributor: TTBDataDistributor</b>	Sets a Data Distributor to a Data Provider	Data Providers
<b>property Name: String</b>	Sets the name of a Data Distributor	Data Providers
<b>property State: TObjectState</b>	Sets a variable of type TObjectState, which can have the following values: osAutoStop, osPause, osPlay, and osStop	Data Providers
<b>property ColCount: Integer</b>	Sets the number of columns in a particular Data Provider to an integer variable	Data Providers
<b>property RowCount: Integer</b>	Sets the number of rows in a particular Data Provider to an integer variable	Data Providers



TTBDataProvider Property/Procedure/Function	Description	Used By
<b>index property Columns(p0: Integer): String</b>	Sets the name of a column, distinguished by its number, to a string variable	Data Providers

## TTBDATADISTRIBUTOR

TTBDataDistributor Property/Procedure/Function	Description	Used By
<b>property Update: Integer</b>	Sets how often to update the source information from a Data Provider in seconds. If you enter '0', the updating will be Automatic	Data Distributors
<b>property Scroll: Integer</b>	Sets the time in seconds, by which to scroll the cursor in the Data Provider. If you enter '0' the scrolling will be Manual	Data Distributors
<b>property ScrollBy: Integer</b>	Sets the number of rows, by which the cursor scrolls within a Data Provider	Data Distributors
<b>property Boolean ResetOnDatasetChange:</b>	Disables / Enables the <a href="#">Reset On Dataset Change</a> property for the scroll options of the Data Distributor	Data Distributors
<b>property Boolean ResetOnFieldChange:</b>	Disables / Enables the <a href="#">Reset On Field Change</a> property for the scroll options of the Data Distributor	Data Distributors
<b>property Boolean StartCursorOnPlay:</b>	Disables / Enables the <a href="#">Start Cursor On Play</a> property for the scroll options of the Data Distributor	Data Distributors
<b>property Boolean NextSlideOnBottom:</b>	Disables / Enables the <a href="#">Next Slide On Bottom</a> property for the scroll options of the Data Distributor	Data Distributors
<b>property Boolean StopObjectsOnBottom:</b>	Disables / Enables the <a href="#">Stop Objects On Bottom</a> property for the scroll options of the Data Distributor	Data Distributors
<b>property Boolean StopObjectsOnAnyStop:</b>	Disables / Enables the <a href="#">Stop Objects On Any Stops</a> property for the scroll options of the Data Distributor	Data Distributors
<b>property Boolean StartObjectsOnAnyPlay:</b>	Disables / Enables the <a href="#">Start Objects On Any Play</a> property for the scroll options of the Data Distributor	Data Distributors
<b>property Boolean StartObjectsOnUpdate:</b>	Disables / Enables the <a href="#">Start Objects On Update</a> property for the scroll options of the Data Distributor	Data Distributors





## TTBPROJECT

TTBProject Property/Procedure/Function	Description	Used By
<b>procedure New</b>	Opens a new project	Projects
<b>procedure Start</b>	Starts the project	Projects
<b>procedure Stop</b>	Stops the project	Projects
<b>procedure Clear</b>	Clears the project	Projects
<b>procedure Close</b>	Closes TitleBox	Projects
<b>function Load(const TemplateName: String): Boolean</b>	Loads a project in TitleBox	Projects
<b>function Merge(const TemplateName: String): Boolean</b>	Merges a new project to the existing one	Projects
<b>procedure SaveAs(const TemplateName: String)</b>	Saves a project under a given name	Projects
<b>procedure NetStart</b>	Switches to Net Control mode of operation	Projects
<b>procedure NetStop</b>	Switches off Net Control mode of operation	Projects
<b>procedure StopAll</b>	Stops all objects in the project	Projects
<b>procedure MoveAll(IncX, IncY: Integer; ExcludeList: String = "")</b>	Moves all objects in the project. The ExcludeList variable is optional. If you enter a value for it, the objects, listed there will not be moved.	Projects
<b>procedure ObjectsList(List: TStrings)</b>	Gets the list of all objects currently loaded in TitleBox.	Projects
<b>procedure TagsList(Tag: Integer; List: TStrings)</b>	Gets the list of all objects with a given tag. If you enter a negative value, a list of all existing tags will be returned.	Projects
<b>procedure TasksList(List: TStrings)</b>	Gets the list of all tasks currently set in TitleBox.	Projects
<b>procedure DataProvidersList(List: TStrings)</b>	Gets the list of all Data Providers currently set in TitleBox.	Projects



TTBProject Property/Procedure/Function	Description	Used By
<b>property State: TObjectState</b>	Gets a variable of type TObjectState, which can have the following values: osPlay, and osStop	Projects
<b>property StateStr: String</b>	Gets the current state of the project in a string variable	Projects
<b>property Modified: Boolean</b>	Returns a true / false value, designating if the project has been modified since the last time it has been saved or opened	Projects
<b>property AspectRatio: Extended</b>	Gets the Aspect Ratio, in which the project is set	Projects
<b>property ResolutionX: Integer</b>	Gets the X resolution, in which the project is set	Projects
<b>property ResolutionY: Integer</b>	Gets the Y resolution, in which the project is set	Projects
<b>property Version: String</b>	Gets the last TitleBox version, in which the project has been saved	Projects

## TITLEBOX FUNCTIONS

Function	Description
<b>Execute(const CommandScript: String)</b>	Executes a script, written for the <a href="#">Command Script</a>
<b>LoadScript(const FileName: String): Boolean</b>	Loads any script, written for TitleBox. If you use the name of an existing script, it will be overwritten by the new script
<b>SetScript(const Name, Script: String)</b>	Saves a script with a given name. If you use the name of an existing script, it will be overwritten by the new script
<b>RunTask(const Task: String)</b>	Runs a task, saved in the <a href="#">Task Manager</a> , designated by its name
<b>TBVersion: String</b>	Returns the version number of the currently opened TitleBox
<b>Pause(Objects: array of String)</b>	Pauses an array of objects, designated by their names
<b>PauseList(List: TString)</b>	Pauses a list of objects
<b>Play(Objects: array of String)</b>	Plays an array of objects, designated by their names
<b>PlayList(List: TString)</b>	Plays a list of objects
<b>Stop(Objects: array of String)</b>	Stops an array of objects, designated by their names
<b>StopList(List: TString)</b>	Stops a list of objects
<b>Toggle(Objects: array of String)</b>	Toggles an array of objects, designated by their names
<b>ToggleList(List: TString)</b>	Toggles a list of objects



## TBITMAP32 (INHERITOR OF TOBJECT)

TBitmap32 Property/Procedure/Function	Description
<b>function Create: Constructor</b>	Creates a TBitmap32 item
<b>procedure Free</b>	Frees a TBitmap32 item
<b>function ClassName: String</b>	Returns the class name of an item in a String variable
<b>procedure Assign(Bmp32: TBitmap32)</b>	Assigns a Bmp32 object to a TBitmap32 item
<b>procedure AssignFrom(Bmp: TBitmap; Flip: Boolean = True)</b>	Assigns a Bmp32 object to the current TBitmap32 item. The image can be flipped.
<b>procedure AssignTo(Bmp: TBitmap; Flip: Boolean = True)</b>	Assigns the current TBitmap32 item to another Bmp32 object. The image can be flipped.
<b>procedure Draw(X, Y: Integer; Bmp32: TBitmap32; Mix: Boolean = False; Flip: Boolean = False)</b>	Draws a TBitmap32 item in given X and Y coordinates. The image can be flipped. It can also be mixed with the current image if it has a defined alpha value.
<b>procedure DrawFrom(X, Y: Integer; Bmp: TBitmap; Flip: Boolean = False)</b>	Copies a Bmp32 object in the current item in given X and Y coordinates. The image can be flipped.
<b>procedure DrawTo(X, Y: Integer; Bmp: TBitmap; Mix: Boolean = False; Flip: Boolean = False)</b>	Copies the current item in given X and Y coordinates to another Bmp32 object. The image can be flipped. It can also be mixed with the current image if it has a defined alpha value.
<b>procedure Fill(Color:TRGBA)</b>	Fills a TBitmap32 item with a given color
<b>procedure FillRect(Left, Top, Right, Bottom: Integer; Color: TRGBA; Mix: Boolean = False)</b>	Creates a Rectangle by giving it coordinates and fills it with a given color. The rectangle can be mixed with the current image if it has an alpha value.
<b>procedure LoadFromFile(const FileName: String)</b>	Loads a TBitmap32 item from a file
<b>procedure LoadFromStream(const S: TStream)</b>	Loads a TBitmap32 item from a stream
<b>procedure SaveToFile(const FileName: String)</b>	Saves a TBitmap32 item to a file
<b>procedure SaveToStream(const S: TStream)</b>	Saves a TBitmap32 item to a stream
<b>procedure SetSize(Width, Height: Integer; PixelFormat: TPixelFormat = pf32bit)</b>	Sets width, height, and pixel format, defined by a TPixelFormat class, to a TBitmap32 item



<b>procedure SetTransparent(Color: TRGBA; Alpha, NoAlpha: Integer)</b>	Searches in an object for a color by its RGB value and sets an alpha value to it in [Alpha], as well as an alpha value for the remaining part of the object, i.e., the one that does not contain the searched color, in [NoAlpha]
<b>property Alpha: Integer</b>	Defines image transparency 0 to 255, where 00 is full transparency, and 255 is solid color
<b>property Canvas: TCanvas</b>	Return the canvas object of class TCanvas for a TBitmap32 item
<b>property Force32bit: Boolean</b>	Formats an image to 32-bit
<b>property Height: Integer</b>	Gets/ sets the height of a TBitmap32 item
<b>property Pixel(p0: Integer; p1: Integer): TRGBA</b>	Gets / sets the pixel value at coordinates X, Y
<b>property PixelFormat: TPixelFormat</b>	Sets a pixel format, defined by a TPixelFormat class to a TBitmap32 item
<b>property Pixels: Variant</b>	Returns the address of the pixel buffer
<b>property ScanLine(p0: Integer): Pointer</b>	Returns the address for the first pixel in Line
<b>property Size: Integer</b>	Returns the pixel size of a TBitmap32 item
<b>property Width: Integer</b>	Gets / Sets the width of a TBitmap32 item

## TTBStorage

TTBStorage Property/Procedure/Function	Description
<b>procedure Clear</b>	Clears the TTBStorage class, i.e., removes all variables saved in it
<b>function Count: Integer</b>	Returns the number of variables saved TTBStorage
<b>index property Items(p0: Integer): Variant</b>	Gets / sets an item in TTBStorage, distinguished by its index

For a detailed description of all other classes, supported by the **TitleBox Program Script Engine**, please, visit:

<http://docwiki.embarcadero.com/VCL/en/Classes>

<http://docwiki.embarcadero.com/VCL/en/Controls>

<http://docwiki.embarcadero.com/VCL/en/Graphics>



## APPENDIX 6 – TitleBox Program Script Examples

Below you can find simple examples, presenting the basic **TitleBox Program Script** classes and functions, as well as some basic programming rules and organization to be used when programming with this new **TitleBox** feature. Each example is shown in all the four language syntaxes, supported by the **TitleBox Script Engine** and has a short explanation.

### Example 1

Pascal	C++
<pre> var    Obj: TTBObject;  begin    Project.Start;    Obj := TTBObject.Create('Text 1');    if Assigned(Obj) then    try      Obj.Play;    finally      Obj.Free;    end;    Obj := TTBObject.CreateNew('PICTURE', 'Pic1');    if Assigned(Obj) then    try      Obj.Left := 10;      Obj.Top := 10;      Obj.Width := 200;      Obj.Height := 200;      Obj.Play;           </pre>	<pre> {    Project.Start;    TTBObject Obj = new TTBObject("Text 1");    if(Obj)    try    {      Obj.Play;    }    finally    {      Obj.Free;    }    Obj = TTBObject.CreateNew("PICTURE", "Pic1");    if(Obj)    try    {      Obj.Left = 10;      Obj.Top = 10;      Obj.Width = 200;      Obj.Height = 200;           </pre>



```
finally
  Obj.Free;
end;
end.
```

```
Obj.Play;
}
finally
{
  delete Obj;
}
}
```

## Basic

```
Project.Start

Obj = new TTBOBJECT("Text 1")
if Assigned(Obj) then
  try
    Obj.Play
  finally
    Obj.Free
  end try
end if

Obj = TTBOBJECT.CreateNew("PICTURE", "Pic1")
if Assigned(Obj) then
  try
    Obj.Left = 10
    Obj.Top = 10
    Obj.Width = 200
    Obj.Height = 200
    Obj.Play
  finally
    delete Obj
```

## Java

```
Project.Start;

var Obj = new TTBOBJECT("Text 1")
if(Obj)
  try
    Obj.Play
  finally
    Obj.Free;

Obj = TTBOBJECT.CreateNew("PICTURE", "Pic1")
if(Obj)
  try
  {
    Obj.Left = 10
    Obj.Top = 10
    Obj.Width = 200
    Obj.Height = 200
    Obj.Play
  }
  finally
    delete Obj;
```



```
end try
```

```
end if
```

```
end
```



## Example 1 Explained:

1. Declaration of an object variable, named **Obj** of type **TTBObject** (in **Pascal** and **C++** only)
2. Starting the Project
3. Linking the **Obj** variable to an existing object, named **"Text 1"**
4. Checking if the **Obj** variable is actually linked to a **"Text 1"** object with an **if** statement
5. **Obj** variable is played and freed within a **try-finally** block
6. A new picture object is created in the work area and linked to the **Obj** variable
7. Check if the picture object is actually assigned for the **Obj** variable
8. The new picture object is given **X** and **Y** coordinates, as well as a **width** and **height** values in pixels and it is played and freed in a **try-finally** block

## Example 2

Pascal	C++
<pre>var   Obj: TTBObject;    St: TObjectState;  begin   Obj := TTBObject.Create('Text 1');    if Assigned(Obj) then   try     St := Obj.State;      if St = osPlay then       Obj.Stop     else       Obj.Play;    finally     Obj.Free;    end; end.</pre>	<pre>{   TTBObject Obj = TTBObject.Create("Text 1");    if (Assigned(Obj))   try   {     TObjectState St = Obj.State;      if (St == osPlay)       Obj.Stop;      else       Obj.Play;   }   finally   {     Obj.Free;   } }</pre>
Basic	Java





```
Obj = TTObject.Create("Text 1")

if Assigned(Obj) then

  try

    St = Obj.State

    if St = osPlay then Obj.Stop

    if St = osStop then Obj.Play

  finally

    Obj.Free

  end try

end if

end
```

```
var Obj = TTObject.Create("Text 1");

if (Assigned(Obj))

  try

  {

    var St = Obj.State

    if (St == osPlay)

      Obj.Stop

    else

      Obj.Play;

    }

  finally

    Obj.Free;
```

## Example 2 Explained:

1. Declaration of a variable of type **TTObject**, named **Obj** and a variable of type **TObjectState**, named **St** (in **Pascal** and **C++** only)
2. Linking the **Obj** variable to an existing object, named **"Text 1"**
3. Check if the **Obj** variable is actually linked to a **"Text 1"** object with an **if** statement
4. Setting a value to the **St** variable, equal to the state of the **Obj** variable in a **try-finally** block
5. Introducing an **if** statement, which checks if the **"Text 1"** is in state play. If yes, the **Obj** variable is stopped. If no, the **Obj** variable is played.
6. The **Obj** variable is freed.



## Example 3

Pascal	C++
<pre>var   List: TStringList;    I: Integer;    Obj: TTBObject;    S1, S2: String;  begin   List := TStringList.Create;    try     Project.ObjectsList(List);      for I := 0 to List.Count - 1 do       begin         Obj := TTBObject.Create(List[I]);          if Assigned(Obj) then           try             S1 := Obj.Name;              if Obj.State = osPlay then               S2 := 'Play'             else               S2 := 'Stop';            finally             Obj.Free;           end;          ShowMessage(Format('%s: State = %s', [S1, S2]));        end;     end;    finally     ShowMessage(Format('%s: State = %s', [S1, S2]));   end; end;</pre>	<pre>{   TStringList List = new TStringList;    try   {     Project.ObjectsList(List);      for(int i = 0; i &lt; List.Count; i++)     {       TTBObject Obj = TTBObject.Create(List[i]);        if (Assigned(Obj))         try         {           char S1 = Obj.Name;            char S2;            if (Obj.State == osPlay)             S2 = "Play";           else             S2 = "Stop";         }        finally       {         Obj.Free;       }        ShowMessage(Format("%s: State = %s", [S1, S2]));     }   }    finally</pre>



```
List.Free;  
  
end;  
  
end.
```

## Basic

```
List = new TStringList  
  
try  
  
    Project.ObjectsList(List)  
  
    for i = 0 to List.Count - 1  
  
        Obj = TTBObject.Create(List[i])  
  
        if Assigned(Obj) then  
  
            try  
  
                S1 = Obj.Name  
  
                if Obj.State = osPlay then  
  
                    S2 = "Play"  
  
                else  
  
                    S2 = "Stop"  
  
                end if  
  
            finally  
  
                Obj.Free  
  
            end try  
  
            ShowMessage(Format("%s: State = %s", [S1, S2]))  
  
        end if  
  
    next i for i  
  
finally  
  
    List.Free  
  
end try  
  
end
```

```
{  
  
    List.Free;  
  
}  
  
}
```

## Java

```
var List = new TStringList  
  
try  
  
    {  
  
        Project.ObjectsList(List);  
  
        for(var i = 0; i < List.Count; i++)  
  
            {  
  
                var Obj = TTBObject.Create(List[i]);  
  
                if (Assigned(Obj))  
  
                    try  
  
                        {  
  
                            var S1 = Obj.Name;  
  
                            var S2;  
  
                            if (Obj.State == osPlay)  
  
                                S2 = "Play"  
  
                            else  
  
                                S2 = "Stop";  
  
                        }  
  
                    finally  
  
                        Obj.Free;  
  
                }  
  
            ShowMessage(Format("%s: State = %s", [S1, S2]));  
  
            }  
  
        }  
  
finally
```



```
List.Free;
```

## Example 3 Explained:

1. Declaration of a variable named **List** of type **TStringList**, an integer variable named **I**, a variable named **Obj** of type **TTBObject**, and two string variables, named **S1** and **S2** (in **Pascal** and **C++** only).
2. Creating the **List** variable
3. Assigning the names of all the objects from the project in the **List** variable
4. Initialization of a **try-finally** block
5. Initialization of a **for** loop, walking through the whole list of names.
6. Linking the **Obj** variable to the corresponding object from the **List** variable (with number **I**).
7. Checking if an object is actually assigned to the **Obj** variable with an **if** statement
8. Assigning a value to the **S1** variable, equal to the current name of the **Obj** variable, i.e., the name of the object with number **I** in the **List**.
9. Checking the state of the **Obj** variable (play or stop) with an **if** block
10. Assigning the state of the **Obj** variable to the **S2** variable (play or stop)
11. The **Obj** variable is freed.
12. Showing the current values of **S1** and **S2** within the loop in a Message Dialog
13. Finishing the loop and setting the **List** variable free

## Example 4

Pascal	C++
<pre>var   Slide: TTBSlide;   List: TStringList;   I: Integer;   S: String;  begin   List := TStringList.Create;    try     Slide := TTBSlide.Create('Slide 1');      if Assigned(Slide) then        try          Slide.Play;          Slide.ObjectsList(List);          S := 'Objects in Slide ' + Slide.Name + ' ';          for I := 0 to List.Count - 1 do</pre>	<pre>{   TStringList List = new TStringList;    try   {     TTBSlide Slide = TTBSlide.Create("Slide 1");      if (Assigned(Slide))        try          {           Slide.Play;            Slide.ObjectsList(List);            char S = "Objects in Slide " + Slide.Name + " ";            for(int i = 0; i &lt; List.Count; i++)              S = S + List[i] + " ";            ShowMessage(S);          }        finally</pre>



```
S := S + List[] + ', ';  
  
ShowMessage(S);  
  
finally  
  
Slide.Free;  
  
end;  
  
finally  
  
List.Free;  
  
end;  
  
end.
```

## Basic

```
List = new TStringList  
  
try  
  
Slide = TTBSlide.Create("Slide 1")  
  
if Assigned(Slide) then  
  
try  
  
Slide.Play  
  
Slide.ObjectsList(List)  
  
S = "Objects in Slide " + Slide.Name + ": "  
  
for i = 0 to List.Count - 1  
  
S = S + List[i] + ", "  
  
next i for i  
  
ShowMessage(S)  
  
finally  
  
Slide.Free  
  
end try  
  
end if  
  
finally  
  
List.Free  
  
end try
```

```
{  
  
Slide.Free;  
  
}  
  
}  
  
finally  
  
{  
  
List.Free;  
  
}  
  
}
```

## Java

```
var List = new TStringList  
  
try  
  
{  
  
var Slide = TTBSlide.Create("Slide 1")  
  
if (Assigned(Slide))  
  
try  
  
Slide.Play  
  
Slide.ObjectsList(List)  
  
var S = "Objects in Slide " + Slide.Name + ": "  
  
for(var i = 0; i < List.Count; i++)  
  
S = S + List[i] + ", "  
  
ShowMessage(S)  
  
}  
  
finally  
  
Slide.Free;  
  
}  
  
finally  
  
List.Free;
```



end

## Example 4 Explained:

1. Declaration of a variable, named **Slide** with a type **TTBSlide**, a variable named **List** of type **TStringList**, an integer variable, named **I**, and a string variable, named **S** (in **Pascal** and **C++** only)
2. Creating the **List** variable
3. Creating the **Slide** variable by linking it to "**Slide 1**" in a **try-finally** block
4. Checking if a slide is actually assigned to the **Slide** variable
5. Playing the **Slide**
6. Assigning the objects' names from **Slide** to the **List** variable
7. Assigning text to the **S** variable, including "Objects in Slide" and the name of the **Slide**, assigned to the **Slide** variable
8. Adding the name of each object from the **List** variable to the **S** variable in a **for** loop
9. Showing the current value of the **S** variable in a Message Dialog
10. The **Slide** variable is freed
11. The **List** variable is freed



## Example 5

Pascal	C++
<pre>var   Obj: TTBObject;  begin   Obj := TTBObject.Create("Text 1");    if Assigned(Obj) then   try     Obj.TextSelect(1, 3);      Obj.FontSize := 50;      Obj.FontStyle := [fsBold, fsItalic];      Obj.FontName := 'Arial';      Obj.UpdateParams;    finally     Obj.Free;    end;  end.</pre>	<pre>{   TTBObject Obj = TTBObject.Create("Text 1");    if (Assigned(Obj))   try   {     Obj.TextSelect(1, 3);      Obj.FontSize = 50;      Obj.FontStyle = [fsBold, fsItalic];      Obj.FontName = "Arial";      Obj.UpdateParams;    }    finally   {     Obj.Free;    }  }</pre>
Basic	Java
<pre>Obj = TTBObject.Create("Text 1")  if Assigned(Obj) then try   Obj.TextSelect(1, 3)    Obj.FontSize = 50    Obj.FontStyle = [fsBold, fsItalic]    Obj.FontName = "Arial"    Obj.UpdateParams  finally   Obj.Free  end try</pre>	<pre>var Obj = TTBObject.Create("Text 1")  if (Assigned(Obj)) try {   Obj.TextSelect(1, 3)    Obj.FontSize = 50    Obj.FontStyle = [fsBold, fsItalic]    Obj.FontName = "Arial"    Obj.UpdateParams  }  finally</pre>



```
end if
```

```
end
```

```
Obj.Free
```

## Example 5 Explained:

1. Declaring a variable named **Obj** of type **TTBObject**
2. Linking the **Obj** variable to an already created text object, named "Text 1"
3. Checking if an object is actually assigned to the **Obj** variable
4. Initialising a **try-finally** block
5. Selecting the first three symbols from the text object and setting their size, font and font style
6. Updating the parameters of the **Obj** variable so that the new text formatting is applied
7. The **Obj** variable is freed





## Example 6

Pascal	C++
<pre>var   DP: TTBDataProvider;   List: TStringList;  begin   DP := TTBDataProvider.Create("FileLink Data Provider Text04");   if Assigned(DP) then   try     DP.CursorStop;     DP.Top;      List := TStringList.Create;   try     DP.ObjectsList(List);     PlayList(List);      DP.Next;     DP.Scroll(2);      StopList(List);   finally     List.Free;   end;   finally     DP.Free;   end; end.</pre>	<pre>{   TTBDataProvider DP = TTBDataProvider.Create("FileLink Data Provider Text04");   if (Assigned(DP))   try   {     DP.CursorStop;     DP.Top;      TStringList List = new TStringList;   try   {     DP.ObjectsList(List);     PlayList(List);      DP.Next;     DP.Scroll(2);      StopList(List);   }   finally   {     List.Free;   }   }   finally   {     DP.Free;   } }</pre>



Basic	Java
<pre>DP = TTBDataProvider.Create("FileLink Data Provider Text04")  if Assigned(DP) then  try      DP.CursorStop      DP.Top      List = new TStringList      try          DP.ObjectsList(List)          PlayList(List)          DP.Next          DP.Scroll(2)          StopList(List)      finally          List.Free      end try  finally      DP.Free  end try  end if  end</pre>	<pre>}  var DP = TTBDataProvider.Create("FileLink Data Provider Text04")  if (Assigned(DP))  try  {      DP.CursorStop      DP.Top      var List = new TStringList      try      {          DP.ObjectsList(List)          PlayList(List)          DP.Next          DP.Scroll(2)          StopList(List)      }      finally          List.Free;      }  finally      DP.Free;</pre>

## Example 6 Explained:

1. Declaration of a variable, named **DP** of type **TTBDataProvider** and a variable, named **List** of type **TStringList** (in **Pascal** and **C++** only). The Data Provider input is already linked to file "**Text 4.txt**" shown [below](#).
2. Checking if the **DP** variable is actually linked to the existing data provider
3. Initialising a **try-finally** block



4. Stopping the cursor scrolling within the linked text file
5. Moving the cursor to the top line of the linked text file
6. Creating the **List** variable
7. Initialising another **try-finally** block
8. Returning a list of all objects, linked to the Data Provider in the **List** variable
9. Playing all the objects in the **List** variable
10. Scrolling the cursor to the next line in the Data Provider
11. Scrolling the cursor with two lines in the Data Provider
12. Stopping all the objects in the **List** variable
13. The **List** and **DP** variables are freed

## Example 7

Pascal	C++
<pre>var   L: TStringList;   O: TTBObject;   DP: TTBDataProvider;  begin   DP := TTBDataProvider.Create('FileLink Data Provider Text04');    if Assigned(DP) then   try     L := TStringList.Create;      try       DP.ColumnsList(L);        O := TTBObject.Create('Text 1');        try         DP.LinkObject(O, L[1]);        finally         O.Free;        end;      finally       L.Free;      end;    end;</pre>	<pre>{   TTBDataProvider DP = TTBDataProvider.Create("FileLink Data Provider Text04");   if (Assigned(DP))   try   {     TStringList L = new TStringList;      try     {       DP.ColumnsList(L);        TTBObject O = TTBObject.Create("Text 1");        try       {         DP.LinkObject(O, L[1]);        }        finally       {         O.Free;        }      }    }    finally   {</pre>



```
DP.Distributor.Scroll := 2; // sec  
  
DP.Distributor.ScrollBy := 1; // row  
  
DP.CursorStart;  
  
DP.Play;  
  
finally  
  
DP.Free;  
  
end;  
  
Project.Start;  
  
end.
```

## Basic

```
DP = TTBDataProvider.Create("FileLink Data Provider Text04")  
  
if Assigned(DP) then  
  
try  
  
List = new TStringList  
  
try  
  
DP.ColumnsList(List)  
  
O = TTBObject.Create("Text 1")  
  
try  
  
DP.LinkObject(O, List[1])  
  
finally  
  
O.Free  
  
end try  
  
finally
```

```
L.Free;  
  
}  
  
DP.Distributor.Scroll = 2; // sec  
  
DP.Distributor.ScrollBy = 1; // row  
  
DP.CursorStart;  
  
DP.Play;  
  
}  
  
finally  
  
{  
  
DP.Free;  
  
}  
  
Project.Start;  
  
}
```

## Java

```
var DP = TTBDataProvider.Create("FileLink Data Provider Text04")  
  
if (Assigned(DP))  
  
try  
  
{  
  
var L = new TStringList  
  
try  
  
{  
  
DP.ColumnsList(L);  
  
var O = TTBObject.Create("Text 1")  
  
try  
  
DP.LinkObject(O, L[1])  
  
finally  
  
O.Free;
```



<pre>List.Free  end try  DP.Distributor.Scroll = 2 'sec  DP.Distributor.ScrollBy = 1 'row  DP.CursorStart  DP.Play  finally  DP.Free  end try  Project.Start  end if  end</pre>	<pre>}  finally  L.Free;  DP.Distributor.Scroll = 2 // sec  DP.Distributor.ScrollBy = 1 // row  DP.CursorStart  DP.Play  }  finally  DP.Free;  Project.Start</pre>
---	--

## Example 7 Explained:

1. Declaration of a variable, named **L** of type **TStringList**, a variable, named **O** of type **TTBObject**, and a variable, named **DP** of type **TTBDataProvider** (in **Pascal** and **C++** only). The Data Provider input is already linked to file "**Text 4.txt**" shown [below](#).
2. Checking if **DP** variable is actually linked to the existing data provider
3. Initialising a **try-finally** block
4. Creating the **L** variable
5. Adding the names of the columns in the Data Provider to the **L** variable
6. Linking the **O** variable to an existing text object, named "**Text 1**"
7. Initialising another **try-finally** block
8. Setting the link between the data provider and the text object to show data from column with index '1' from the Data Provider, i.e., the second column
9. Setting the **O** variable and the **L** variable free
10. Setting the Data Distributor for the Data Provider to scroll every 2 seconds
11. Setting the Data Distributor for the Data Provider to scroll by 1 line
12. Starting the cursor movement in the source file
13. Playing all the objects, linked to the Data Provider
14. The **DP** variable is freed
15. Starting the project

## Example 8

Pascal

C++



```
var
  DP: TTBDataProvider;
  I, Rows, Cols: Integer;
  ColumnValues: Variant;
  DT: TDPDataType;
  S: String;

begin
  DP := TTBDataProvider.Create('FileLink Data Provider Text04');
  if Assigned(DP) then
    try
      Rows := DP.RowCount;
      Cols := DP.ColCount;

      S := '';

      ColumnValues := DP.GetColumn(0, DT);
      if DT = dtText then
        for I := 0 to Rows - 1 do
          S := S + VarArrayElement(ColumnValues, I) + #13#10;

        ShowMessage(S);

      ColumnValues := null;
    finally
      DP.Free;
    end;
  end.
```

## Basic

```
DP = TTBDataProvider.Create('FileLink Data Provider Text04')
```

```
{
  TTBDataProvider DP = TTBDataProvider.Create("FileLink Data Provider Text04");
  if (Assigned(DP))
    try
      {
        int Rows = DP.RowCount;
        int Cols = DP.ColCount;

        char S = "";

        TDPDataType DT;

        variant ColumnValues = DP.GetColumn(0, DT);
        if (DT == dtText)
          {
            for(int I = 0; I < Rows; I++)
              S = S + VarArrayElement(ColumnValues, I) + #13#10;
          }
        ShowMessage(S);

        ColumnValues = null;
      }
    finally
      {
        DP.Free;
      }
  }
}
```

## Java

```
var DP = TTBDataProvider.Create("FileLink Data Provider Text04");
```



```
if Assigned(DP) then
try
    Rows = DP.RowCount
    Cols = DP.ColCount
    S = ""
    ColumnValues = DP.GetColumn(0, DT)
    if DT = dtText then
        for I = 0 to Rows - 1
            S = S + VarArrayElement(ColumnValues, I) + #13#10
        next
    end if
    ShowMessage(S)

    ColumnValues = null
finally
    DP.Free
end try
end if

end
```

```
if (Assigned(DP))
try
{
    var Rows = DP.RowCount
    var Cols = DP.ColCount
    var S = ""
    var ColumnValues = DP.GetColumn(0, DT)
    if (DT == dtText)
    {
        for(I = 0; I < Rows; I++)
            S = S + VarArrayElement(ColumnValues, I) + #13#10;
    }
    ShowMessage(S)

    ColumnValues = null;
}
finally
    DP.Free;
```

## Example 8 Explained:

1. Declaration of a variable, named **DP** of type **TTBDataProvider**, three integer variables, named **I**, **Rows**, and **Cols**, a variant variable, named **ColumnValues**, a variable, named **DT** of type **TDPDataType**, and a **String** variable, named **S** (in **Pascal** and **C++** only). The Data Provider input is already linked to file "**Text 4.txt**" shown [below](#).
2. Checking if **DP** variable is actually linked to the existing data provider
3. Initialising a **try-finally** block
4. Returning the number of rows in the Data Provider in the **Rows** variable
5. Returning the number of columns in the Data Provider in the **Cols** variable
6. Setting a value of '' to the string variable
7. Setting the values from column 1, i.e., the column with index 0 to the **ColumnValues** variable
8. Checking if the data type in the abovementioned column is of type **dtText**, i.e., text data
9. Initialising a for loop, covering all the rows in the column and adding their values to the **S** variable one by one
10. Showing the value of **S** in a message dialog
11. Setting a null value to the **ColumnValues** variable to free memory
12. The **DP** variable is freed



## Example 9

Pascal	C++
<pre>var   Slide: TTBSlide;   List: TStringList;  begin   List := TStringList.Create;    try     Slides.SlidesList(List);      Slide := TTBSlide.Create(List[0]);     if Assigned(Slide) then       try         Slide.Play;         Slides.Move(Slide.Index, List.Count);       finally         Slide.Free;       end;     finally       List.Free;     end;   end. end.</pre>	<pre>{   TStringList List = new TStringList;    try   {     Slides.SlidesList(List);      TTBSlide Slide = TTBSlide.Create(List[0]);     if (Assigned(Slide))       try       {         Slide.Play;         Slides.Move(Slide.Index, List.Count);       }     finally     {       Slide.Free;     }   }   finally   {     List.Free;   } }</pre>
Basic	Java
<pre>List = new TStringList  try    Slides.SlidesList(List)</pre>	<pre>var List = new TStringList  try  {    Slides.SlidesList(List)</pre>





```
Slide = TTBSlide.Create(List[0])

if Assigned(Slide) then

try

    Slide.Play

    Slides.Move(Slide.Index, List.Count)

finally

    Slide.Free

end try

end if

finally

    List.Free

end try

end
```

```
var Slide = TTBSlide.Create(List[0]);

if (Assigned(Slide))

try

{

    Slide.Play;

    Slides.Move(Slide.Index, List.Count);

}

finally

    Slide.Free;

}

finally

    List.Free;
```

## Example 9 Explained:

1. Declaration of a variable, named **Slide** of type **TTBSlide** and a variable, named **List** of type **TStringList** (in **Pascal** and **C++** only)
2. Creating the **List** variable.
3. Initialising a **try-finally** block
4. Adding the names of the current slides in the project to the **Slides** variable
5. Linking the **Slide** variable to the first slide, i.e. the slide with an index of '0'
6. Checking if a slide is actually linked to the **Slide** variable
7. Initialising another **try-finally** block
8. Playing the **Slide**
9. Changing the index of the slide to the index of the last slide in the **SlidesList**, i.e., moving it at the end of the slide list
10. The **Slide** and the **List** variables are freed



## Example 10

Pascal	C++
<pre>var   Obj: TTBObject;  begin   Project.New;    Obj := TTBObject.CreateNew('TEXT', 'Text 1');    if Assigned(Obj) then   try     Obj.Left := 20;      Obj.Top := 30;      Obj.Width := 300;      Obj.Height := 100;      Obj.Text := 'Test Object';      Obj.Play;    finally     Obj.Free;   end;    Project.Start;  end.</pre>	<pre>{   Project.New;    TTBObject Obj = TTBObject.CreateNew("TEXT", "Text 1");    if (Assigned(Obj))   try   {     Obj.Left = 20;      Obj.Top = 30;      Obj.Width = 300;      Obj.Height = 100;      Obj.Text = "Test Object";      Obj.Play;   }   finally   {     Obj.Free;   }    Project.Start; }</pre>
Basic	Java
<pre>Project.New  Obj = TTBObject.CreateNew("TEXT", "Text 1")  if Assigned(Obj) then try   Obj.Left = 20</pre>	<pre>Project.New;  var Obj = TTBObject.CreateNew("TEXT", "Text 1")  if (Assigned(Obj)) try {</pre>



<pre>Obj.Top = 30  Obj.Width = 300  Obj.Height = 100  Obj.Text = "Test Object"  Obj.Play  finally    Obj.Free  end try  end if  Project.Start  end</pre>	<pre>Obj.Left = 20  Obj.Top = 30  Obj.Width = 300  Obj.Height = 100  Obj.Text = "Test Object"  Obj.Play }  finally    Obj.Free;  Project.Start;</pre>
--	---

## Example 10 Explained:

1. Declaration of a variable, named **Obj** of type **TTBObject** (in **Pascal** and **C++** only)
2. Opening a new project.
3. Creating a new text object within the project, named "**Text 1**" and linking it to the **Obj** variable
4. Checking if an object is actually linked to the **Obj** variable
5. Initialising a **try-finally** block
6. Assigning **X** and **Y** coordinates to the text object, as well as width and height
7. Entering the text "**Test Object**" in the object
8. Playing the object
9. Setting the **Obj** variable free
10. Starting the project



## Example 11

Pascal	C++
<pre>var   I: Integer;  begin   I := 0;    try     I := 5 div I;    except      ShowMessage(ExceptionMessage);    end;    ShowMessage(I);  end.</pre>	<pre>{   int I = 0;    try     { I = 5 % I; }    catch      {       I = -1;        ShowMessage(ExceptionMessage);     }    ShowMessage(I); }</pre>
Basic	Java
<pre>dim I = 0  try    I = 5 / I  catch    I = -1    ShowMessage(ExceptionMessage)  end try  ShowMessage(I)</pre>	<pre>var I = 0  try    I = 5 / I  Catch    {     I = -1      ShowMessage(ExceptionMessage)    }  ShowMessage(I)</pre>

## Example 11 Explained:

1. Declaration of an integer variable, named **I**(in **Pascal** and **C++** only)
2. Setting a value of '0' to the **I** variable
3. Initialising a **try-except / try-catch** block
4. Attempting to divide '5' by the **I** variable



5. Setting a value of '-1' to the I variable in case there is an error between the **"try"** and **"except"** statements and shows a message with a description of the type of error that has occurred. In this case the step will not be executed, since I is equal to '0' and you cannot divide by '0'.
6. Showing the value of I in a message dialog



## Example 12

Pascal	C++
<pre>function Transform(Obj: TTBObject; Text: String): Boolean;  var DP: TTBDataProvider;  begin  if Text &lt;&gt; "" then      Result := False  else  begin      DP := TTBDataProvider.Create("FileLink Data Provider Text04");      DP.Next;      DP.Free;      Result := True;  end;  end;  begin  end.</pre>	<pre>bool Transform(TTBObject Obj, char Text)  {  if (Text != "")      return(False);  else  {      TTBDataProvider DP = TTBDataProvider.Create("FileLink Data Provider Text04");      DP.Next;      DP.Free;      return(True);  }  }  {  }  }</pre>
Basic	Java
<pre>function Transform(Obj, Text)  if Text &lt;&gt; "" then      return False  else      DP = TTBDataProvider.Create("FileLink Data Provider Text04")      DP.Next      DP.Free      return True  end if  end function</pre>	<pre>function Transform(Obj, Text);  {  if (Text != "")      return(False)  else  {      var DP = TTBDataProvider.Create("FileLink Data Provider Text04")      DP.Next      DP.Free      return(True);  }  }</pre>



```
end
}
```

## Example 12 Explained:

1. Declaration of a **Transform Function**
2. Declaration of a variable, named **DP** of type **TTBDataProvider**(in **Pascal** and **C++** only)
3. Checking if the Text in the object, to which the transform function is linked is different from ' ', i.e., if there is an empty line.
4. If an empty line is not found, returning a **False** result to the **Transform Function**, i.e., the object will display information as it is fed by the Data Provider
5. If text is not different from ' ', i.e., an empty line is found, the **DP** variable is linked to the \*.txt file Text4, shown [below](#).  
The cursor is moved to the next line in the **Text 4** file
6. The **DP** variable is set free
7. A **True** result is returned to the **Transform Function**, i.e., the object has already shown the proper data

## Example 13

Pascal	C++
<pre>begin   if DEBUG then     ShowMessage("Debug!")   else     ShowMessage("Executing!");   end. end.</pre>	<pre>{   if(DEBUG)     ShowMessage("Debug!");   else     ShowMessage("Executing!"); }</pre>
Basic	Java
<pre>if DEBUG then   ShowMessage("Debug!") else   ShowMessage("Executing!") end if end</pre>	<pre>if(DEBUG)   ShowMessage("Debug!") else   ShowMessage("Executing!");</pre>

## Example 13 Explained:

1. Checking if TitleBox is in debug mode
2. If yes, showing a message with text "Debug!"



3. If no, showing a message with text "Executing!"





## Example 14

Pascal	C++
<pre>var   S: String;  begin   S := 'Stored Value';    if Storage[0] &lt;&gt; 500 then   begin     Storage[0] := 500;      Storage[1] := S;   end;    ShowMessage(Storage.Count);    if Storage.Count &gt; 10 then     Storage.Clear;   end. end.</pre>	<pre>{   char S = "Stored Value";    if(Storage[0] != 500)   {     Storage[0] = 500;      Storage[1] = S;   }    ShowMessage(Storage.Count);    if(Storage.Count &gt; 10)     Storage.Clear; }</pre>
Basic	Java
<pre>dim S = "Stored Value"  if Storage[0] &lt;&gt; 500 then    Storage[0] = 500    Storage[1] = S  end if  ShowMessage(Storage.Count)  if Storage.Count &gt; 10 then    Storage.Clear  end if  end</pre>	<pre>var S = "Stored Value"  if(Storage[0] != 500) {   Storage[0] = 500    Storage[1] = S }  ShowMessage(Storage.Count)  if(Storage.Count &gt; 10)    Storage.Clear</pre>

## Example 14 Explained:

1. Declaration of a string variable, named **S** (in **Pascal** and **C++** only)
2. Setting a value, equal to "**Stored Value**" to **S**.



3. Initialization of an **if** statement for checking whether the first item in the **Storage** is different from 500. If it is different, assigning a value of 500 to the first item in the **Storage** and a value, equal to **S** to the second item in the **Storage**.
4. Ending the **if** statement.
5. Showing the number of items stored in a Message dialog.
6. Clearing the **Storage** if the number of items, stored is greater than 10.

## Example 15

Pascal	C++
<pre>uses 'Unit.pas', 'Test.cpp', 'String.vb', 'Script.js';  begin ... end.</pre>	<pre>#include "Unit.pas", "Test.cpp", "String.vb", "Script.js"  { ... }</pre>
Basic	Java
<pre>imports "Unit.pas", "Test.cpp", "String.vb", "Script.js"  ...  end</pre>	<pre>import "Unit.pas", "Test.cpp", "String.vb", "Script.js"  { ... }</pre>

## Example 15 Explained:

1. Assigning the following saved modules to be used in the script: **"Unit.pas"**, **"Test.cpp"**, **"String.vb"**, and **"Script.js"**. Notice that the four files are written in different script syntaxes.
2. Initialising and ending the main part of the script.

# TitleBox User Manual



## Test 4.txt

Header Col_1	Header Col_2	Header Col_3
Line 1 Col 1	Line 1 Col 2	Line 1 Col 3
Line 2 Col 1	Line 2 Col 2	Line 2 Col 3
Line 3 Col 1	Line 3 Col 2	Line 3 Col 3
Line 4 Col 1	Line 4 Col 2	Line 4 Col 3
Line 5 Col 1	Line 5 Col 2	Line 5 Col 3
Line 6 Col 1	Line 6 Col 2	Line 6 Col 3
Line 7 Col 1	Line 7 Col 2	Line 7 Col 3
Line 8 Col 1	Line 8 Col 2	Line 8 Col 3
Line 9 Col 1	Line 9 Col 2	Line 9 Col 3
Line 10 Col 1	Line 10 Col 2	Line 10 Col 3
Line 11 Col 1	Line 11 Col 2	Line 11 Col 3
Line 12 Col 1	Line 12 Col 2	Line 12 Col 3
Line 13 Col 1	Line 13 Col 2	Line 13 Col 3
Line 14 Col 1	Line 14 Col 2	Line 14 Col 3
Line 15 Col 1	Line 15 Col 2	Line 15 Col 3
Line 16 Col 1	Line 16 Col 2	Line 16 Col 3
Line 17 Col 1	Line 17 Col 2	Line 17 Col 3
Line 18 Col 1	Line 18 Col 2	Line 18 Col 3
Line 19 Col 1	Line 19 Col 2	Line 19 Col 3
Line 20 Col 1	Line 20 Col 2	Line 20 Col 3
Line 21 Col 1	Line 21 Col 2	Line 21 Col 3
Line 22 Col 1	Line 22 Col 2	Line 22 Col 3
Line 23 Col 1	Line 23 Col 2	Line 23 Col 3
Line 24 Col 1	Line 24 Col 2	Line 24 Col 3
Line 25 Col 1	Line 25 Col 2	Line 25 Col 3
Line 26 Col 1	Line 26 Col 2	Line 26 Col 3
Line 27 Col 1	Line 27 Col 2	Line 27 Col 3
Line 28 Col 1	Line 28 Col 2	Line 28 Col 3
Line 29 Col 1	Line 29 Col 2	Line 29 Col 3
Line 30 Col 1	Line 30 Col 2	Line 30 Col 3

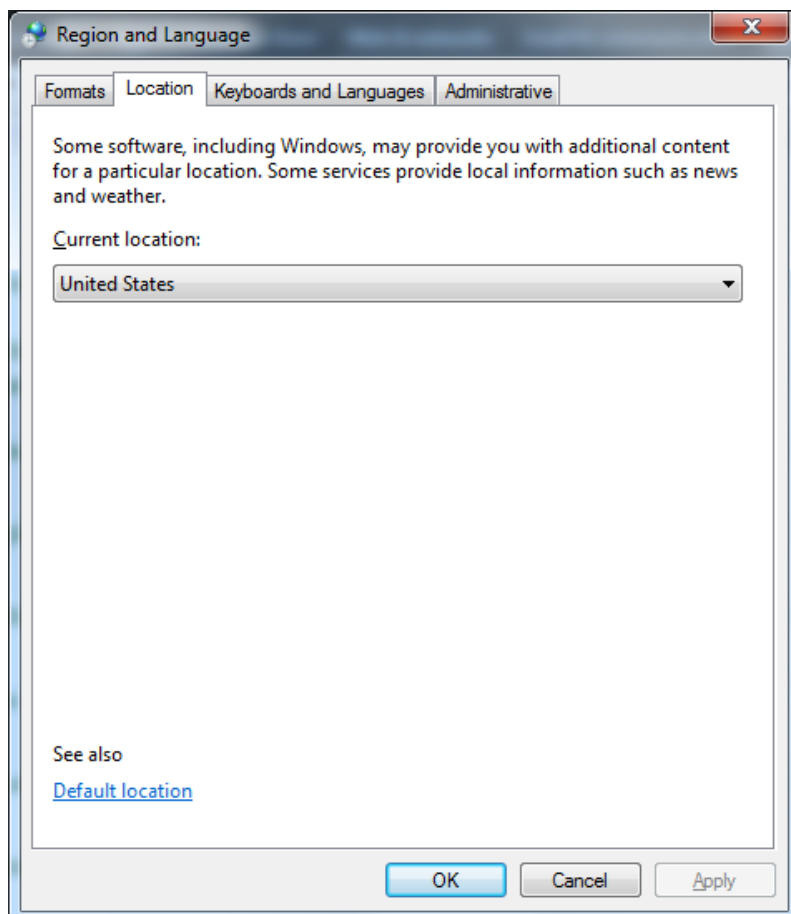


## APPENDIX 7 – Windows Configurations for Using Non-English Object Names in AirBox – TitleBox Net Control Mode

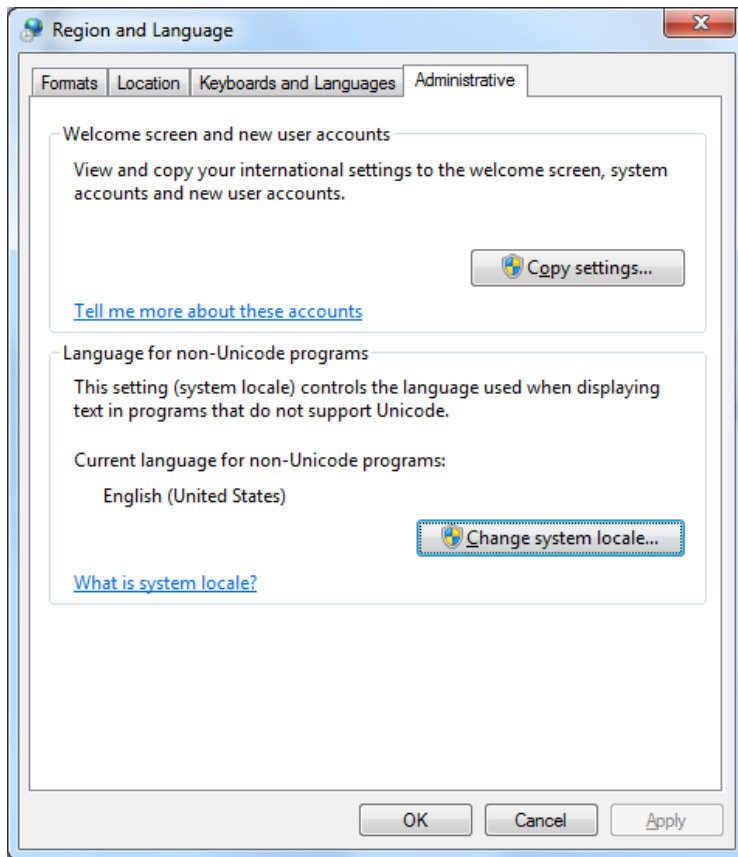
The following instruction is applicable for users who work with non-English languages for TitleBox objects and use TitleBox together with AirBox. Execute the steps, provided below and you will be able to use TitleBox objects in your target language.

NOTE: If you are using TitleBox and AirBox on different machines, make sure to make the following configurations on both machines.

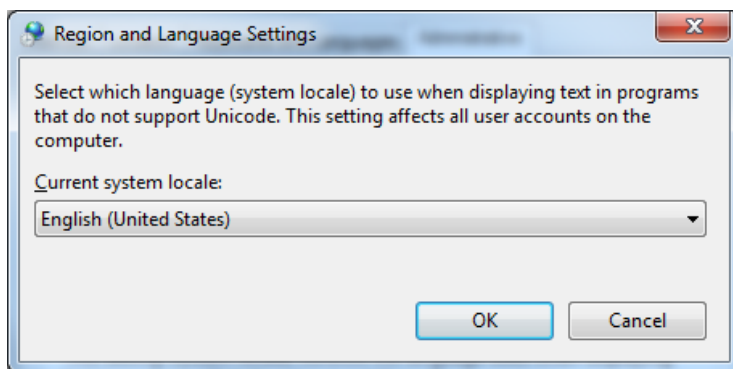
1. Go to the **Windows Start Menu → Control Panel → Region and Language → Location** tab.



2. From the *Current location* drop-down list select the corresponding location, relative to the language you are using.
3. Go to the **Administrative** tab of the same dialog:



4. Click **Change system locale...** A new dialog will be invoked:



5. From the dialog, displayed above, select the language you are using.



## APPENDIX 8 – TITLEBOX KEYBOARD SHORTCUTS

Function	Shortcut
Open project	Ctrl+O
Save project	Ctrl+S
Exit TitleBox	Alt+F4
Undo	Ctrl+Z
Cut selection	Ctrl+X
Copy selection	Ctrl+C
Paste selection	Ctrl+V
Delete	Del
Select All	Ctrl+A
Add to Schedule	Ctrl+H
Copy schedule settings	Ctrl+Alt+C
Paste schedule settings	Ctrl+Alt+V



## PROPERTY TOOL

Add point	A
Align to the previous point	Alt+ left arrow
Align to the next point	Alt +right arrow
Align to the first point	Alt +down arrow
Move to the next point	Ctrl+ left arrow
Move to the previous point	Ctrl+ right arrow
Movement into the object graphics	Left/right/up/down arr.



## Index

### QUICK START

TitleBox, 9

### TITLEBOX

Analogue clock, 40

Animation file, 43

Animation properties, 41

Banner, 46

Browser object, 51

Chat Line, 52

Chat note, 46

Chat Roll, 52

Creating objects, 28

Dataproviders-EAS, 71

Dataproviders-FileLink, 65

Dataproviders-HTML, 70

Dataproviders-ODBC, 67

Dataproviders-RSS, 69

Dataproviders-Weather, 65

Dataproviders-XML, 68

DataSource Manager, 62

Deleting objects, 29

Digital clock, 47

Direct Show media, 44

Dynamic speed, 38

Editing objects, 28

Flash objects, 48

General options, 23

Lists of Items, 84

Menu bar, 17

Network control, 17

Object palette, 15

Objects List, 29

Power Point objects, 49

Primary Shapes, 50

Program Script, 75, 103, 110, 112, 115, 117, 119, 122, 126, 129, 131, 133, 135, 138, 140, 142, 145, 147, 149

Roll&crawl properties, 35

Schedule mode, 53

Screen capture, 51

Set File link, 30

Slide controller, 61

Slide Manager, 55

Sound objects, 47

Still picture properties, 31

System bar, 16

Task Manager, 72

Task Manager-Events, 88

Task Manager-Tasks, 72

TBitmap32, 117

Text field properties, 31

Toolbar, 11

Transform Events, 89



# TitleBox User Manual



TTBDataDistributor, 83, 114

TTBDataProvider, 82, 112

TTBObject, 81, 103, 110

TTBProject, 83, 115

TTBSlide, 82, 110

TTBSlidesManager, 82, 111

TTBStorage, 83, 118

User interface, 10

# TitleBox User Manual



PlayBox Technology UK Ltd  
Brookmans Park Teleport  
Great North Road  
AL96NE Hatfield  
United Kingdom

[info@playboxtechnology.com](mailto:info@playboxtechnology.com)

[www.playboxtechnology.com](http://www.playboxtechnology.com)



All rights reserved copyright © 2004-2020